

**МИНОБРНАУКИ РОССИИ**  
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**  
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**  
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**  
**Кафедра МОЭВМ**

**КУРСОВАЯ РАБОТА**  
по дисциплине «Разработка приложений для мобильных платформ»  
**Тема: Копировщик ключей**

Студент гр. 1303	_____	Гирман А.В.
Студент гр. 1303	_____	Коренев Д.А.
Студент гр. 1303	_____	Смирнов Д.Ю.
Преподаватель	_____	Заславский М.М.

Санкт-Петербург  
2025

## **ЗАДАНИЕ**

Студент Гирман А.В.

Студент Коренев Д.А.

Студент Смирнов Д.Ю.

Группа 1303

Тема: Копировщик ключей

Исходные данные:

Необходимо реализовать Android–приложение, которое позволяет на основании образца металлического ключа, приложив его к экрану, сделать слепок.

Содержание пояснительной записи:

«Содержание»

«Введение»

«Сценарии использования»

«Пользовательский интерфейс»

«Разработанное приложение»

«Выводы»

«Приложения»

«Список использованных источников»

Предполагаемый объем пояснительной записи:

Не менее 15 страниц.

Дата выдачи задания: 05.02.2025

Дата сдачи реферата: 23.03.2025

Дата защиты реферата: 23.03.2025

Студент \_\_\_\_\_ Гирман А.В.

Студент \_\_\_\_\_ Коренев Д.А.

Студент \_\_\_\_\_ Смирнов Д.Ю.

Преподаватель \_\_\_\_\_ Заславский М.М.

## **АННОТАЦИЯ**

В рамках курсовой работы разработано Android-приложение, представляющее собой копировщик ключей, которое позволяет на основании образца металлического ключа, приложив его к экрану, сделать слепок. Возможности приложения: поддержка разных типов ключей, хранение созданных копий, отправка в социальные сети, добавление фотографии ключа, удаление слепка ключа.

Для разработки использованы технологии Kotlin, Jetpack Compose, Android SDK, Hilt, Ksp, Room, Gson, Coil.

Найти исходный находится по ссылке: [Duplikeytor](#)

## **SUMMARY**

As part of the course work, an Android application has been developed, which is a key copier that allows you to make an impression based on a metal key sample by attaching it to the screen. Application features: support for different types of keys, storing created copies, sending to social networks, adding a photo of the key, removing a key impression.

Kotlin, Jetpack Compose, Android SDK, Hilt, Ksp, Room, Gson, Coil technologies were used for the development.

The source code can be found at the link: [Duplikeytor](#)

## СОДЕРЖАНИЕ

1.	Введение	7
1.1.	Актуальность проблемы	7
1.2.	Постановка задачи	7
1.3.	Предлагаемое решение	8
1.4.	Почему решение необходимо реализовать как мобильное приложение	8
2.	Сценарии использования	9
2.1.	Сценарий «Создание профиля ключа»	9
2.2.	Сценарий «Создание профиля ключа»	9
2.3.	Сценарий «Просмотр информации разработчиков»	10
3.	Пользовательский интерфейс	11
3.1.	Макет интерфейса с графиком переходов	11
3.2.	Целевые устройства, обоснование требований и максимально подробные характеристики	11
4.	Разработанное приложение	14
4.1	Краткое описание	14
4.2	Схема архитектуры	14
4.2.1	Используемые технологии	14
4.2.2	Архитектура приложения	15
4.3	Использованные библиотеки	18
5.	Выводы	21А
5.1	Достигнутые результаты	21
5.2	Недостатки и пути для улучшения	21
5.3	Будущее развитие решения	21
6.	Литература	22
7.	Приложения	23
7.1	Инструкция для пользователя	23



## **1. ВВЕДЕНИЕ**

### **1.1. Актуальность проблемы**

Актуальность приложения для создания электронных слепков ключей заключается в удобстве и экономии времени для пользователей. Традиционно процесс создания дубликата ключа требует минимум двух визитов к мастеру: сначала для передачи оригинала, потом для получения копии. Это может быть неудобно и затратно по времени. С помощью приложения пользователи могут делать электронные слепки ключей в любом месте — дома, на работе, и сразу отправлять слепок мастеру через социальные сети. Это исключает необходимость лично посещать мастерскую, экономя время и усилия. Более того, пользователи не остаются без ключа, поскольку оригинал остается у них. Также, возможность отправки слепка любому мастеру, независимо от его местоположения, и получения готового ключа с доставкой, открывает доступ к услугам лучших специалистов. Это делает процесс получения копий быстрым, удобным и максимально адаптивным к требованиям современного пользователя.

### **1.2. Постановка задачи**

Задача проекта заключается в разработке Android-приложения, которое позволяет пользователям создавать электронные слепки ключей, прикладывая их к экрану устройства. Приложение должно предоставлять следующие возможности:

- Создание слепка ключа: пользователь может сделать слепок ключа выбранного типа.
- Добавление названия и фотографии: возможность дать название слепку ключа и фотографии ключа
- Хранение и управление созданными копиями: созданные копии и информация о них сохраняется в постоянное хранилище. Пользователь может удалить желаемую копию.

- Редактирование копии: пользователь может изменить слепок, дать ему другое название, добавить, удалить или изменить фотографию копии.
- Отправка копии в социальные сети: приложение должно позволять отправлять созданные слепки напрямую через социальные сети и другие мессенджеры.

Кроме того, приложение должно быть интуитивно понятным и минималистичным, чтобы пользователи могли легко освоить его функционал без обучения и быстро выполнять необходимые операции.

### **1.3. Предлагаемое решение**

Разработать Android-приложение, которое позволит пользователям создавать и хранить электронные слепки ключей, отправлять их через социальные сети и управлять своими копиями. Приложение будет поддерживать различные типы ключей, обеспечивая при этом интуитивно понятный интерфейс. Для реализации функционала будет использован ряд современных технологий, внутреннее хранилище на основе базы данных для сохранения данных о ключах, включая их фотографии

### **1.4. Почему решение необходимо реализовать как мобильное приложение**

- Мобильные устройства всегда находятся под рукой, что позволяет пользователю быстро создать и отправить электронный слепок ключа в любое время и из любого места.
- Мобильные платформы предоставляют простые инструменты для интеграции с различными социальными сетями и мессенджерами, что упрощает отправку слепков напрямую.
- Приложение может работать локально без необходимости постоянного доступа к интернету, что обеспечивает его функциональность в любых условиях.

## **2. СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ**

### **2.1. Сценарий «Создание профиля ключа»**

**Действующее лицо:**

Пользователь

**Основной сценарий:**

1. Пользователь открывает приложение.
2. Пользователь выбирает формат ключа.
3. Пользователь нажимает на кнопку «Выбрать».
4. Пользователь настраивает размер ключа с помощью специального ползунка.
5. Пользователь нажимает кнопку «Продолжить».
6. Пользователь выбирает нужные пины и настраивает глубину выреза.
7. Пользователь нажимает кнопку «Продолжить».
8. Пользователь вводит имя профиля ключа.
9. Пользователь нажимает кнопку «Сохранить».
10. Пользователь переходит к шагу 2.

**Альтернативные сценарии:**

- Пользователь на шаге 6 может скрыть кнопки и текст названия окна
- Пользователь не вводит имя профиля ключа, дается имя по умолчанию
- Пользователь прикрепляет фотографию ключа
- Пользователь нажимает на кнопку «Поделиться» и выбирает соц. сеть для отправки.
- Пользователь решил отменить создание нового профиля ключа и вернуться на главный экран.
- Пользователь решил перейти к экрану «Мои ключи»
- Пользователь решил перейти к экрану «О нас»

### **2.2. Сценарий «Создание профиля ключа»**

**Действующее лицо:**

Пользователь

**Основной сценарий:**

1. Пользователь открывает приложение.
2. Пользователь переходит к экрану «Мои ключи».
3. Пользователь выбирает профиль ключа для просмотра.
4. Пользователь переходит к экрану «Информация о ключе».

**Альтернативные сценарии:**

- Пользователь нажимает на кнопку «Изменить» и переходит к промежуточному этапу 6 сценария «Создание профиля ключа».
- Пользователь нажимает на кнопку «Удалить» и удаляет профиль ключа.
- Пользователь нажимает на кнопку «Поделиться» и выбирает соц. сеть для отправки.

### **2.3. Сценарий «Просмотр информации разработчиков»**

**Действующее лицо:**

Пользователь

**Основной сценарий:**

1. Пользователь открывает приложение.
2. Пользователь переходит к экрану «О нас».
3. Пользователь читает информацию о разработчиках приложения.

### 3. ПОЛЬЗОВАТЕЛЬСКИЙ ИНТЕРФЕЙС

#### 3.1. Макет интерфейса с графиком переходов в темной и светлой темах

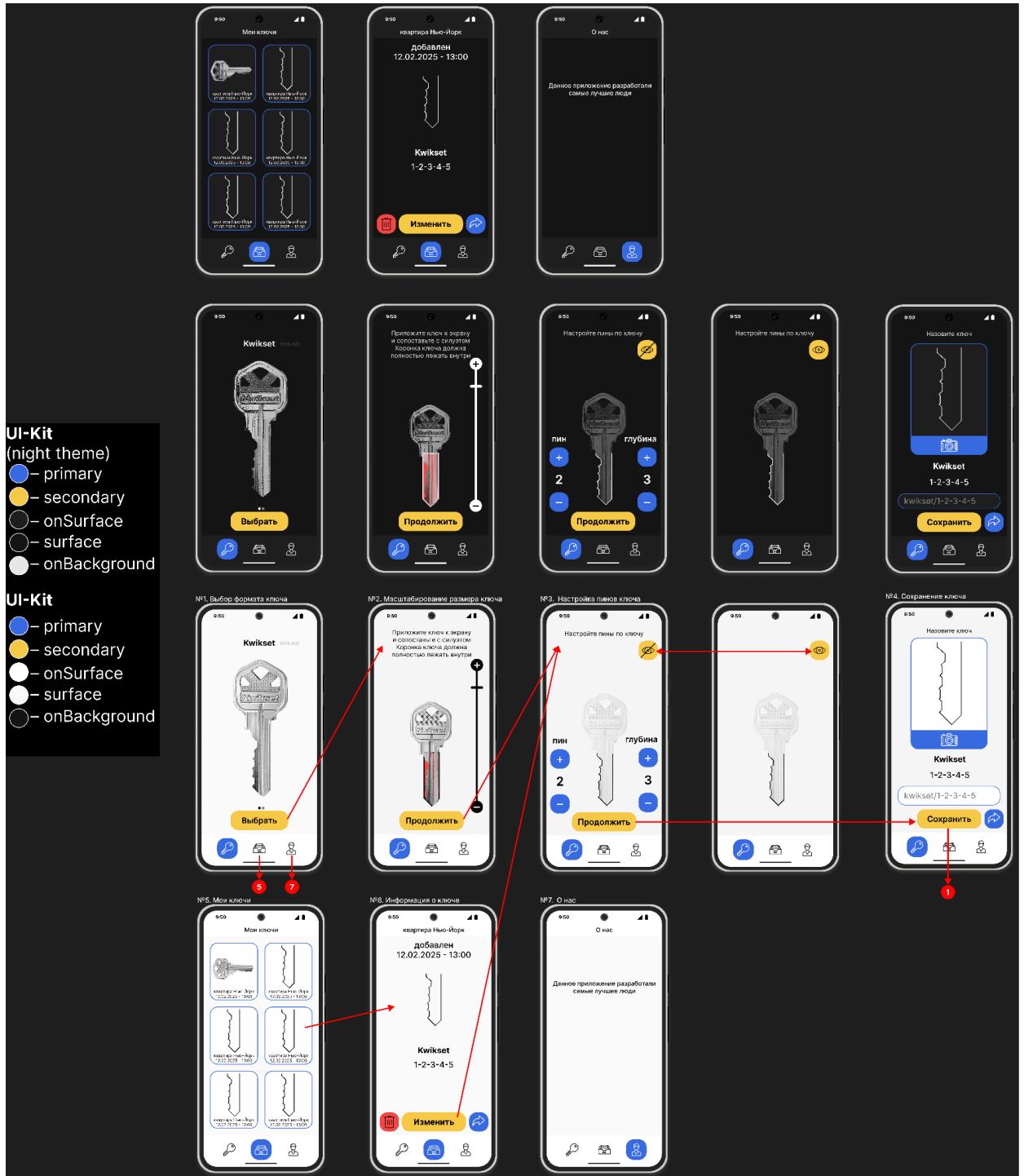


Рисунок 1 – Макет интерфейса с графиком переходов

#### 3.2 Целевые устройства, обоснование требований и максимально подробные характеристики

Тип устройства: Смартфон

Модель: Google Pixel 9 Pro

Характеристики:

Версия API 35

Экран:

hw.lcd.density 480

hw.lcd.height 2856

hw.lcd.width 1280

Остальное:

avd.ini.displayname Pixel 9 Pro API 35

avd.ini.encoding UTF-8

AvdId Pixel\_9\_Pro\_API\_35

disk.dataPartition.size 2G

fastboot.chosenSnapshotFile

fastboot.forceChosenSnapshotBoot no

fastboot.forceColdBoot no

fastboot.forceFastBoot yes

hw.accelerometer yes

hw.arc false

hw.audioInput yes

hw.battery yes

hw.camera.back virtualscene

hw.camera.front emulated

hw.cpu.ncore 2

hw.device.hash2 MD5:73e7b35d09e3a8055043aca4688e0dad

hw.device.manufacturer Google

hw.device.name pixel\_9\_pro

hw.dPad no

hw.gps yes

hw.gpu.enabled yes

hw.gpu.mode auto

hw.initialOrientation portrait  
hw.keyboard yes  
hw.mainKeys no  
hw.ramSize 11548  
hw.sdCard yes  
hw.sensors.orientation yes  
hw.sensors.proximity yes  
hw.trackBall no  
image.androidVersion.api 35  
image.sysdir.1  
system-images/android-35/google\_apis\_playstore/x86\_64/  
PlayStore.enabled true  
runtime.network.latency none  
runtime.network.speed full  
showDeviceFrame yes  
skin.dynamic yes  
tag.display Google Play  
tag.displaynames Google Play  
tag.id google\_apis\_playstore  
tag.ids google\_apis\_playstore  
vm.heapSize 256

## **4. РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ**

### **4.1 Краткое описание**

Разработанное приложение представляет собой Android-решение для создания, хранения и управления электронными слепками ключей.

Пользователи могут создавать электронные копии своих ключей, прикладывая металлический ключ к экрану устройства, что делает процесс создания слепка простым и доступным. Все слепки хранятся на устройстве, обеспечивая надежное хранение и быстрый доступ к данным.

Интерфейс приложения интуитивно понятен: пользователи могут легко систематизировать и управлять своими электронными слепками ключей, добавляя, удаляя или редактируя информацию о каждом ключе. В приложении предусмотрены возможности для быстрой отправки электронных слепков через социальные сети и мессенджеры, что упрощает процесс передачи ключей другим людям.

Данные о копиях ключей пользователя хранятся локально в базе данных на устройстве и не требует подключения к интернету для работы с ним.

Помимо основной функциональности, приложение включает страницу «О нас», где пользователи могут узнать информацию о разработчиках.

### **4.2 Схема архитектуры**

#### **4.2.1 Используемые технологии**

- **Язык программирования:** Kotlin
- **Генерация Kotlin кода:** KSP
- **UI:** Jetpack Compose
- **Работа с асинхронным отображением картинок:** Coil
- **Dependency Injection библиотека:** Hilt
- **База данных:** Room
- **Сериализация и десериализация Java-объектов:** Gson
- **Набор инструментов для разработки приложений:** Android SDK

## **4.2.2 Архитектура приложения**

### **Data слой**

Содержит в себе единственный абстрактный класс базы данных приложения.

### **Domain слой**

Содержит в себе конвертеры типов данных, Dao интерфейс с описанием методов базы данных, data класс модели ключа, класс–репозиторий ключей с suspend методами использующие Dao экземпляр для доступа к базе данных, класс–репозиторий изображений ключей для копирования фотографий пользователя и последующего удаления копий при необходимости.

### **Presentation слой**

- **ui**
  - uikit с описанием основных, часто используемых ui–элементов приложения: кнопки, диалоговое окно, тема приложения, которая включает в себя Color.kt, Theme.kt, Type.kt – определяют текстовые стили и цветовую схему интерфейса.
  - utils содержит в себе методы расширения для удобства использования
- **shared.models**
  - KeyType – тип ключа с его названием и id ресурса изображения
  - Screen интерфейс – описание методов, которые должен реализовать класс наследник, который включает в себя логику экрана
- **holder**
  - MainActivity – единственная Activity приложения, которая управляет жизненным циклом приложения. Имеет единственное поле – viewModel – экземпляр MainViewModel и единственный переопределенный метод ComponentActivity

- onCreate, в котором устанавливается безрамочный режим и отображается MainScreen (Composable функция) в которую передаются основные лямбда функции из viewModel.
- MainScreen отображает панель навигации, статус бар и NavHost, в котором определяется какой экран контента отображать. Тут же определяются ViewModel трех экранов контента: CreateViewModel, ArchiveViewModel, AboutViewModel, которые регистрируют себя в MainViewModel. Для делегации логики навигации используется NavigationHandler.
- NavigationHandler создан для упрощения работы с навигацией приложения: переход с одного экрана на другой, определение экрана, при нажатии пользователем системной кнопки «назад».
- MainViewModel определяет в себе логику отображения текущего состояния навигации: какой экран отображается, какие состояния должны быть у навигационной панели и статус бара.
- StatusBar и NavigationBar – верстка навигационной панели и статус бара.
- AppEvent – определяем в себе события приложения для передачи их своевременного и корректного реагирования экранов не зависящих от других экранов.

- **create**

- CreateFragment – отображает нужный экран создания ключа и определяет работу BackHandler.
- CreateViewModel – отвечает за логику создания ключа, хранение корректного состояния создания ключа, доступ к данным из базы данных и сохранение данных в базу данных. Наследуется от ViewModel и реализует интерфейс Screen.

- Экраны создания ключа: ChooseScreen, ScaleScreen, CreateScreen, SaveScreen отвечают за отображение соответствующего состояния экрана создания ключей и передачи во ViewModel событий от пользователя.
- Key – кастомное отображение ключа и пинов в зависимости от его типа.
- KeyConfig – определяет конфигурации типов ключей
- CreateScreenState – состояния экрана создания ключа
- CreateEvent – определяет события экрана создания ключа
- CreateStep – определяет типы экранов создания ключа

- **archive**

- ArchiveFragment – отображает нужный экран архива ключей и определяет работу BackHandler.
- ArchiveViewModel – определяет состояние экрана, логику удаления ключа, получение списка ключей из базы данных и одного конкретного ключа. Наследуется от ViewModel и реализует интерфейс Screen.
- ArchiveScreen – экран списка ключей, отображает вертикальную сетку из карточек ключей, которые создал пользователь, либо текст об отсутствии ключей, если пользователь не создали ни одного ключа. Элемент сетки ключа – KeyCard.
- KeyInfoScreen – экран информации ключа, на котором находится слепок ключа или фотография, если она есть, кнопки удаления, изменения и поделиться, также отображается информация о названии и даты создания ключа.
- KeyState – модель ключа и информации о нем.
- KeyArchiveState – состояния экрана архива ключей.
- KeyArchiveEvent – события экрана архива ключей.

- **about**

- AboutFragment – отображает экран «О нас» и определяет работу BackHandler.
- AboutScreen – верстка экрана «О нас» с карточками разработчиков. По нажатию на карточку открывается GitHub аккаунт разработчика.
- AboutViewModel – ViewModel экрана отвечает за хранение статичных данных о разработчиках. Наследуется от ViewModel и реализует интерфейс Screen.

### **DI (dependency injection)**

Содержит в себе интерфейс AppComponent и объект AppModule, в котором описаны необходимые provide–методы.

### **Application класс**

Класс Application наследуемый от системного класса Application, проаннотирован HiltAndroidApp для корректного внедрения инъекции зависимости с помощью библиотеки Hilt

## **4.3 Использованные библиотеки**

- Core и жизненный цикл
  - androidx.core:core-ktx: Расширение для Android Core API, упрощающее использование API Android с помощью Kotlin.
  - androidx.lifecycle:lifecycle-runtime-ktx: Предоставляет компоненты для реализации жизненного цикла Android-приложений, упрощая управление жизненным циклом компонентов.
- Compose UI Framework
  - androidx.activity:activity-compose: Обеспечивает поддержку Activity в Jetpack Compose.

- androidx.compose.ui:ui: Базовый модуль, обеспечивающий основы для работы с Jetpack Compose.
  - androidx.compose.ui:ui.graphics: Модуль для работы с графикой, цветами и рисованием в Compose.
  - androidx.compose.material3:material3: Используется для интеграции компонентов Material Design 3 в интерфейсе приложения.
  - androidx.navigation:navigation-compose: Реализует навигацию внутри приложения с использованием Compose.
  - androidx.lifecycle:lifecycle-viewmodel-compose: Интеграция ViewModel в Compose, позволяющая использовать модель жизненного цикла для управления состоянием.
- Вспомогательные библиотеки Compose
    - androidx.compose.ui:ui.tooling.preview: Предоставляет инструменты для предпросмотра компонентов Compose в Android Studio.
    - androidx.compose.ui:ui.tooling: Позволяет переносить Compose UI между разными сборками приложения и управлять иерархией UI.
  - Coil
    - coil-compose: Библиотека для асинхронной загрузки и отображения изображений в Jetpack Compose.
    - coil-network.okhttp: Поддержка сети через библиотеку OkHttp для работы Coil.
  - Room БД
    - androidx.room:room-runtime: Обеспечивает основные классы для работы с реляционной базой данных.

- androidx.room:room-ktx: Расширения Kotlin для Room, упрощающие разработку.
  - androidx.room:room-compiler: Компилятор аннотаций для генерации кода доступ к данным.
  - androidx.lifecycle:lifecycle-viewmodel-compose: Обеспечивает ViewModel для управления данными в Jetpack Compose.
- Dagger-Hilt dependency injection
    - hilt-android: Фреймворк для внедрения зависимостей в Android-приложения.
    - hilt-android.compiler: Компилятор аннотаций для поддержки Hilt.
    - androidx.hilt.navigation.compose: Интеграция навигации с Hilt в Compose.
  - Gson
    - gson: Библиотека для сериализации и десериализации Java-объектов в JSON и обратно.

## **5. ВЫВОДЫ**

### **5.1 Достигнутые результаты**

В ходе работы было разработано мобильное приложение «Duplikeytor», позволяющее создавать электронные слепки ключей, прикладывая их к экрану устройства. В приложении доступны функции: добавление названия и фотографии для созданного слепка, хранение и управление созданными копиями, а также редактирование и отправка ключа в социальные сети.

### **5.2 Недостатки и пути для улучшения**

- 1) На данный момент приложение не предоставляет возможности поиска ключей, что затрудняет пользователю быстро находить нужный ключ среди большого количества сохраненных. Для решения этой проблемы необходимо добавить функциональность поиска ключей по названию или другим параметрам.
- 2) Приложение в настоящее время поддерживает только два типа ключей, что ограничивает его универсальность для пользователей, которым может потребоваться управление другими типами ключей. Для решения этой проблемы необходимо реализовать поддержку дополнительных типов ключей, чтобы удовлетворить более широкий круг пользователей.

### **5.3 Будущее развитие решения**

В качестве развития решения возможны:

- 1) Локализация для поддержки нескольких языков.
- 2) Внедрение виджета карты для быстрого поиска ближайшего места изготовления ключей.

## **6. ЛИТЕРАТУРА**

1. Ссылка на GitHub. – [Электронный ресурс]. – URL: <https://github.com/moevm/adfmp1h25-keys> (дата обращения: 05.02.2025).
2. Android Studio. – [Электронный ресурс]. – URL: <https://developer.android.com/studio> (дата обращения: 03.02.2025).
3. Kotlin. – [Электронный ресурс]. – URL: <https://kotlinlang.org/> (дата обращения: 03.02.2025).
4. Jetpack Compose. – [Электронный ресурс]. – URL: <https://developer.android.com/jetpack/compose> (дата обращения: 04.02.2025).
5. Библиотека Room. – [Электронный ресурс]. – URL: <https://developer.android.com/training/data-storage/room> (дата обращения: 15.02.2025).

## **7. ПРИЛОЖЕНИЯ**

### **7.1 Инструкция для пользователя**

#### **Создание нового слепка ключа**

Для того, чтобы создать новый слепок, необходимо выбрать тип ключа на стартовой странице. После этого осуществляется переход на страницу с масштабированием, на данной странице нужно настроить масштаб, для определения размера ключа, таким образом, чтобы не было видно красной зоны за приложенным ключом. На следующем шаге нужно настроить глубину пинов, так чтобы они описывали форму вашего ключа. На странице с сохранением слепка, можно задать название ключа, а также прикрепить любую фотографию из галереи телефона.

#### **Просмотр слепков**

Пользователь может просматривать сохраненные ключи на странице архива. Новые ключи автоматически добавляются в конец списка. При нажатии на любой ключ отображается детальная информация о нем, а также доступны функции редактирования, удаления или отправки ключа через социальные сети.

#### **Удаление слепка**

Для того, чтобы удалить слепок, необходимо перейти в архив, выбрать нужный ключ, открыть его страницу и нажать на красную кнопку с изображением мусорного ведра, затем подтвердить удаление.

## 7.2 Снимки экрана приложения

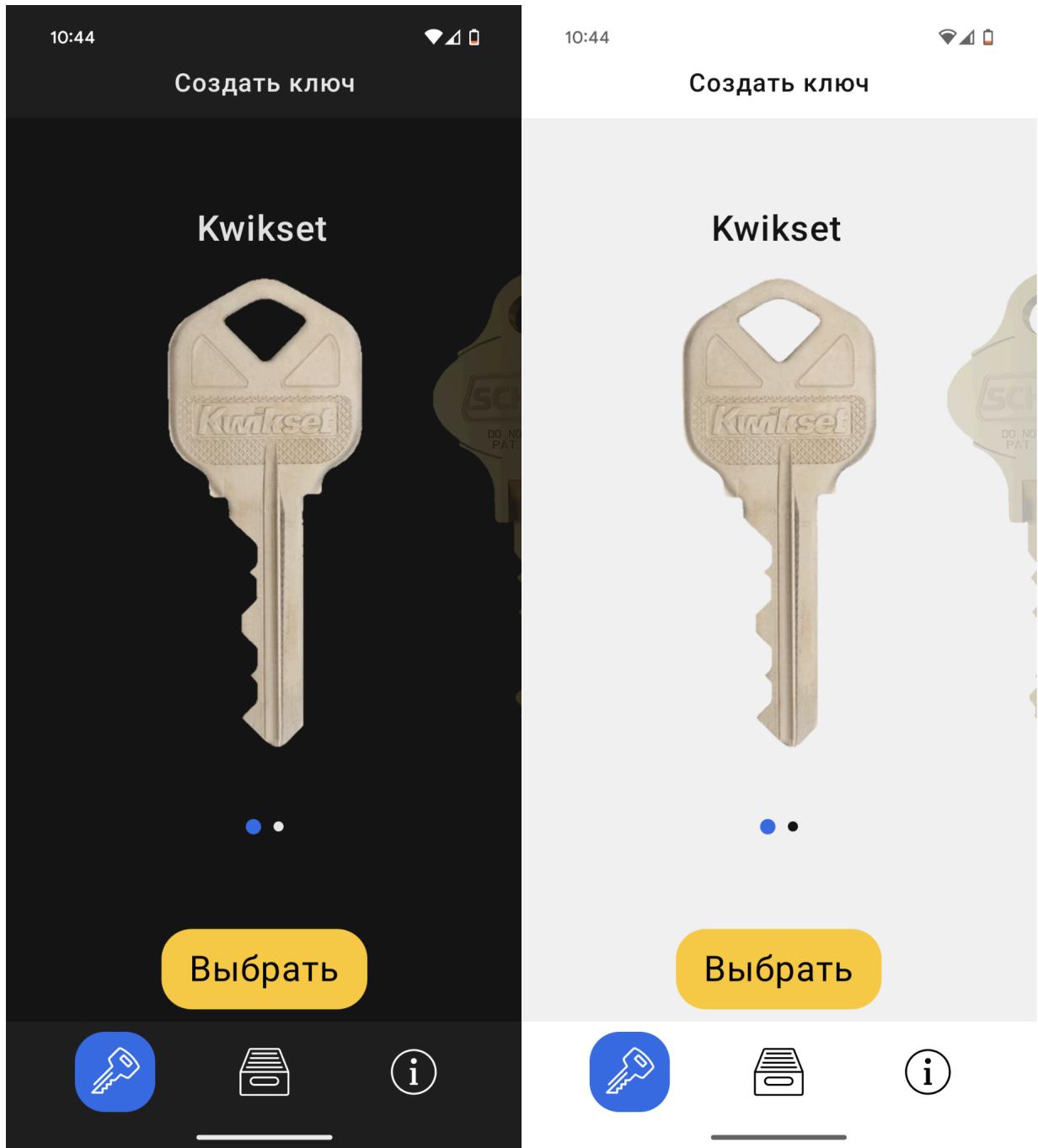


Рисунок 2 – Экран выбор типа ключа

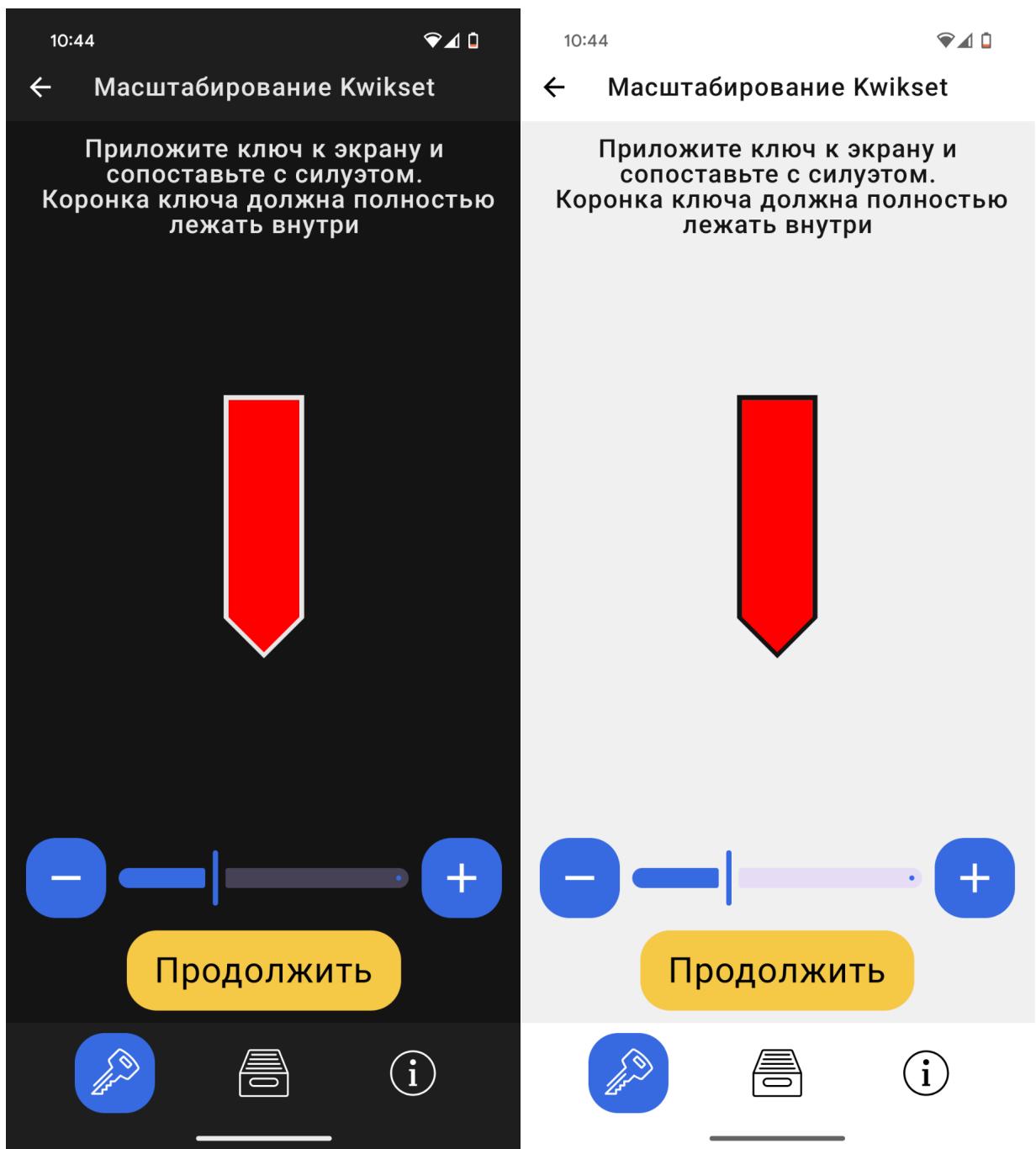


Рисунок 3 – Экран масштабирования копии с ключем

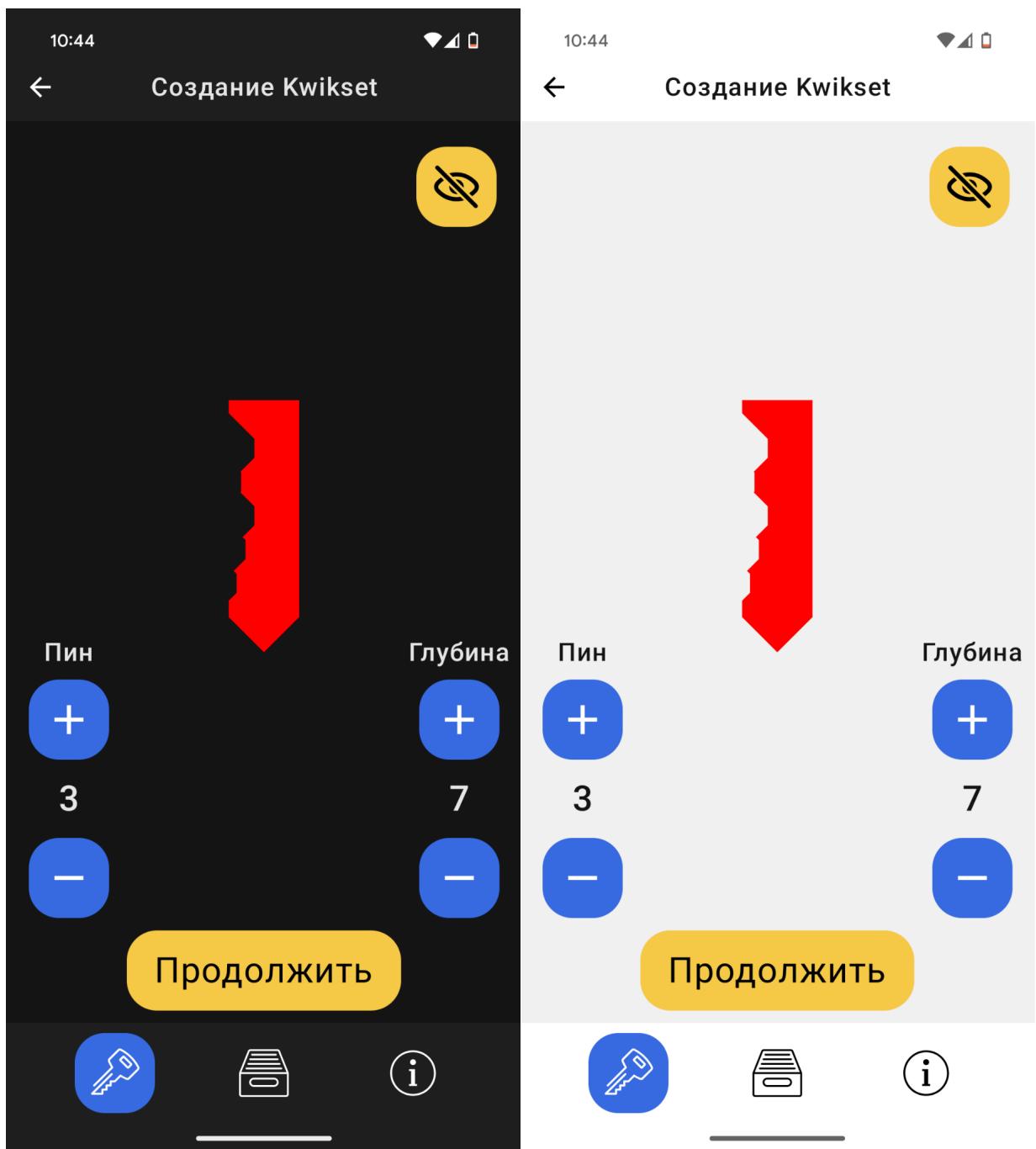


Рисунок 4 – Экран копирования пинов ключа

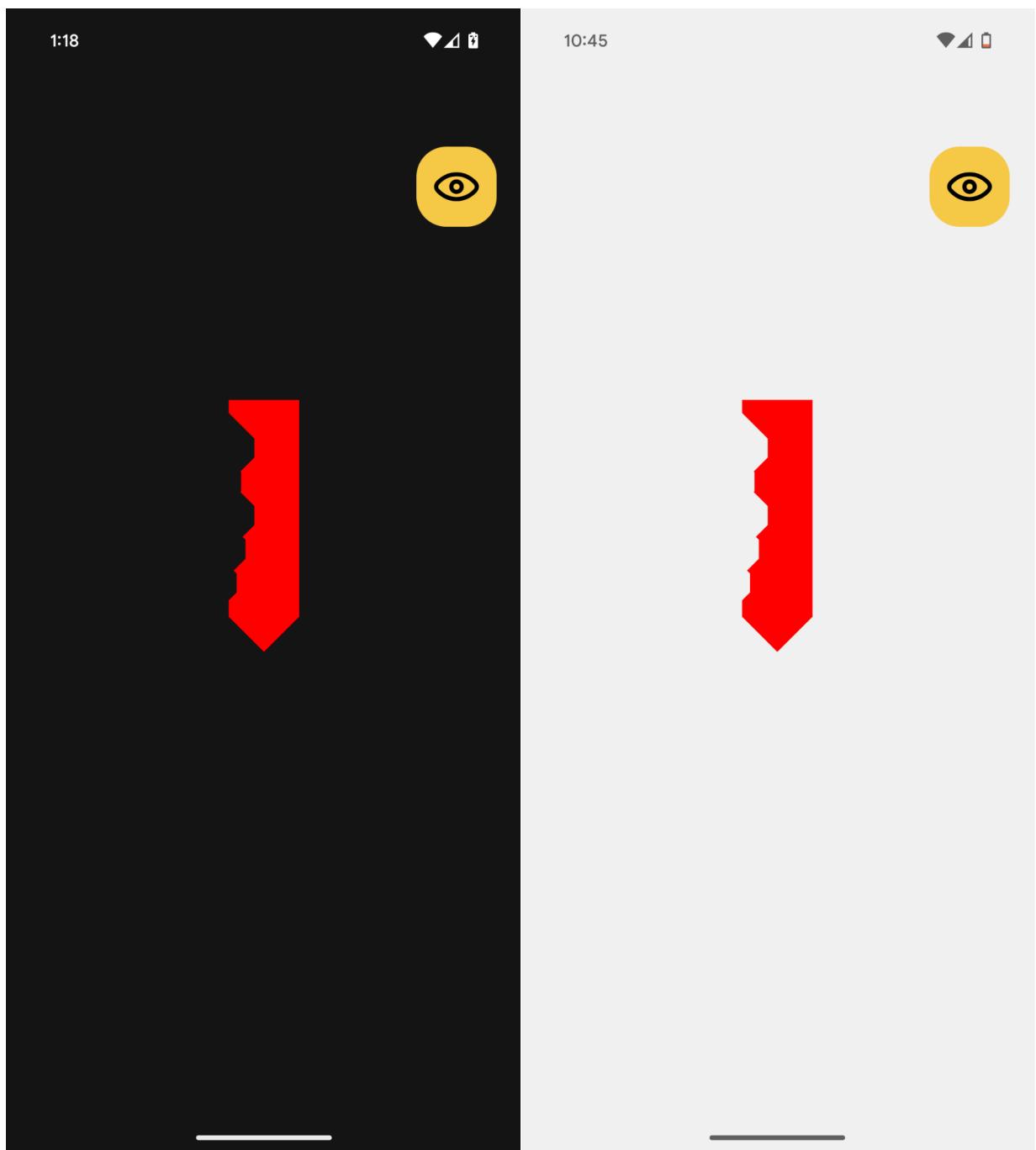


Рисунок 5 – Экран копирования пинов ключа, скрытый интерфейс

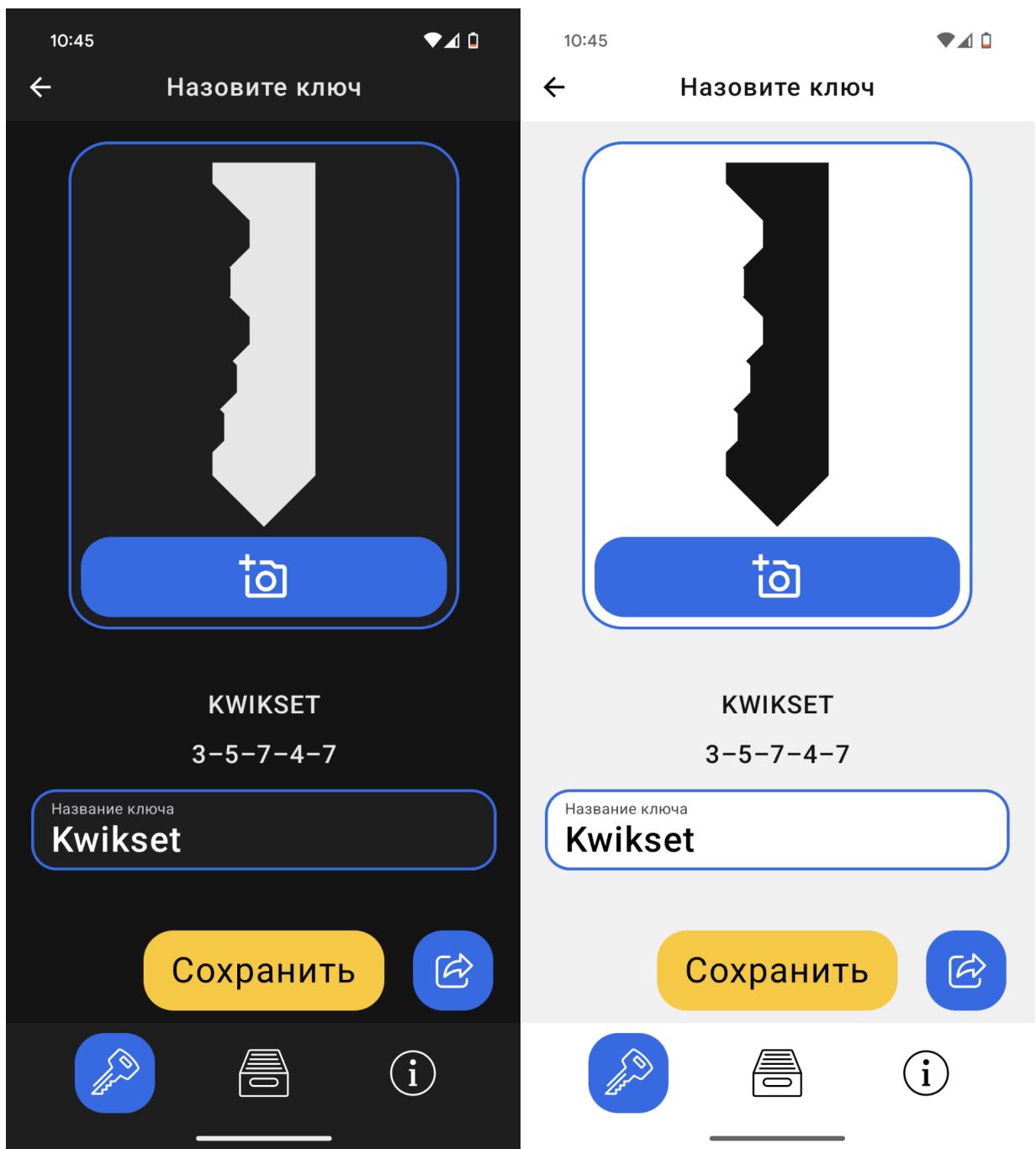


Рисунок 6 – Экран сохранения ключа

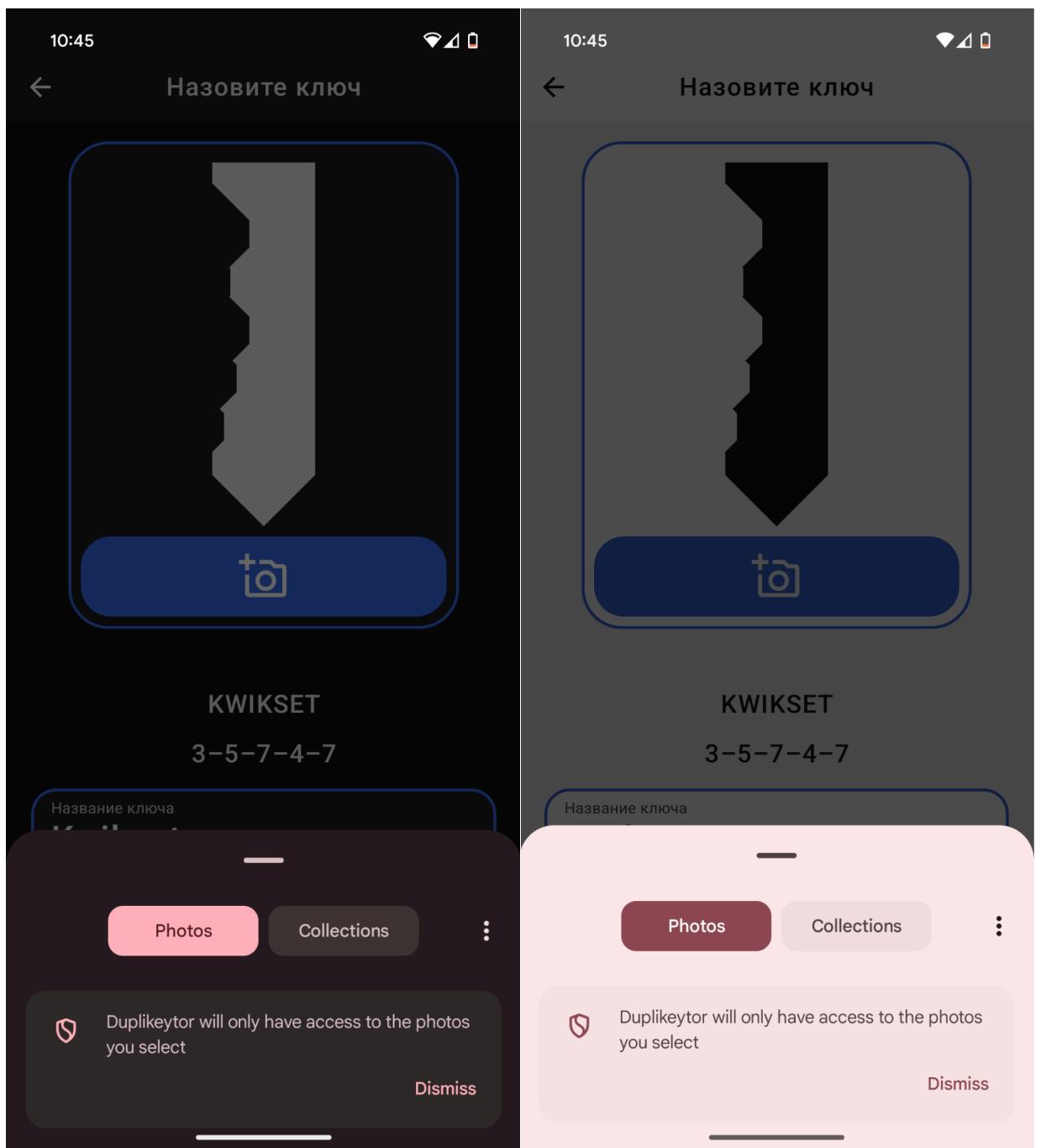


Рисунок 7 – Выбор фотографии ключа

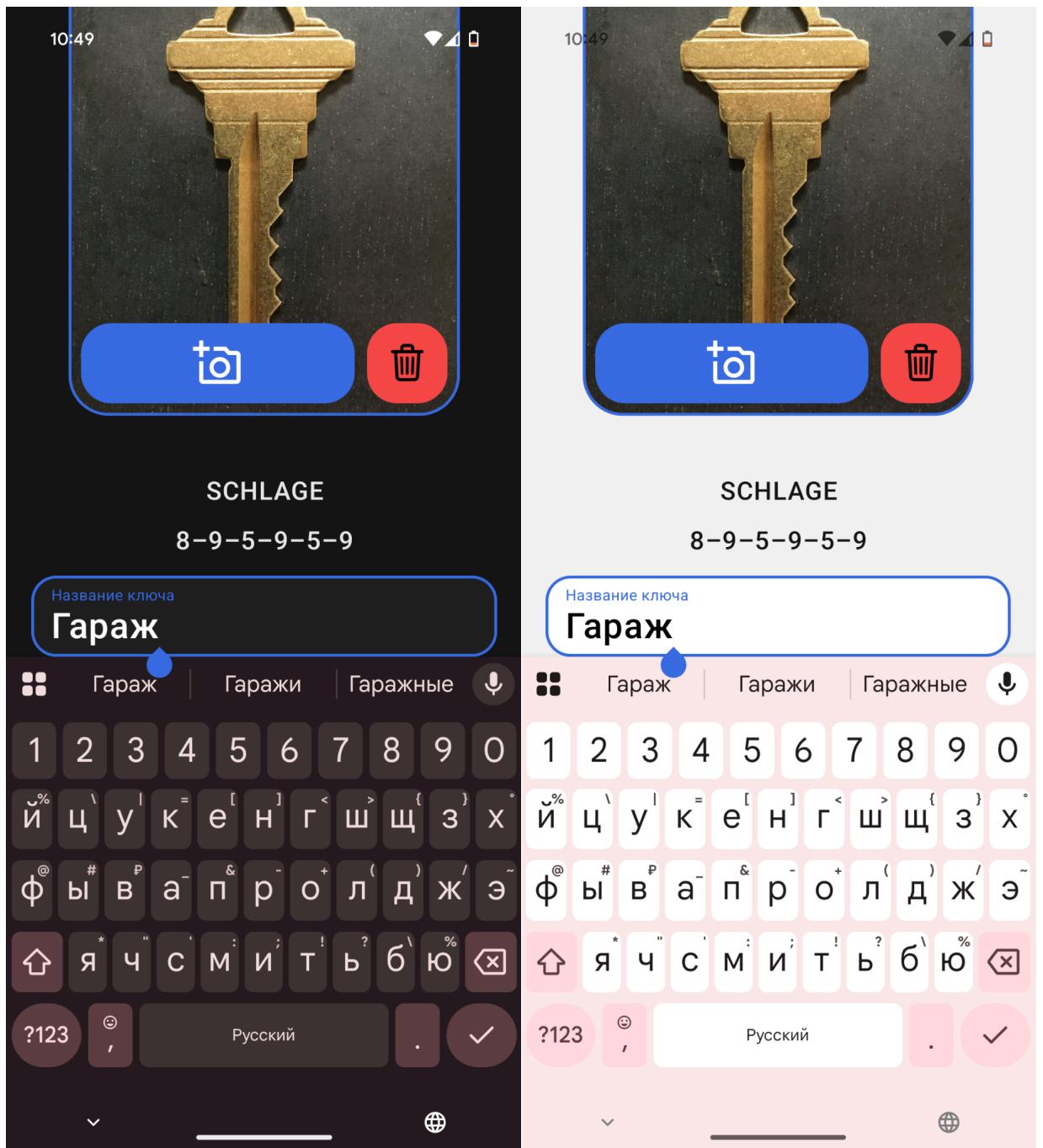


Рисунок 8 – Ввод названия ключа

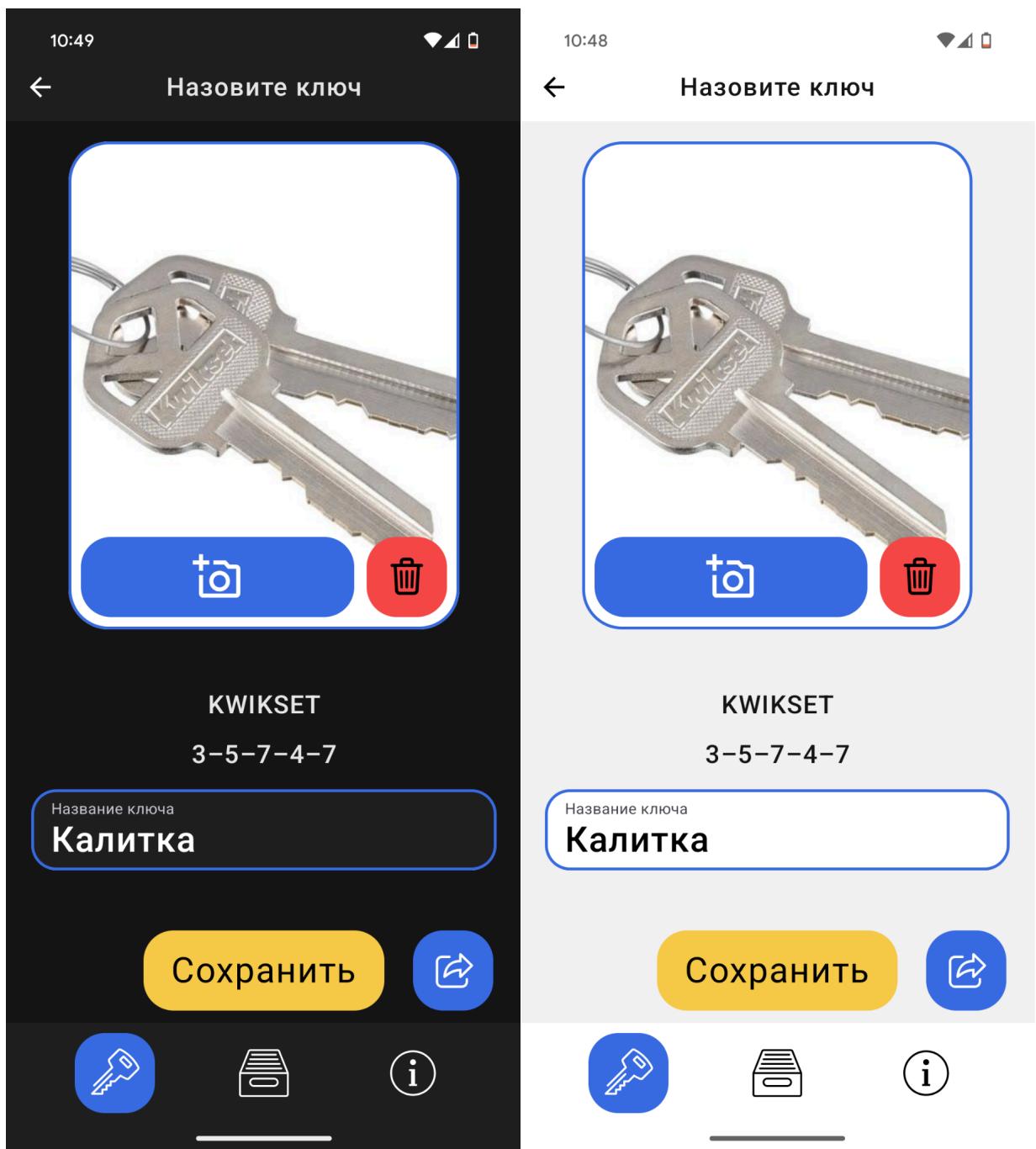


Рисунок 9 – Экран сохранения ключа с фотографией

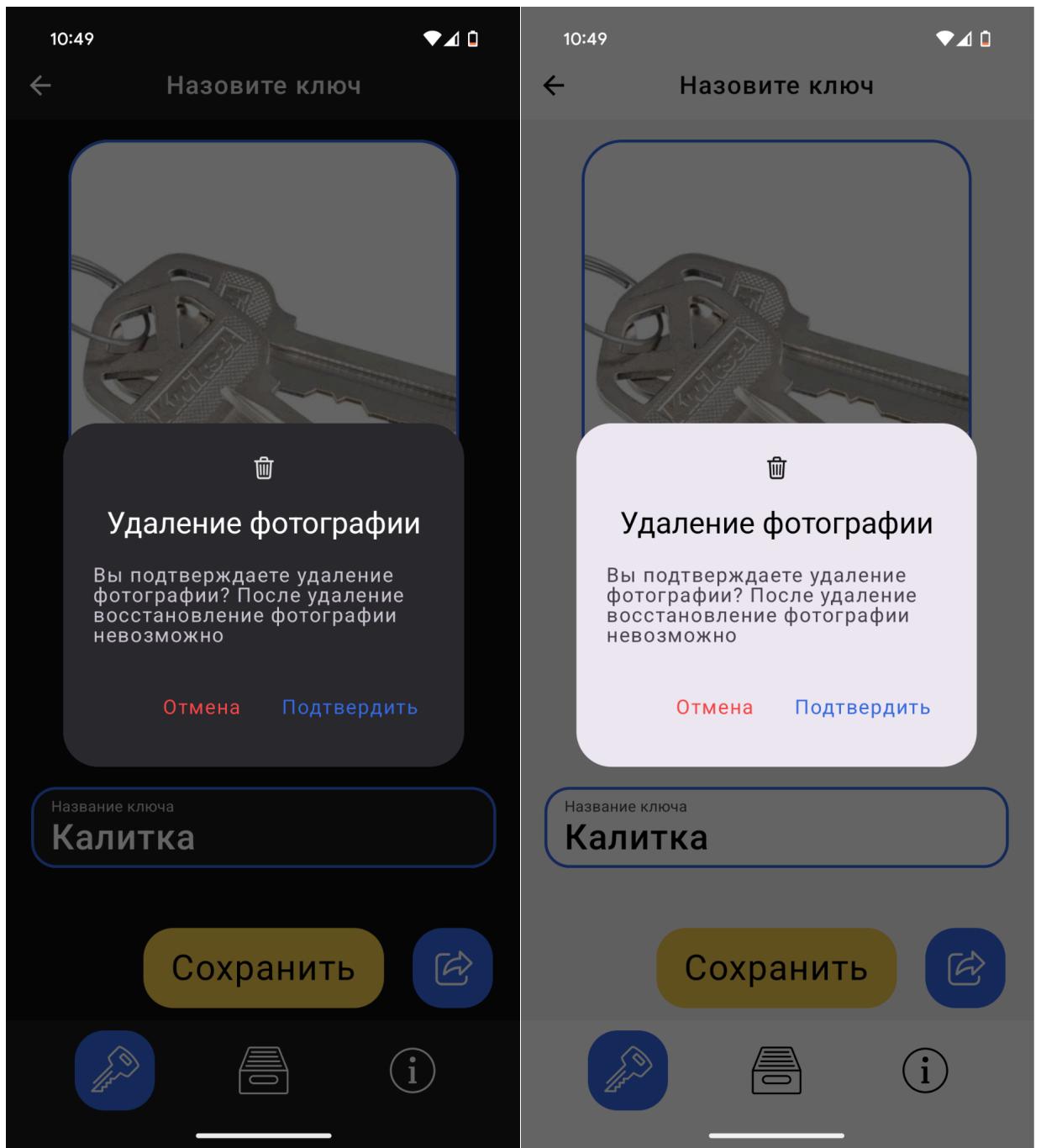


Рисунок 10 – Подтверждение удаления фотографии

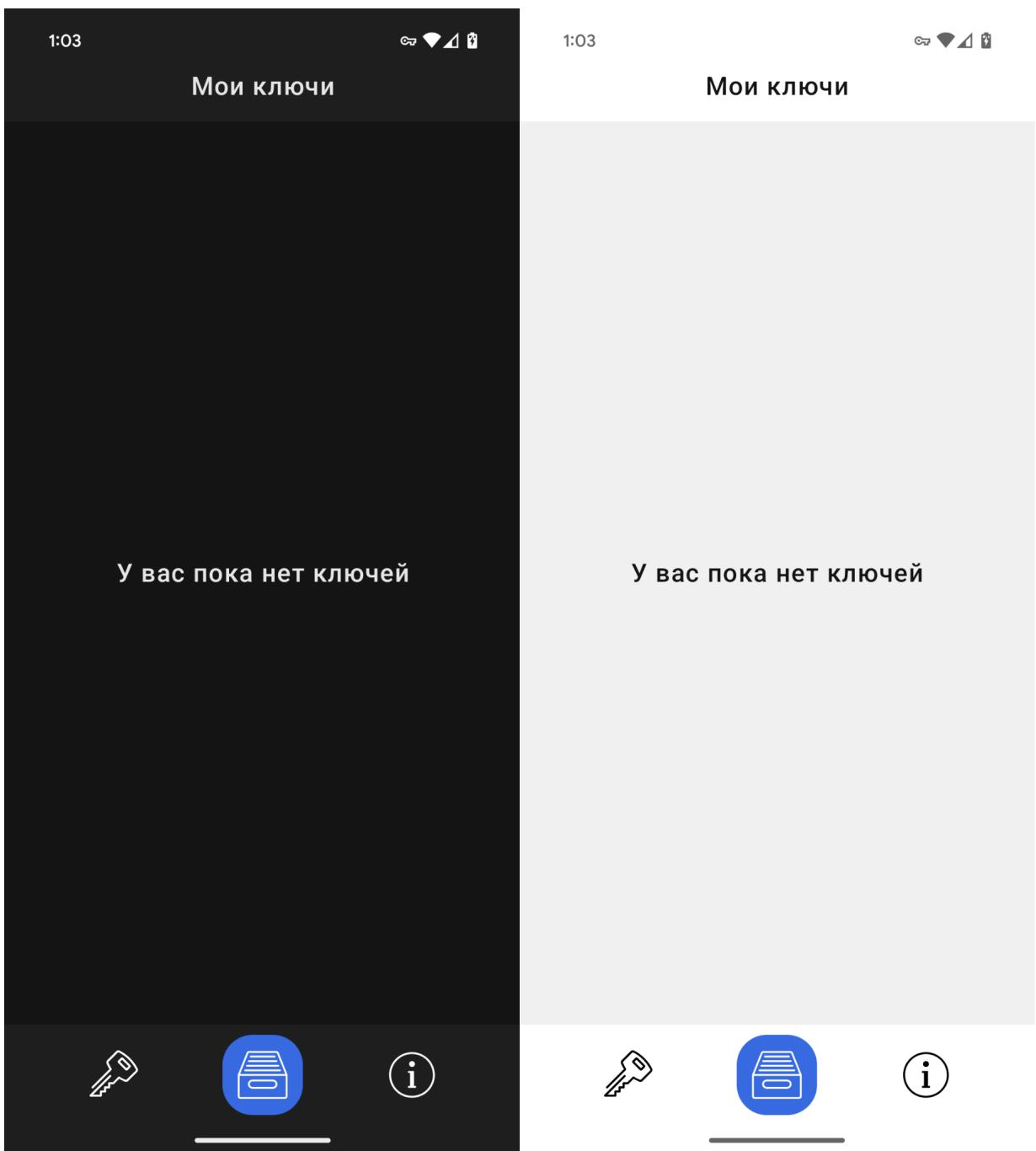


Рисунок 11 – Экран списка созданных копий, пустой

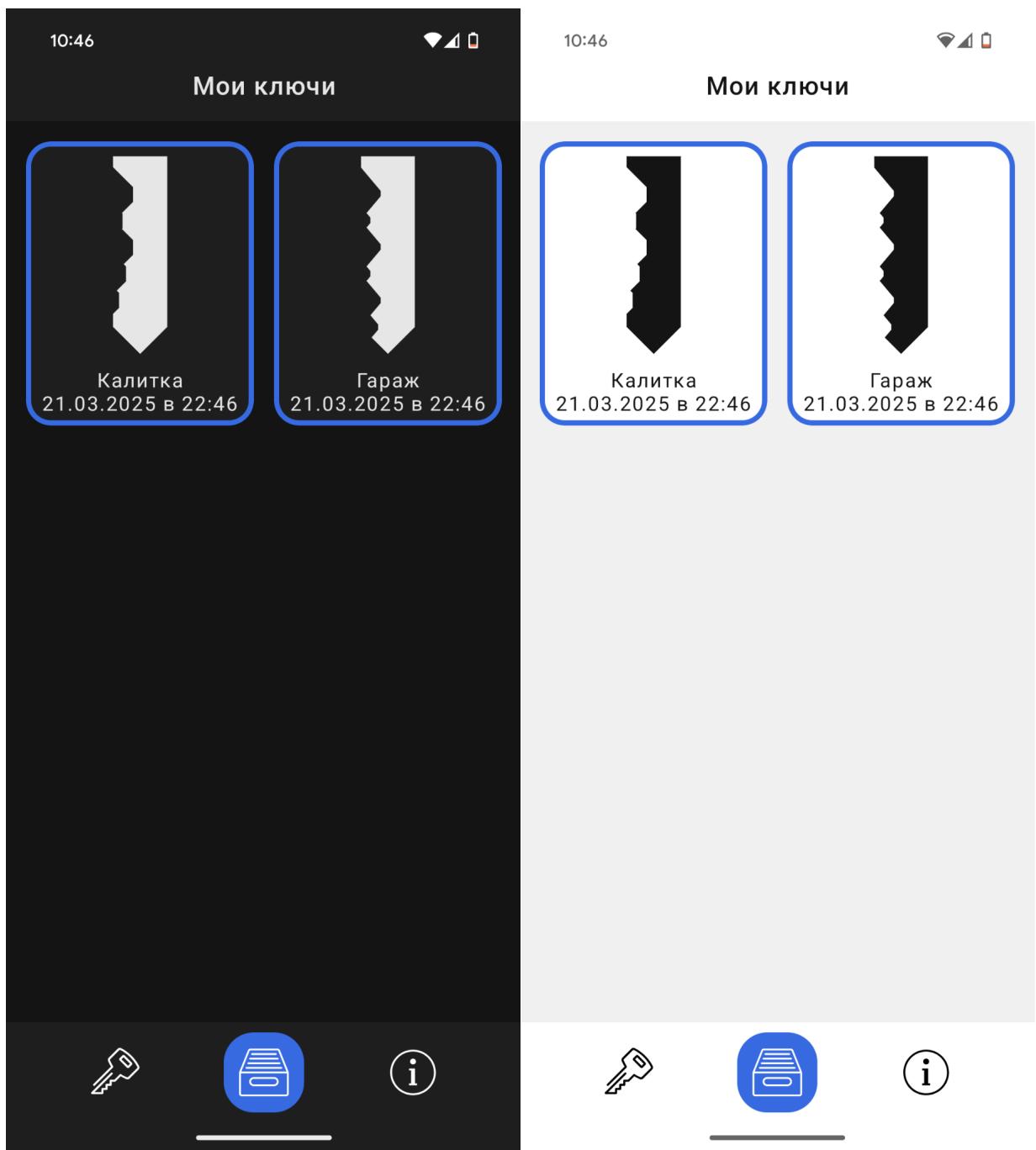


Рисунок 12 – Экран списка созданных копий без фотографий

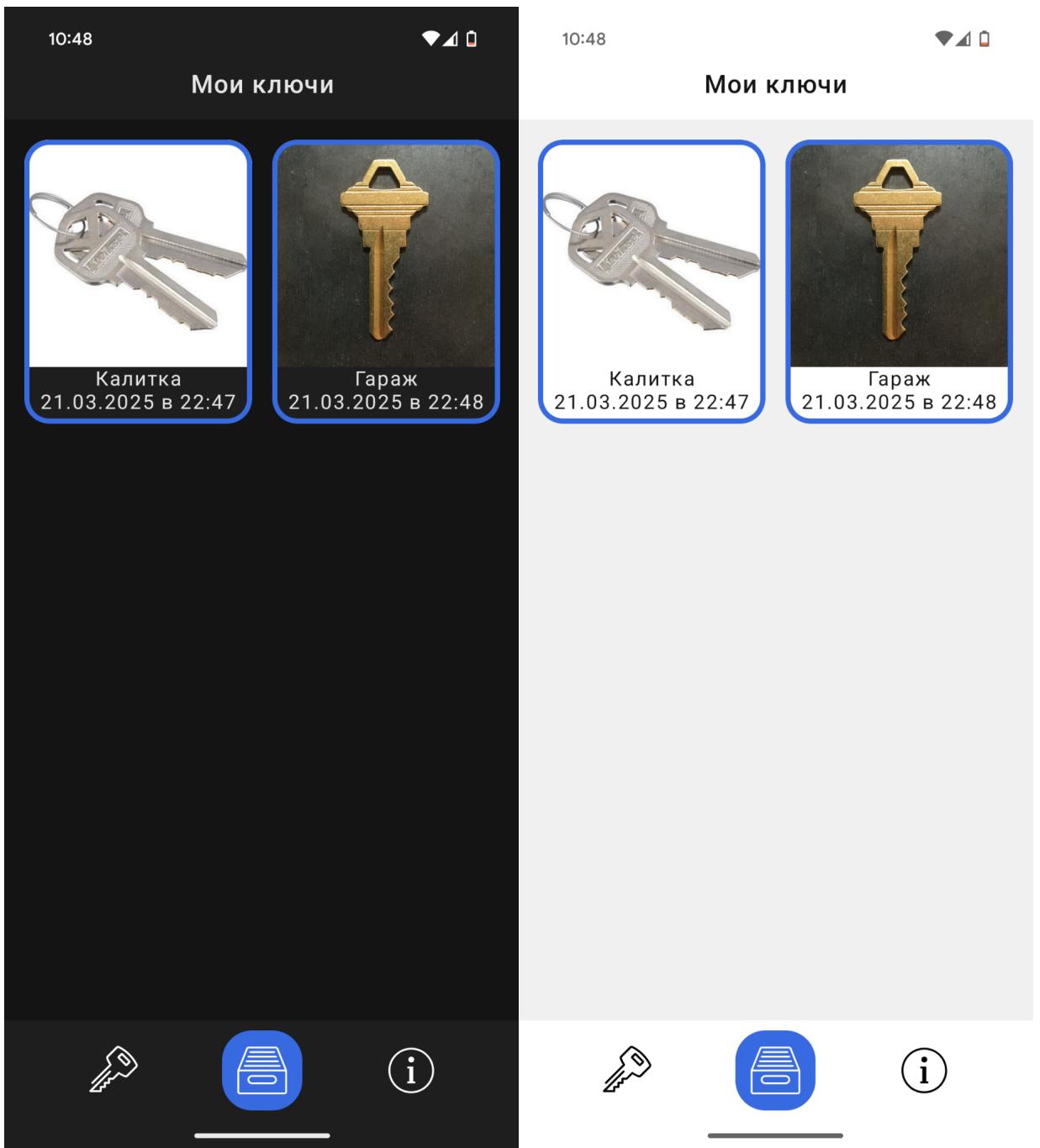


Рисунок 13 – Экран списка созданных копий с фотографиями

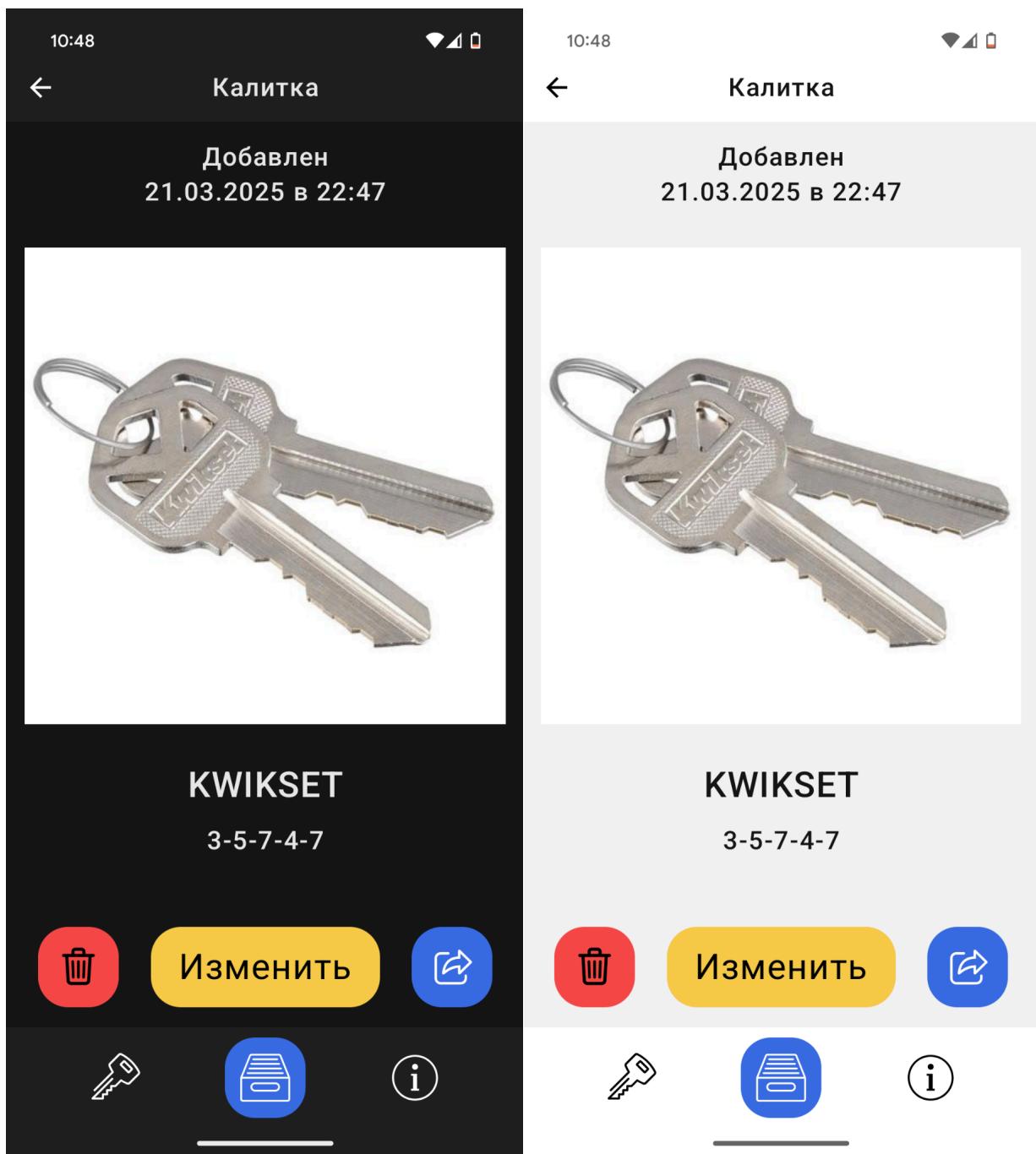


Рисунок 14 – Экран информации о копии ключа

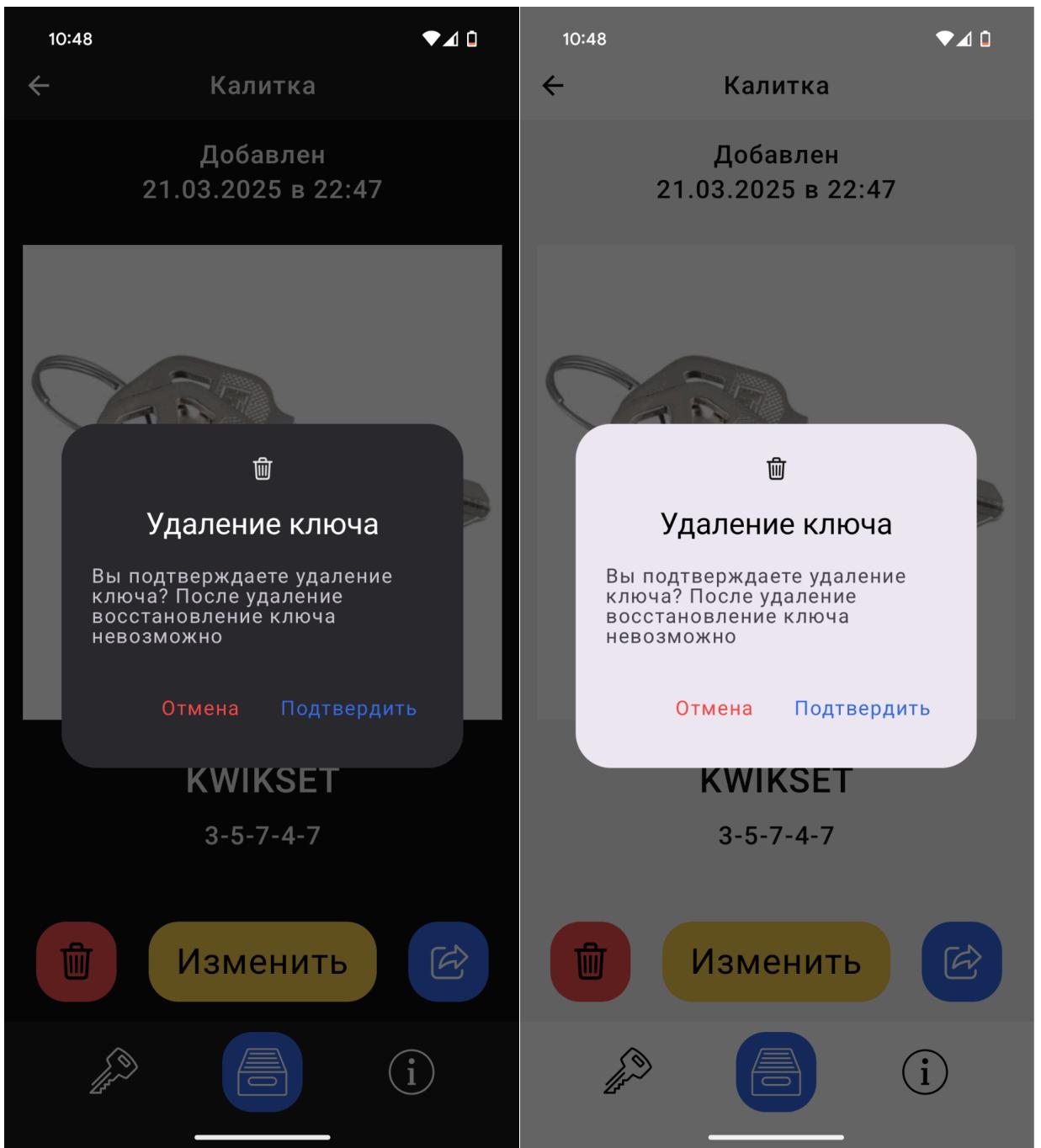


Рисунок 15 – Подтверждение удаления ключа

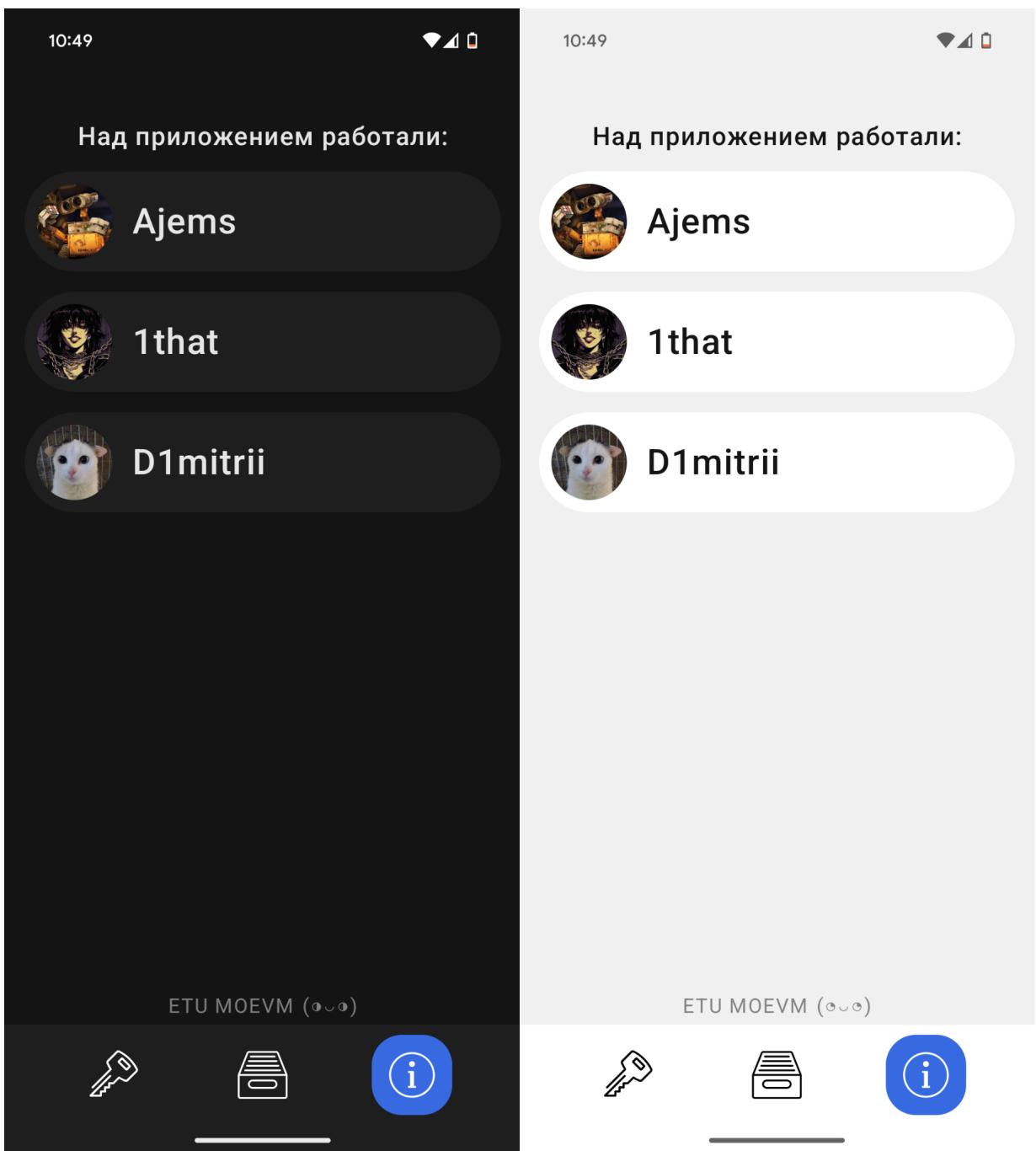


Рисунок 16 – Экран о разработчиках приложения

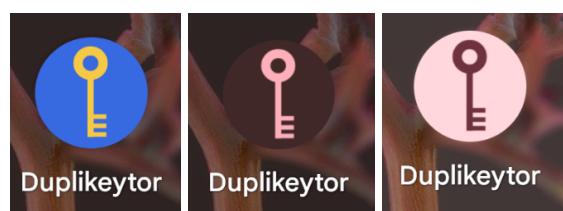


Рисунок 17 – Стандартная и тематические иконки приложения

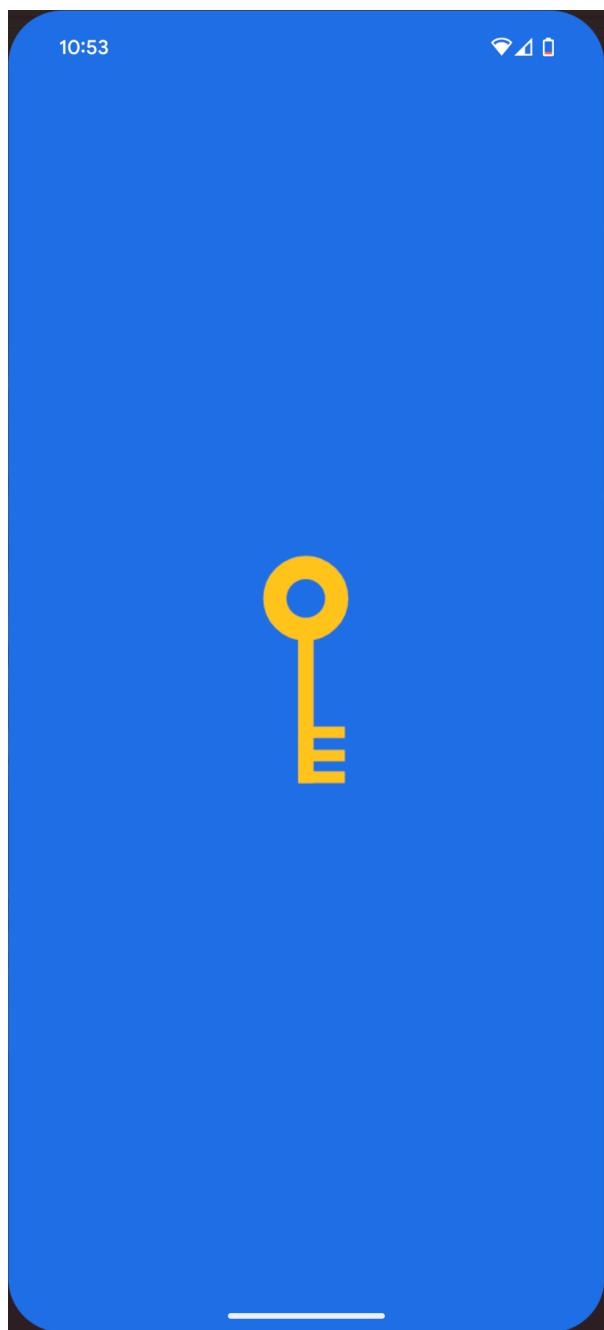


Рисунок 18 – Splash screen приложения