УДК 004

РАЗРАБОТКА ANDROID-ПРИЛОЖЕНИЯ ДЛЯ УПРАВЛЕНИЯ ЗАЯВКАМИ СЕРВИСНОГО ЦЕНТРА

Мифтахова Р.Э., студент гр. 4303, III курс
Научный руководитель: Пыстогов С.В., к.т.н., доцент каф. КС
Казанский национальный исследовательский технический университет
имени А.Н. Туполева – КАИ
г. Казань

В современном цифровом мире, где информационные технологии быстро развиваются, эффективность управления сервисными центрами становится важным аспектом в построении успешного бизнеса. С увеличением числа клиентов, необходимость в автоматизации процессов управления заявками становится более востребованной. Разработка Android-приложения для управления заявками сервисного центра не только упрощает взаимодействие между клиентами и сервисом, но и повышает общую эффективность работы. В данной статье будут рассмотрены основные этапы создания такого приложения, его функциональные возможности и преимущества, которые оно может предоставить как клиентам, так и сотрудникам сервисного центра.

Создание Android-приложения имеет такие преимущества, как:

– Популярность и доступность. Android является самой популярной мобильной операционной системой в мире, охватывающей более 70% рынка. Это значит, что приложение сможет достичь большой аудитории и привлечь больше пользователей.

– Разнообразие устройств. Данная платформа работает на множестве устройств разных производителей, что позволяет пользователям выбрать то, что им больше нравится. Это открывает возможности для большего числа клиентов.

– Гибкость, персонализация и доступность инструментов разработки. Платформа Android предоставляет множество инструментов и ресурсов для разработчиков, включая Android Studio, где разрабатывается данное приложение, что упрощает процесс его создания и тестирования.

Каждое приложение включает в себя фронтенд – часть, с которой взаимодействует пользователь, и бэкенд – внутренняя часть, которая реализует логику приложения [4]. В качестве замены традиционного бэкенда используется Firebase – облачная платформа для разработки мобильных и веб-приложений, предоставляющая готовые сервисы для аутентификации, базы данных, возможность хостинга статических файлов и серверных функций (Cloud Functions), что позволяет обрабатывать серверные задачи без необходимости управлять собственным сервером [3]. Это делает разработку более эффективной и менее затратной. Также Firebase обеспечивает надежную защиту данных

с помощью правил безопасности и шифрования, что очень важно для защиты персональных данных клиентов и сотрудников.

Архитектура мобильного приложения построена на базовых компонентах Android-разработки с соблюдением общепринятых стандартов. В основе лежат элементы Android SDK: Activity – инструмент, создающий окно для компонентов пользовательского интерфейса для управления экранами, а также Fragments – инструмент, отвечающий за разделение интерфейса на части и их использования в разных экранах приложения, система обработки событий [2]. Для работы с заявками реализована интеграция с Firebase, обеспечивающая хранение данных и их синхронизацию.

Кодовая структура приложения спроектирована для эффективного управления функционалом, удобного взаимодействия пользователей (клиентов и сотрудников) и интеграции со сторонними сервисами. Такая организация позволяет поддерживать гибкость, масштабируемость и надежность системы.

Основным этапом разработки выступает внедрение механизмов аутентификации (проверки личности) и авторизации (управления правами доступа) [5]. В разрабатываемом приложении система аутентификации и авторизации пользователей выполнена с помощью Firebase Authentication для обеспечения безопасности и контроля доступа к функциональности. Это инструмент от Google, который предоставляет простой и безопасный способ аутентификации пользователей в мобильных и веб-приложениях. Firebase Authentication поддерживает аутентификацию по электронной почте и паролю.

В системе будут предусмотрены два вида пользователей – это клиент и сотрудник.

Приложение предоставляет клиентам возможность:

– Просматривать историю своих заявок.

– Редактировать личные данные в профиле.

Для сотрудников сервисного центра реализован отдельный интерфейс, позволяющий:

– Управлять всеми поступающими заявками.

– Обновлять статусы ремонта (включая предварительную и итоговую стоимость).

– Отправлять клиентам уведомления об изменениях статуса.

Для проектирования физической модели данных, необходимо структурировать данные с помощью концептуальной модели данных. Концептуальная модель данных – это представление данных и их взаимосвязей визуально, в рамках системы. На рисунке 1 изображена концептуальная модель данных.

Рисунок 1 – Концептуальная модель данных

Данная модель данных позволит определить, какие таблицы и поля нужны, а также как они будут связаны. Это помогает предотвратить избыточность базы данных Firebase Realtime Database.

Firebase Realtime Database – это облачная база данных, поддерживающая синхронизацию данных в реальном времени [3]. Данные хранятся в JSON-формате и обновляются у всех подключенных клиентов при изменении данных автоматически.

На рисунке 2 изображено то, как информация о заявке выглядит в базе данных Firebase Realtime Database.

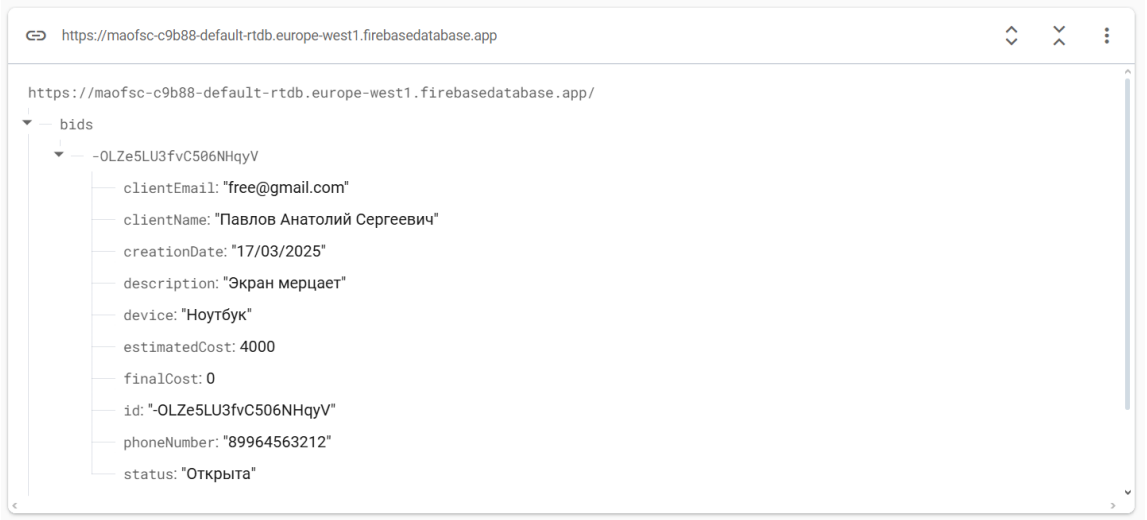

Рисунок 2 – Данные о заявке в базе данных

Следующий этап разработки – реализация интерфейса приложения в Android Studio. На рисунке 3 представлены окно регистрации и окно авторизации.

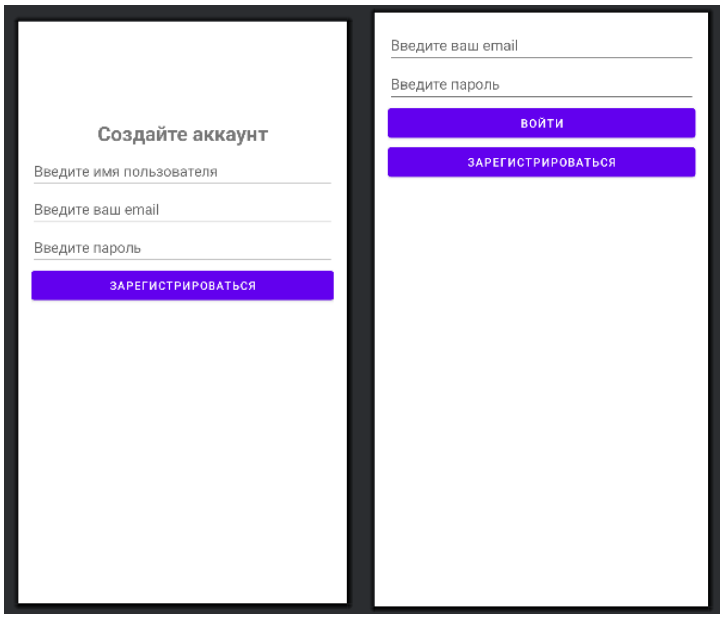

Рисунок 3 – Окно регистрации и авторизации пользователей

На рисунке 4 приведен блок программного кода для окна регистрации в приложении, где в обработчике нажатия кнопки происходит преобразование текста из полей с именем пользователя в строки без пробелов, тоже самое для поля «*email*», «*password*». Далее проверяется, заполнены ли все поля, если да, то вызывается метод *registerUser* с передачей параметров (*email*, *password*, *username*). Если хотя бы одно из полей пустое, то на экране появится всплывающее сообщение с просьбой заполнить все поля.

```
buttonRegister.setOnClickListener {
    val username = editTextUsername.text.toString().trim()
    val email = editTextEmail.text.toString().trim()
    val password = editTextPassword.text.toString().trim()

    if (email.isNotEmpty() && password.isNotEmpty() && username.isNotEmpty()) {
        registerUser(email, password, username)
    } else {
        Toast.makeText( context: this,  text: "Заполните все поля", Toast.LENGTH_SHORT).show()
    }
}
```

Рисунок 4 – Блок кода с обработкой нажатия кнопки «Зарегистрироваться»

На рисунке 5 приведен блок программного кода регистрации в приложении, где метод для регистрации пользователя принимает три параметра: электронную почту, пароль и имя пользователя. Используется метод *Firebase auth.createUserWithEmailAndPassword(email, password)* для того, чтобы добавить нового пользователя с указанной электронной почтой и паролем. Устанавливается слушатель, который будет выполнен после завершения попытки регистрации, а «*task*» содержит результат выполнения [1]. Далее проводится проверка, была ли регистрация успешной, если это так, то происходит получение текущего пользователя. Идет проверка о том, существует ли пользователь, и, если это так, то происходит вызов метода *saveUserDataToDatabase* для сохра-

нения дополнительных данных пользователя в базе данных. В случае успешной регистрации, на экране появится всплывающее сообщение об этом и, в случае ошибки, сообщение об ошибке из исключения.

```
private fun registerUser(email: String, password: String, username: String) {
    auth.createUserWithEmailAndPassword(email, password)
        .addOnCompleteListener(this) { task ->
            if (task.isSuccessful) {
                // Регистрация успешна
                val user = auth.currentUser
                user?.let {
                    // Сохраняем дополнительные данные пользователя в Realtime Database
                    saveUserDataToDatabase(it.uid, username, email, password)
                }
                Toast.makeText( context: this,  text: "Регистрация успешна", Toast.LENGTH_SHORT).show()
            } else {
                Toast.makeText( context: this,  text: "Ошибка: ${task.exception?.message}", Toast.LENGTH_SHORT)
                    .show()
            }
        }
}
```

Рисунок 5 – Блок программного кода метода регистрации

На рисунке 6 приведен блок кода для активности регистрации в приложении, где метод *saveUserDataToDatabase* сохраняет данные о пользователе в Realtime Database, принимает идентификатор пользователя, имя пользователя и его электронную почту.

```
private fun saveUserDataToDatabase(userId: String, username: String, email: String, password: String) {
    val database = Firebase.database( url: "https://maofsc-c9b88-default-rtdb.europe-west1.firebasedatabase.app").reference

    // Создаем объект User
    val user = User(
        username = username,
        email = email,
        phone = "",
        fullName = ""
    )

    // Сохраняем данные в Realtime Database
    database.child( pathString: "users").child(userId).setValue(user)
        .addOnSuccessListener {
            Toast.makeText( context: this,  text: "Данные пользователя сохранены", Toast.LENGTH_SHORT).show()
        }
        .addOnFailureListener { exception ->
            Toast.makeText( context: this,  text: "Ошибка: ${exception.message}", Toast.LENGTH_SHORT).show()
        }
}
```

Рисунок 6 – Блок программного кода для сохранения данных о пользователе

Затем *val database = Firebase.database("https://...").reference* инициализирует ссылку на базу данных Firebase по указанному URL, а *val user = User(...)* создает объект *User* с полями *username*, *email*, *phone* и *fullName*. Поля *phone* и *fullName* оставлены пустыми, но они заполняются пользователем в личном кабинете. Сохраняется объект *User* в базу данных, создавая новую запись под ключом *userId* в разделе *users*. Устанавливается слушатель, который выполняется при успешном сохранении данных. В этом на экране появится всплывающее сообщение о том, что данные пользователя успешно сохранены, и также слушатель, который выполняется в случае ошибки при сохранении данных. Показывается сообщение об ошибке с текстом, полученным из исключения.

После регистрации и авторизации пользователь переходит в окно с заявками и меню, из которого можно перейти во вкладки Личный кабинет, Пользователи/Уведомления (в зависимости от клиента или сотрудника).

На рисунке 7 изображено окно личного кабинета пользователя, где можно изменить данные, нажав на кнопку «Изменить» и сохранить или отменить новые введенные данные.

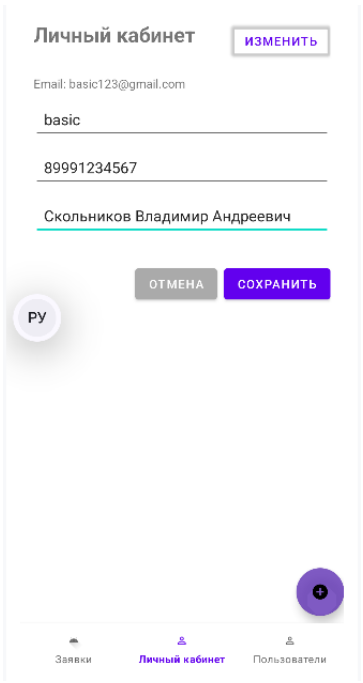

Рисунок 7 – Окно личного кабинета

В данной статье рассмотрены основные этапы разработки Android-приложения для управления заявками сервисного центра. Приложение разработано с целью упрощения взаимодействия между клиентами и сотрудниками сервисного центра, а также повышения эффективности обработки заявок. Основные функции приложения включают регистрацию и авторизацию пользователей, создание и отслеживание заявок, управление статусами и уведомлениями.

Использование Firebase в качестве бэкенда позволило очень сильно упростить процесс разработки, обеспечив надежное хранение данных, синхронизацию в настоящем времени, а также безопасную аутентификацию пользователей. Приложение имеет интуитивно понятный интерфейс и это делает его более удобным как для клиентов, так и для сотрудников сервисного центра.

Разработка такого приложения не только повышает уровень удовлетворенности клиентов, но и оптимизирует рабочие процессы внутри сервисного центра, сокращая время на обработку заявок и улучшая общую эффективность работы. В будущем возможно расширение функциональности приложения, например, добавление интеграции с другими сервисами или внедрение аналитики для отслеживания эффективности работы сервисного центра.

Список литературы:

1. **Жемеров Д., Исакова С.** Kotlin в действии / Жемеров Д., Исакова С. – 2. – Москва: ДМК Пресс, 2018 – 402 с.

2. **Моськин А.** Создание первого проекта в Android Studio / Моськин А. [Электронный ресурс] // habr : [сайт]. – URL: https://habr.com/ru/articles/717394/ (дата обращения: 10.03.2025).

3. **Моськин А.** По следам Google I/O 2016 — новый Firebase: интеграция с Android / Моськин А. [Электронный ресурс] // habr : [сайт]. – URL: https://habr.com/ru/companies/google/articles/305308/ (дата обращения: 14.03.2025).

4. **Пирская, Л. В.** Разработка мобильных приложений в среде Android Studio [Текст] / Л. В. Пирская – 1. – Ростов-на-Дону – Таганрог: Южный федеральный университет, 2019 – 123 с.

5. **Черников В.** Разработка мобильных приложений для iOS и Android / Черников В. – 4. – Москва: ДМК Пресс, 2020 – 188 с.