РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ ДЛЯ ОТСЛЕЖИВАНИЯ ПОГОДЫ

Бондарев К.Н.[1], Саврасов Ф.В.[2]
[1]Томский политехнический университет, студент гр. 8ПМ22, email: knb6@tpu.ru
[2]Томский политехнический университет, ОИТ ИШИТР, доцент, email: savrasov@tpu.ru

Аннотация

В данной статье рассматривается разработка мобильного приложения для слежения за прогнозом погоды. Описывается архитектура приложения, включая принцип его работы и функциональные возможности. Описывается взаимодействие приложения с серверной частью, выделяется роль серверной части в системе. В заключении приводятся сведения о технологиях, использованных авторами для реализации системы.

Ключевые слова: мобильное приложение, агрегация метеоданных, вариативность источников.

Введение

В настоящее время существует востребованный и популярный класс программных систем, отображающих прогноз погоды с различной детальностью и в различные периоды времени. Одними из самых популярных приложений в России являются Яндекс.Погода, Гисметео, Weather.com, Windy.com, AccuWeather [1, 2], но у данных систем имеются некоторые недостатки. Основная проблема заключается в том, что эти системы используют данные из своих источников, причем источники у них разные [2, 3, 4]. Как следствие, прогнозы погоды часто не совпадают как между источниками, так и с действительностью. Данное утверждение можно проверить, сравнив прогнозы погоды между разными сервисами. На рисунке 1 представлено расхождение десятидневного прогноза погоды между сервисами Яндекс.Погода и Гисметео соответственно.

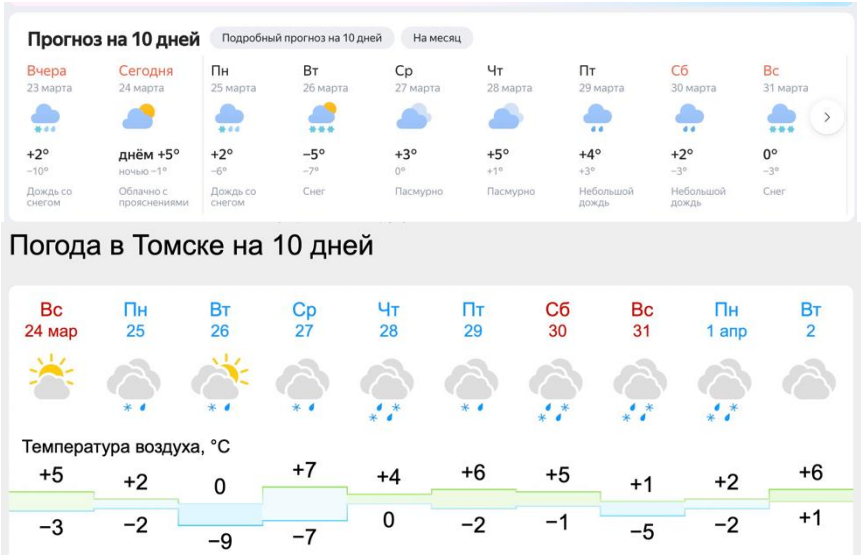

Рис. 1. Расхождение десятидневного прогноза погоды

Пользователи, которым известна данная проблема, вынуждены самостоятельно загружать прогнозы погоды из нескольких источников и сравнивать их вручную.

Мобильное приложение

Решением проблемы было бы мобильное приложение, которое показывало бы данные о погоде из нескольких источников в удобном для сопоставления виде.

Информация, получаемая из нескольких источников, может быть более точной, так как каждый источник использует свои собственные методы сбора данных и алгоритмы прогнозирования погоды. Кроме того, приложение может предоставлять пользователю более подробную информацию о погоде,

например, о скорости ветра, влажности воздуха, атмосферном давлении и других параметрах, что позволит более точно спланировать свой день.

В результате точного выявления необходимой функциональности приложения были составлены макеты графического интерфейса. Благодаря макетам были составлены диаграммы сценариев использования и анализа. Макеты представлены на рисунке 2.

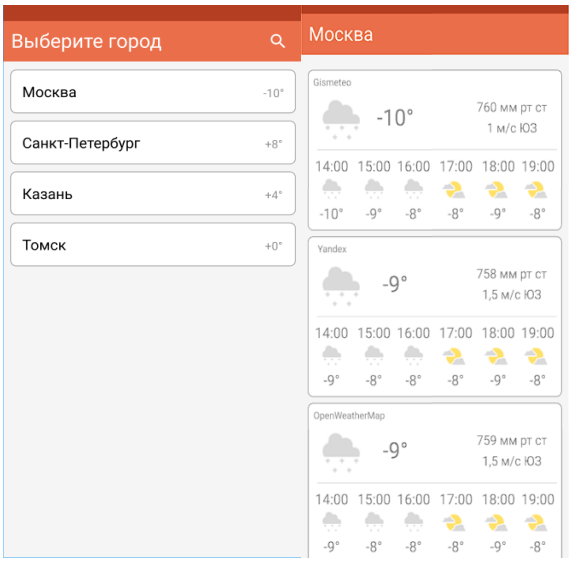

Рис. 2. Графический интерфейс приложения

Архитектура системы

Предполагается, что мобильное приложение будет интенсивно работать с источниками данных, количество которых может варьироваться. Следовательно, приложение подобного рода не может обойтись без серверной части, причины для этого – следующие:

- для добавления нового источника метеоданных не придется выпускать новую версию мобильного приложения;
- в мобильном приложении не будет логики работы с несколькими источниками данных (агрегация, сравнение, и так далее);
- мобильному приложению не придется опрашивать каждый из источников на предмет метеоданных;
- источники в свою очередь могут иметь ограничения по количеству обращений.

Самой главной причиной из вышеперечисленных является наличие логики работы с несколькими источниками данных, поскольку каждый из источников может иметь свой формат представления данных. Наличие серверной части позволяет решить данную проблему. Таким образом, мобильному приложению достаточно будет общаться только с серверной частью.

Мобильное приложение и серверная часть взаимодействуют с помощью API, основывающегося на технологии GraphQL [6], которая предоставляет данные о населенных пунктах и метеоданные мобильному приложению. Серверная часть, в свою очередь, получает метеоданные из внешних систем. Способ получения информации в этом случае заключается в периодическом опросе источников. Используя их открытые интерфейсы, происходит аккумулирование различного рода метеорологической информации из различных районов сбора данных в пределах одного погодного сервиса, далее эта информация анализируется на стороне сервиса.

В составе системы также необходима база данных для кэширования информации, полученных от внешних сервисов, это позволит сэкономить ресурсы серверной части и снизить количество обращений к внешним сервисам. Взаимодействие серверной части и внешних сервисов происходит с помощью REST API, так как это наиболее популярный формат общения сервисов в настоящее время. На рисунке 3 представлена архитектура системы в виде схемы.

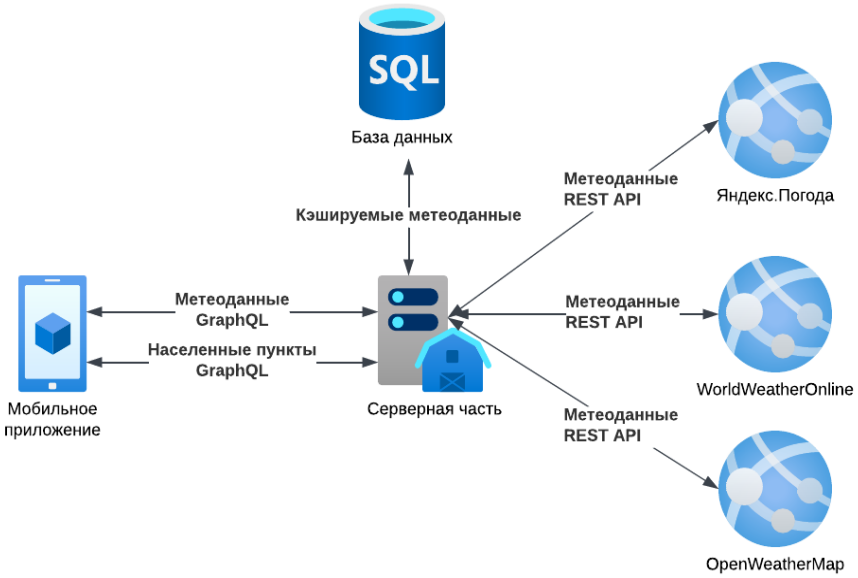

Рис. 3. Архитектура системы

Технологический стек

Мобильные приложения становятся все более сложными, требуя эффективных и поддерживаемых решений для разработки. Технологический стек, используемый при разработке данной системы, предоставляет набор взаимодополняющих инструментов и библиотек, которые помогают разработчикам создавать высококачественные мобильные приложения.

Мобильное приложение разрабатывается с использованием следующих технологий и подходов:

- Jetpack Compose: позволяет создавать пользовательские интерфейсы с помощью декларативного подхода, описывая пользовательский интерфейс в виде кода на языке программирования Kotlin.
- Clean Architecture: набор подходов, разделяющий приложение на отдельные слои (домен, данные, представление), что делает код более управляемым, тестируемым и поддерживаемым [7].
- Room: предоставляет объектно-ориентированный API для работы с базами данных SQLite, упрощая создание и выполнение запросов к базе данных, а также управление схемами и миграциями.
- Kotlin Coroutines: позволяет разработчикам писать асинхронный код с использованием синтаксиса, похожего на синхронный код, что упрощает обработку асинхронных операций и параллелизм.

Серверная часть, в свою очередь, использует следующие технологии и подходы:

- «Луковая архитектура»: шаблон проектирования программного обеспечения, который организует компоненты приложения в концентрические слои, подобно слоям луковицы. Каждый слой зависит только от внутренних слоев и предоставляет абстракцию для внешних слоев [7].
- Koin: внедряет зависимости в классы приложения, упрощая управление зависимостями и тестирование.
- Ktor: обеспечивает сетевое взаимодействие, обрабатывает HTTP-запросы и ответы, поддерживает различные форматы данных.
- KGraphQL: позволяет серверу реализовать GraphQL-интерфейс, который предоставляет клиентам гибкий способ запроса данных.
- Exposed: предоставляет объектно-ориентированный API для работы с базами данных, упрощая создание и выполнение запросов к базе данных.
- PostgreSQL: мощная и надежная система управления базами данных с открытым исходным кодом, используемая для хранения прогнозов погоды [8].
- Kotlin Serialization: сериализует и десериализует объекты Kotlin в различные форматы данных, такие как JSON, для передачи данных между клиентом и сервером.

Заключение

В результате выполнения работы были выявлены требования к приложению, выбран технологический стек, осуществлено проектирование мобильного приложения, представлены макеты графического интерфейса, проведено проектирование серверной части приложения, выполнен анализ источников метеоданных.

Список использованных источников

1. Популярные мобильные приложения в категории – Погода // Электронный ресурс. – URL: https://app.sensortower.com/top-charts?category=weather&country=RU&date=2024-02-10&device=iphone&os=android (дата обращения 10.02.2024)

2. Магазин приложений Apple AppStore категория – Погода // Электронный ресурс. – URL: https://apps.apple.com/ru/charts/iphone/%D0%BF%D0%BE%D0%B3%D0%BE%D0%B4%D0%B0-apps/6001 (дата обращения 10.02.2024)

3. Гисметео – Условия использования // Электронный ресурс. – URL: https://www.gismeteo.ru/informers/offer/ (дата обращения 24.03.2024)

4. О сервисе Windy.com // Электронный ресурс. – URL: https://community.windy.com/topic/4/about-windy (дата обращения 24.03.2024)

5. Как Яндекс прогнозирует погоду // Электронный ресурс. – URL: https://yandex.ru/company/technologies/meteum (дата обращения 24.03.2024)

6. Документация GraphQL // Электронный ресурс. – URL: https://graphql.org/learn/ (дата обращения 15.02.2024)

7. Эрик Эванс Предметно-ориентированное проектирование. Структуризация сложных программных систем. // 1 изд. – Диалектика. – 2018. – 443 с.

8. PostgreSQL – Документация // Электронный ресурс. – URL: https://www.postgresql.org/docs/ (дата обращения 17.02.2024)