УДК 004.75
DOI 10.25205/1818-7900-2019-17-4-5-66-73

Разработка бессерверных мобильных приложений

А. Л. Мархакшинов, А. А. Тонхоноева, Е. Р. Урмакшинова

Бурятский государственный университет им. Доржи Банзарова
Улан-Удэ, Россия

Аннотация

Описаны принципы создания мобильных приложений, не требующих написания программного кода серверной части. Базу данных, функции аутентификации пользователей и отправки уведомлений предлагается реализовывать по модели BaaS (Backend as a Service – «бэкенд как услуга»). В качестве примера продемонстрирована архитектура бессерверного мобильного приложения, использующего облачные сервисы Google Firebase.

Ключевые слова

мобильные приложения, бессерверная архитектура, облачные сервисы, разработка приложений

Для цитирования

Мархакшинов А. Л., *Тонхоноева А. А.*, *Урмакшинова Е. Р.* Разработка бессерверных мобильных приложений // Вестник НГУ. Серия: Информационные технологии. 2019. Т. 17, № 4. С. 66–73. DOI 10.25205/1818-7900-2019-17-4-66-73

Serverless Mobile Applications Development

A. L. Markhakshinov, A. A. Tonkhonoeva, E. R. Urmakshinova

Banzarov Buryat State University
Ulan-Ude, Russian Federation

Annotation

Principles of mobile applications development, requiring no server code, are described. Database, user authentication and notification sending functions are suggested to be implemented by BaaS model (Backend as a Service). Example architecture of serverless mobile application using Google Firebase cloud services is shown. Brief description of basic interactions between serverless architecture components is given.

Keywords

mobile applications, serverless architecture, cloud services, application development

For citation

Markhakshinov A. L., Tonkhonoeva A. A., Urmakshinova E. R. Serverless Mobile Applications Development. *Vestnik NSU. Series: Information Technologies*, 2019, vol. 17, no. 4, p. 66–73. (in Russ.) DOI 10.25205/1818-7900-2019-17-4-66-73

Введение

В настоящее время неотъемлемой частью практически любого мобильного приложения стали функции предоставления пользователям индивидуальных учетных записей, хранения и обработки информации с помощью единой базы данных, рассылки Push-уведомлений для оповещения о различных событиях. Как правило, реализация этих функций подразумевает

написание дополнительного программного кода серверной части приложения (в современной разработке ПО часто называемой backend-частью приложения или просто backend'ом), помимо непосредственно мобильного клиента.

Кроме описанных, по сути, базовых серверных функций, разработчики часто стремятся встраивать функции аналитики и сбора статистики для дальнейшего улучшения процесса взаимодействия пользователей с приложением [1].

Очевидно, что сокращение затрат на создание backend-части, или даже полный ее перенос в готовые облачные сервисы позволит существенно сократить сроки выпуска приложения и/или освободить ресурсы для работы над клиентской частью приложения, что особенно актуально для небольших команд разработчиков [2; 3].

К требованиям, предъявляемым к сервисам BaaS, можно отнести [4]:

- независимость от конкретной платформы клиентского приложения;
- масштабируемость предлагаемых решений;
- наличие гибкого тарифного плана.

Наиболее известными поставщиками BaaS-услуг на сегодняшний день являются Google Firebase, Microsoft Azure, AWS Lambda, IBM Cloud Functions [1]. Далее в статье рассматривается бессерверная архитектура приложения, основанная на использовании сервисов Google Firebase. Выбор поставщика обосновывается наличием наиболее полного набора backend-услуг: аутентификация пользователей, облачное файловое хранилище, база данных, облачные функции, аналитика сбоев приложения и его производительности, сервис тестирования приложений, рассылка уведомлений и др. Особый интерес представляет база данных Cloud Firestore, имеющая возможность автоматической синхронизации информации на подключенных клиентских устройствах.

Архитектура бессерверного мобильного приложения

На рис. 1 показана диаграмма возможных взаимодействий в бессерверном мобильном приложении, построенном на базе сервисов Google Firebase. В приложениях, реализованных по одному из наиболее распространенных паттернов MVC, MVP или MVVM, обработка бизнес-логики выносится в отдельный слой, называемый слоем модели. Модель отвечает за различные операции над данными приложения, которые затем некоторым образом, зависящим от выбранного паттерна, визуализируются в пользовательском интерфейсе [5].

В архитектуре, использующей BaaS-подход, взаимодействия с локальной базой данных на устройстве клиента, отмеченные на рис. 1 цифрами *1* и *2*, фактически могут понадобиться лишь для кэширования данных с целью предоставления возможности работы с приложением в период отсутствия подключения к сети Интернет. С учетом того, что сервис Cloud Firestore также поддерживает функцию кэширования часто используемых данных, возможен вариант архитектуры без локальной базы данных.

Множество современных приложений предлагает своим пользователям персонализированный контент, используя для этого систему учетных записей для идентификации и различные методы аутентификации для подтверждения достоверности учетных записей. Сервис Firebase Authentication предлагает готовое решение для создания учетных записей на основе адреса электронной почты или номера телефона. Кроме того, существует возможность использовать учетные записи некоторых других популярных сайтов (Google, Facebook, Twitter, Github).

В процессе аутентификации (*3*) введенные пользователем логин и пароль пересылаются по защищенному соединению в облачный сервис Firebase, где выполняется их сверка с базой существующих учетных записей приложения. Результат запроса на аутентификацию (*4*) воз-

[1] Comparing Serverless Architecture Providers: AWS, Azure, Google, IBM, and Other FaaS Vendors. URL: https://dzone.com/articles/comparing-serverless-architecture-providers-aws-az

вращается в клиент-приложения. При положительном результате становится доступен уникальный идентификационный номер пользователя и данные его учетной записи. Эти сведения затем могут использоваться для обращений к базе данных и получению информации, актуальной для конкретного пользователя.

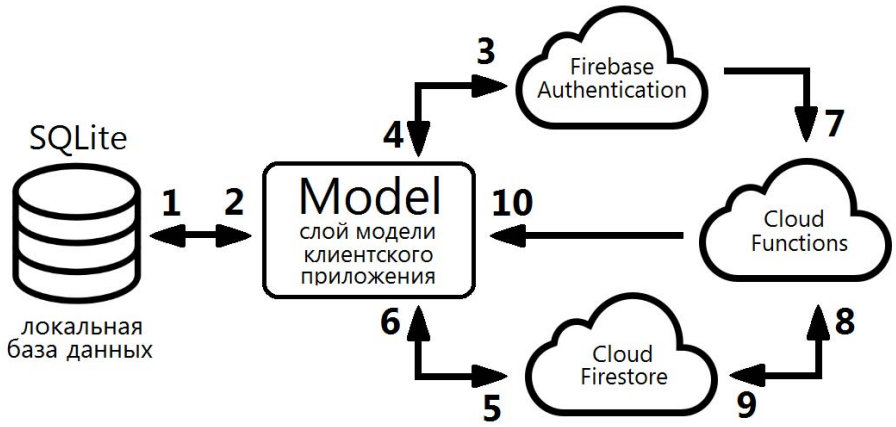

Рис. 1. Пример архитектуры бессерверного мобильного приложения
Fig. 1. Example of serverless mobile application architecture

В качестве базы данных в предложенной архитектуре используется сервис Cloud Firestore, представляющий собой облачную NoSQL-базу данных. В Firestore данные имеют гибкую структуру и хранятся в виде документов, которые индексируются по умолчанию и организовываются в коллекции и субколлекции. Преимуществами Firestore, по заявлению разработчика, являются система запросов с удобной фильтрацией и сортировкой результатов, синхронизация с клиентскими устройствами в реальном времени, поддержка офлайн-режима работы с базой данных и легкая масштабируемость.

Обращения к Firestore (*5*) делятся на два типа: однократный запрос данных и подписка на события изменения данных в определенном узле базы. Результатом обращения (*6*) в обоих случаях может являться как отдельный документ, так и группа документов, отобранная и отсортированная в соответствии с параметрами запроса. При однократном запросе существует возможность выбрать источник данных: только облачная база данных, только локальный кэш или облачная база с возвратом результата из кэша в случае недоступности базы данных. При подписке на изменения в определенном документе, коллекции или группе коллекций сервис Firestore в автоматическом режиме отслеживает события изменения, удаления или создания данных, присылая на клиент обновленную информацию.

Для работы с сервисами Firebase Authentication и Cloud Firestore доступен API для наиболее популярных языков программирования (Java, Swift, Objective-C, Kotlin, PHP, Python, Go, C#, Ruby), а также REST API.

Сервис Cloud Functions позволяет автоматически запускать серверный код по триггеру или по HTTPS-запросу. Код функций пишется на языке JavaScript и исполняется на облачных серверах Google, избавляя разработчиков от необходимости запускать и поддерживать собственные серверы.

Триггерами для Cloud Functions являются различные события сервисов Firebase. Например, для Firebase Authentication триггерами (*7*) являются события создания и удаления учетных записей. Для Cloud Firestore – события (*8*) создания, изменения и удаления документов в базе данных. Функции, вызываемые из Cloud Functions, могут использоваться для обслужи-

вания базы данных Firestore (*9*), либо для взаимодействия с клиентскими приложениями пользователей (*10*).

Бессерверная архитектура на примере Android-приложения

Коротко продемонстрируем практическое использование сервисов Google Firebase на примере простого приложения, позволяющего пользователям обмениваться сообщениями в реальном времени в тематических каналах.

Для того чтобы начать пользоваться сервисами Firebase в своем проекте, необходимо пройти процедуру регистрации в консоли Firebase[2]. После этого можно добавлять в приложение требующиеся сервисы. Начнем с аутентификации пользователей с помощью Firebase Authentication. В зависимости Android-проекта следует добавить соответствующую библиотеку:

```
implementation 'com.google.firebase:firebase-auth:17.0.0'
```

Доступ к услугам авторизации пользователей в программном коде предоставляется через экземпляр объекта `FirebaseAuth`:

```
private FirebaseAuth mAuth;
```

Данный экземпляр необходимо инициализировать в методе `onCreate()` стартовой активности приложения:

```
mAuth = FirebaseAuth.getInstance();
```

В методе `onStart()` можно получить сведения о текущем пользователе:

```
FirebaseUser currentUser = mAuth.getCurrentUser();
```

Если объект `currentUser` равен `null`, то пользователь не авторизовался в приложении. В противном случае данный объект может предоставить такие сведения о пользователе, как имя, электронный адрес, ссылку на фото профиля, уникальный идентификатор.

Вызывая у объекта `mAuth` методы `createUserWithEmailAndPassword` и `signInWithEmailAndPassword`, можно выполнять операции создания новой учетной записи и входа в приложение.

Сервис облачной базы данных Cloud Firestore добавляется в проект путем включения следующей зависимости:

```
implementation 'com.google.firebase:firebase-firestore:19.0.1'
```

Как упоминалось выше, Cloud Firestore является NoSQL-базой, в которой данные не имеют жестко заданной структуры, хранятся в виде документов и организовываются в коллекции или субколлекции. Информация в документах представляется согласно схеме JSON, т.е. в виде наборов пар «ключ-значение».

В качестве примера на рис. 2 изображена структура базы данных Firestore для приложения, позволяющего пользователям обмениваться сообщениями в отдельных групповых каналах.

[2] Firebase console. URL: https:// console.firebase.google.com

Документ является базовой единицей хранения информации в Firestore, и содержит в себе набор ключей и значений, причем структура набора может быть произвольной. Название документа должно быть уникальным в пределах его коллекции, а ограничение на максимальный объем документа – 1 МБ. Коллекции фактически представляют собой контейнеры для группировки документов. Иерархия данных в базе Firestore обеспечивается за счет привязки субколлекций к определенному документу.

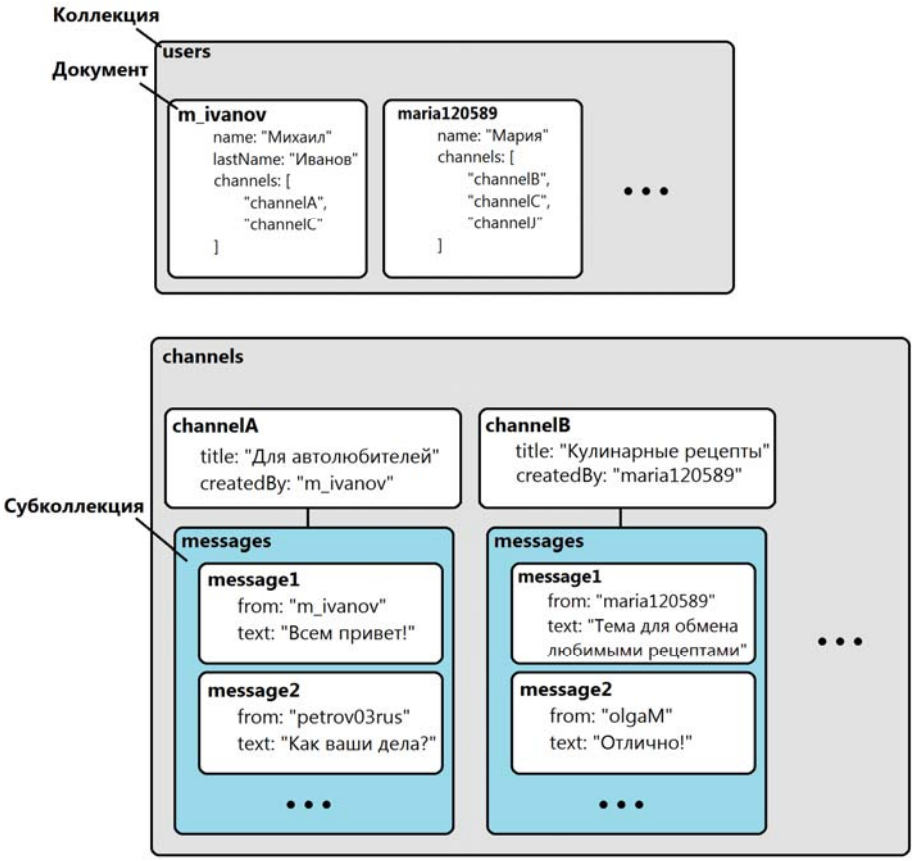

Рис. 2. Хранение данных в базе Cloud Firestore
Fig. 2. Data storing in Cloud Firestore database

В программном коде работа с базой данных осуществляется путем обращения к экземпляру класса `FirebaseFirestore`:

```
FirebaseFirestore db = FirebaseFirestore.getInstance();
```

Добавление документа происходит с помощью создания структуры Map с соответствующими полями и ее передачи в метод `add()`:

```
Map<String, Object> user = new HashMap<>();
user.put("name", "Михаил");
user.put("lastName", "Иванов");
db.collection("users").add(user);
```

Запрос документа или документов выполняется с помощью метода `get()`, которому могут предшествовать методы отбора результатов:

```
//Запрос всех сообщений пользователя с идентификатором m_ivanov
//в канале с идентификатором channelA
db.collection("channels").document("channelA")
.collection("messages").whereEqualTo("from", "m_ivanov")
.get();
```

Для получения обновлений в реальном времени, например, для автоматического отображения новых сообщений в интерфейсе пользователя, Firestore позволяет подписаться на события коллекций, субколлекций или документов. Для осуществления подписки необходимо создать ссылку на нужный узел базы данных и реализовать слушатель изменений, который будет обрабатывать поступающие данные, а также потенциальные ошибки:

```
final CollectionReference chRef = db.collection("channels")
                .document("channelA").collection("messages");
chRef.addSnapshotListener(new EventListener<DocumentSnapshot>() {
    @Override
    public void onEvent(@Nullable DocumentSnapshot snapshot,
                        @Nullable FirebaseFirestoreException e) {
        if (e != null) {
            //Обработать ошибку
            return;
        }
        if (snapshot != null && snapshot.exists()) {
            //Извлечь и отобразить новые сообщения
        } else {
            //Новых сообщений не было;
        }
    }
});
```

Сервис Cloud Functions предназначен для выполнения программного кода на серверной стороне и, в отличие от Firebase Authentication и Cloud Firestore, взаимодействие с которыми происходит в коде клиентского приложения, предполагает написание отдельных функций на языке JavaScript. После предварительной настройки окружения на компьютере разработчика, включающей в себя установку платформы Node.js и плагина Firebase CLI для взаимодействия с Firebase через командную строку, можно создавать JavaScript-функции, которые будут запускаться в облаке по HTTPS-запросу либо в ответ на события сервисов Firebase.

Ниже приведен пример функции makeUpperCaseMessage, которая автоматически исполняется каждый раз при поступлении нового сообщения в базу данных Firestore и изменяет регистр букв в тексте сообщения на прописной:

```
exports.makeUpperCaseMessage = functions.firestore
        .document('channels/{channelId}/messages/{messageId}')
        .onCreate((snap, context) => {
    //Получить объект документа, т.е. сообщение
    const newMessage = snap.data();
    //Получить текст сообщения и изменить регистр букв
    const text = newMessage.text;
    const upperCaseText = text.toUpperCase();
```

```
    //Применить изменения
    return snap.ref.set({text: upperCaseText}, {merge: true});
});
```

Заключение

Рассмотренный подход к построению серверной части мобильных приложений все чаще применяется на практике. Значительное снижение затрат на разработку ускоряет создание как прототипов приложений, так и полноценных продуктов, готовых к коммерческому запуску. Стоит отметить, что нужды приложений с небольшой пользовательской базой и относительно редкими сетевыми взаимодействиями могут быть удовлетворены даже полностью бесплатными тарифами описанных облачных сервисов [6].

Построенная по принципу BaaS серверная часть может использоваться без каких-либо изменений не только в мобильной разработке, но и в настольных и веб-приложениях [7; 8], являясь полноценным «бэкендом», не зависящим от конкретного типа клиентских платформ.

Список литературы / References

1. **Wadkar M., Patil P.** Traditional Infrastructure vs. Firebase Infrastructure. *International Journal of Trend in Scientific Research and Development*, 2018, vol. 2, no. 4, p. 2050–2053. DOI 10.31142/ijtsrd14550
2. **Vemula R.** A New Era of Serverless Computing. In: Integrating Serverless Architecture. Apress, Berkeley, CA, 2019, p. 1–22. DOI 10.1007/978-1-4842-4489-0_1
3. **Sharma D., Dand H.** Firebase as BaaS for College Android Application. *International Journal of Computer Applications*, 2019, vol. 178, no. 20, p. 1–6. DOI 10.5120/ijca2019918977
4. **Stigler M.** Understanding Serverless Computing. In: Beginning Serverless Computing. Apress, Berkeley, CA, 2018, p. 1–14. DOI 10.1007/978-1-4842-3084-8_1
5. **Khawas C., Shah P.** Application of Firebase in Android App Development-A Study. *International Journal of Computer Applications*, 2018, vol. 179, no. 46, p. 49–53. DOI 10.5120/ijca2018917200.
6. **Albertengo G., Debele F., Hassan W., Stramandino D.** On the performance of Web Services, Google Cloud Messaging and Firebase Cloud Messaging. In: Digital Communications and Networks, 2019. DOI 10.1016/j.dcan.2019.02.002.
7. **Hajian M.** Deploying to Firebase as the Back End. In: Progressive Web Apps with Angular. Apress, Berkeley, CA, 2019, p. 9–27. DOI 10.1007/978-1-4842-4448-7_2
8. **Moroney L.** Using Firebase Hosting. In: The Definitive Guide to Firebase. Apress, Berkeley, CA, 2017, p. 93–106. DOI 10.1007/978-1-4842-2943-9_5

Материал поступил в редколлегию
Received
22.06.2019

Сведения об авторах

Мархакшинов Аюр Лувсаншаравович, кандидат технических наук, старший преподаватель кафедры ВТ и информатики, Бурятский государственный университет (Улан-Удэ, Россия)
ayurmar@yandex.ru

Тонхоноева Антонида Антоновна, кандидат педагогических наук, доцент кафедры ВТ и информатики, Бурятский государственный университет (Улан-Удэ, Россия)
ant_ton@mail.ru

Урмакшинова Елена Ронославовна, кандидат технических наук, доцент, заведующая кафедрой ВТ и информатики, Бурятский государственный университет (Улан-Удэ, Россия)
helurm@mail.ru

Information about the Authors

Antonida A. Tonkhonoeva, Cand. Sci. (Education), A/Prof., Department of Computer Engineering and Computer Science, Buryat State University (Ulan-Ude, Russia)
ant_ton@mail.ru

Ayur L. Markhakshinov, Cand. Sci. (Engineering), Senior Lecturer, Department of Computer Engineering and Computer Science, Buryat State University (Ulan-Ude, Russia)
ayurmar@yandex.ru

Elena R. Urmakshinova, Cand. Sci. (Engineering), A/Prof., Head of Department, Department of Computer Engineering and Computer Science, Buryat State University (Ulan-Ude, Russia)
helurm@mail.ru