Лабораторная работа №4

Разработка клиент-серверного приложения

Цель работы

Освоить технологии взаимодействия мобильного приложения с API сервера приложения и/или внешним API.

Теоретический материал

API (Application Programming Interface, программный интерфейс приложения) является жизненно важным компонентом в современном ландшафте разработки программного обеспечения, обеспечивая строительные блоки для взаимодействия приложений друг с другом. В этой статье рассмотрим пять основных типов API: REST, SOAP, WebSocket, gRPC и GraphQL, чтобы получить более четкое представление об их функциях, особенностях и идеальных сценариях использования.

REST API (Representational State Transfer)

REST API является наиболее распространенным типом API в клиент-серверной архитектуре и имеет ряд свойств — например, отсутствие хранения состояния и кэшируемость. REST API построен на стандартных протоколах HTTP и обычно возвращает данные в формате JSON, хотя может поддерживать и другие форматы.

REST API использует стандартные HTTP-методы для взаимодействия, включая GET, POST, PUT, DELETE и другие. Каждый метод соответствует определенному типу действий, которые могут быть выполнены над ресурсами API.

Простота и гибкость REST API сделали его лучшим выбором для многих разработчиков, но при работе с большими объемами данных он может быть неэффективными, так как для получения всех необходимых данных может требоваться несколько запросов.

SOAP API (Simple Object Access Protocol)

SOAP — это протокол для обмена структурированной информацией в веб-сервисах с использованием XML. Он обладает высокой расширяемостью и

позволяет осуществлять обмен данными через несколько различных транспортных протоколов, включая HTTP, SMTP и другие.

API SOAP известны своей надежностью и часто используются в энтерпрайз-разработке. Они предлагают встроенную обработку ошибок и их можно использовать с различными сетевыми протоколами. Однако зависимость SOAP от XML может приводить к большим объемам передаваемых данных, что делает его менее эффективным по сравнению с другими API в каких-то сценариях использования.

WebSocket API

WebSocket API обеспечивает постоянный, полнодуплексный канал связи между клиентом и сервером. В отличие от REST и SOAP, которые придерживаются формата «запрос-ответ», WebSocket сохраняет соединение открытым, что позволяет передавать данные в режиме реального времени. Это делает WebSocket API идеальным для приложений, требующих функциональности в реальном времени, таких как чат-приложения, онлайн-игры и системы отслеживания в реальном времени.

Несмотря на свою мощь, WebSocket API могут быть более сложными в реализации и требуют больше ресурсов для поддержания открытых соединений.

gRPC API (Google Remote Procedure Call)

gRPC — это высокопроизводительная платформа с открытым исходным кодом, разработанная компанией Google. Он использует протокол HTTP/2 для передачи данных и Protocol Buffers (protobuf), высокопроизводительный формат двоичных данных, в качестве языка определения интерфейса.

gRPC поддерживает четыре типа взаимодействия: унарный (стандартный запрос-ответ), потоковая передача данных с сервера, потоковая передача данных от клиента и двунаправленная потоковая передача. Эти возможности в сочетании с эффективностью делают gRPC идеальным решением для микросервисной архитектуры.

GraphQL API

Разработанный компанией Facebook, GraphQL представляет собой язык запросов к API и рабочую среду для выполнения этих запросов. В отличие от REST API, где вам нужно делать запросы к различным эндпоинтам, чтобы получить соответствующие данные, GraphQL позволяет сделать один единственный запрос, чтобы получить именно те данные, которые вам нужны.

В GraphQL пользователи определяют форму и размер ответа, что приводит к более эффективной загрузке данных и уменьшению избыточной выборки данных. Он отлично подходит для сложных систем и микросервисов, где данные распределены между различными сервисами.

Краткое заключение

Каждый из этих API имеет свои сильные стороны и сценарии использования, понимание которых может помочь в принятии обоснованных проектных решений. REST API являются популярным выбором благодаря своей простоте и широкому распространению. SOAP API хорошо работают в энтерпрайз-средах благодаря своим функциям безопасности. WebSocket API отлично подходят для приложений реального времени, gRPC API — для высокопроизводительных микросервисов, а API GraphQL обеспечивают гибкий поиск данных, что отлично подходит для сложных систем.

Сравнительная таблица

	REST API	SOAP API	WebSocket API	gRPC API	GraphQL API
Формат данных	Как правило, JSON, но может поддерживать несколько форматов	XML	Обычно JSON, но может быть любой тип данных	Protocol Buffers (бинарный формат)	JSON
Протокол	HTTP/HTTPS	HTTP/HTTPS, SMTP, XMPP, и другие	WS/WSS (WebSocket secure)	HTTP/2	HTTP/HTTPS
Тип связи	Односторонняя связь (запрос/ответ)	Односторонняя связь (запрос/ответ)	Двусторонняя связь (полный дуплекс)	Двунаправленная потоковая передача	Односторонняя связь (запрос/ответ)
Service Discovery	Не стандартизированы, часто используют OpenAPI (Swagger) для документации	Стандартизация с помощью WSDL	Не стандартизировано	Не стандартизирован, но поддерживает Server Reflection	Не стандартизировано, интроспекция для обнаружения схем

	REST API	SOAP API	WebSocket API	gRPC API	GraphQL API
Производительность	Варьируется, обычно ниже, чем бинарные протоколы из-за текстовой природы	Снижена из-за словесного формата XML	Выше благодаря полнодуплексной связи и отсутствию накладных расходов HTTP	Высокий (бинарный протокол, преимущества HTTP/2)	Варьируется, эффективная загрузка данных может повысить производительность
Сценарий использования	Общее назначение, широкое применение	Распределенные энтерпрайз-среды, унаследованные системы	Приложения в режиме реального времени, когда серверу необходимо пушить обновления	Микросервисы, системы, критичные к производительности	Когда требуется гибкая и эффективная загрузка данных
Сложность	От низкого до умеренного	Высокая	Умеренная	От умеренной до высокой	Умеренная

Пример спецификации API

Таблица 1 Спецификация получения содержимого сессии активности

ID	Наименование бизнес-операции	Описание
1	Получить содержимое сессии активности	Возвращает содержимое одной сессии
ОПИСАНИЕ ЗАПРОСА		
Название метода	Тип запроса	Endpoint
Sessions	GET	https://www.example.com/fitness/v1/users/me/session
ПРИМЕР		
https://www.example.com/fitness/v1/users/me/session?id=someSessionId		
ПАРАМЕТРЫ ЗАПРОСА		

№	Метка параметра	Обязат.	Формат параметра	Пример
1.	id	0..1	идентификатор сессии, за которую необходимо получить содержимое	id=someSessionId
ТЕЛО ЗАПРОСА				
–				
ОПИСАНИЕ ОТВЕТА				
№	Метка атрибута	Обязат.	Формат атрибута	Пример
1.	statusCode	1..1	200 OK в случае успешного запроса	200 OK
2.	details	0..n	список активностей	
2.1	id	1..1	идентификатор активности	«id»: «someSessionId»

Задание на работу

1. Выбрать и описать предметную область мобильного приложения (можно по теме курсового проекта)
2. Спроектировать архитектуру системы из мобильного приложения и API собственного сервера / внешнего API. Отобразить на диаграмме развёртывания
3. Представить спецификацию API сервера / внешнего API. Обязательно должны быть представлены:
    - простые запросы (не требующие передачи параметров / с небольшим количеством параметров)
    - сложные (включают тело запроса со сложным объектом)
    - get-запросы
    - post-запросы
    - минимум 4 запроса
4. Представить структуру файлов и код в мобильном приложении, которые отвечают за взаимодействие по API
5. Сделать вывод

Отчёт по работе

1. Описание предметной области (примерно 1/2 страницы)
2. Диаграмма развёртывания
3. Спецификация API (минимум 4 таблицы)
4. Реализация мобильного приложения с кодом и скриншотами
5. Вывод