Задание 1. Трехзвенная (клиент-серверная) архитектура

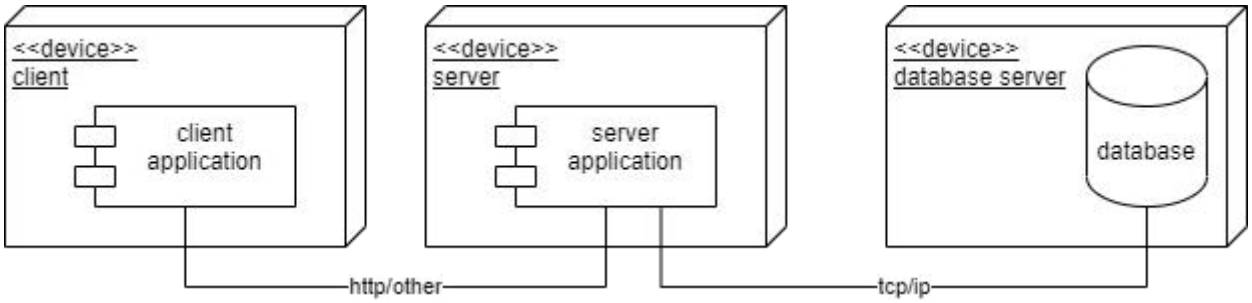

Рис. 1 Обобщенная диаграмма развертывания приложения на основе трехзвенной архитектуры

Основная идея трехуровневой архитектуры заключается в разделении информационной системы на три компоненты: клиент, сервер, сервер базы данных. В простейшей конфигурации физически сервер приложений может быть совмещён с сервером базы данных на одном компьютере, к которому по сети подключается один или несколько терминалов. В «правильной» (с точки зрения безопасности, надёжности, масштабирования) конфигурации сервер базы данных находится на выделенном компьютере (или кластере), к которому по сети подключены один или несколько серверов приложений, к которым, в свою очередь, по сети подключаются терминалы.

Можно выделить основные преимущества использования трехзвенной архитектуры:

1. масштабируемость - один сервер может одновременно обслуживать множество клиентов, так же, как и один сервер баз данных может одновременно обслуживать множество серверов приложений;
2. устойчивость - изоляция уровней друг от друга позволяет быстро заменить вышедший из строя узел;
3. высокая безопасность - за счет отсутствия прямого доступа клиентов к базе данных;
4. низкие требования к клиентской машине - за счет выполнения основной бизнес-логики на сервере приложений.

Из недостатков:

1. сложность реализации;
2. сложность развертывания и поддержки.

API (application programming interface) позволяет упростить процесс разработки приложений, абстрагируя базовую реализацию и предоставляя только объекты или действия, необходимые разработчику. В общем виде — это набор способов и правил, по которым различные программы общаются между собой и обмениваются данными. Все эти взаимодействия происходят с помощью функций, классов, методов, структур, а иногда констант одной программы, к которой обращаются другие. Это основной принцип работы API.

Отличный пример хорошо организованного и задокументированного web-API можно подглядеть, например, в музыкальном сервисе Spotify.

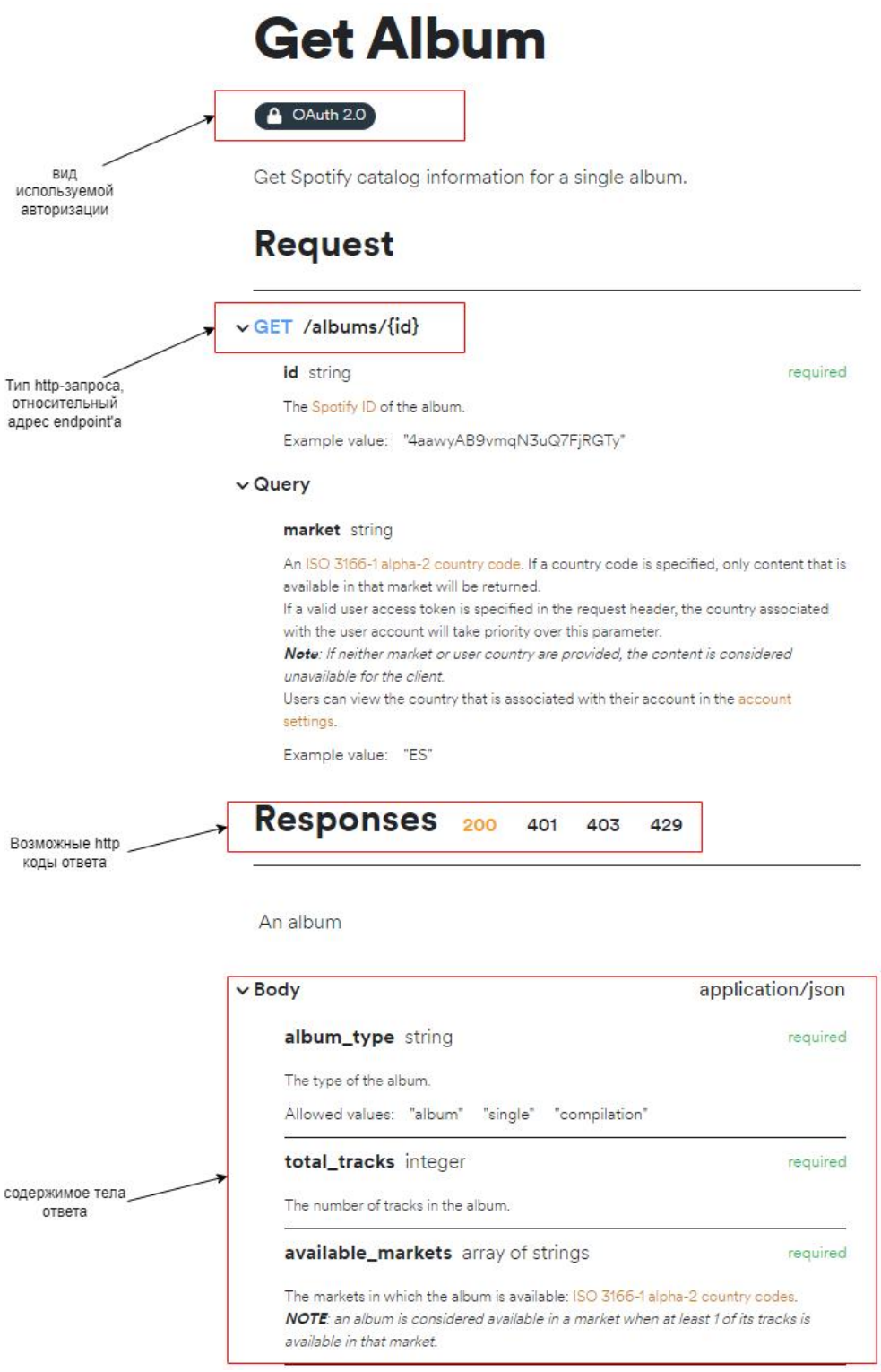

Рис. 2. Описание http-endpoint'a сервиса Spotify

Как можно заметить - в документации к этому web-API зафиксированы контракты взаимодействия между клиентом и сервером, описаны входные и выходные данные, а также все возможные варианты кодов ошибок, которые могут возникнуть в процессе обработки запроса клиента.

Задание

Необходимо разработать программный комплекс, соответствующий следующим требованиям:

1. комплекс должен реализовывать трехуровневую архитектуру;
2. протокол взаимодействия между клиентом и сервером может быть любым (http, gRPC и др.);
3. клиент может быть любым (браузерное приложение, мобильное приложение, PC приложение)
4. клиентское приложение должно взаимодействовать с серверным через API;
5. в базе данных должно быть **не менее** 5 таблиц данных, каждая из которых должна быть задействована в бизнес-логике приложения.

Будет плюсом, если:

- клиент и сервер будут поддерживать какой либо-механизм авторизации.

Задание 2. Электронно-цифровая подпись. Работа с почтой.

В общем виде работа с электронно-цифровой подписью (ЭЦП) происходит по следующей схеме:

Рис. 3 Общий принцип работы ЭЦП

Механизм ЭЦП базируется на идеях криптосистем с открытым ключом (ассиметричном шифровании).

В ассиметричном шифровании применяется пара ключей: открытый (Public key) и закрытый, также называемый секретным (Private key). Открытый и закрытый ключи в данном случае позволяют криптографическому алгоритму шифровать и дешифровать сообщение. При этом сообщения, зашифрованные открытым ключом, расшифровать можно только с помощью закрытого ключа. Открытый ключ публикуется в сертификате владельца и доступен подключившемуся клиенту, а закрытый – хранится у владельца сертификата.

Открытый и закрытый ключ между собой связаны математическими зависимостями, поэтому подобрать открытый или закрытый ключ невозможно за короткое время (срок действия сертификата).

Предположим, что у нас есть 2 абонента – А и Б. В таком случае основная часть протокола взаимодействия пользователей при использовании ЭЦП на базе алгоритма RSA состоит из следующих шагов:

1. Сначала пользователь А шифрует хэш сообщения своим закрытым ключом. В результате этих действий пользователь А подписывает сообщение.

Подписанным сообщением обычно считается открытый текст сообщения + зашифрованный хэш.

Подписанное сообщение может быть зашифровано перед отправкой.

2. Полученная последовательность передается пользователю Б.
3. Пользователь Б получает подписанное сообщение (если оно зашифровано – расшифровывает), и, используя открытый ключ пользователя А, расшифровывает хэш.
4. Затем Б считает хэш от полученного открытого текста сообщения и сверяет его с расшифрованным.

Если в результате хэши совпадают, то мы можем быть уверены, что сообщение отправил именно *абонент* А. Данная схема позволяет защититься от нескольких видов возможных нарушений, а именно:

- пользователь А не может отказаться от своего сообщения, если он признает, что секретный ключ известен только ему;
- нарушитель без знания секретного ключа не может ни сформировать, ни сделать осмысленное изменение сообщения, передаваемого по линии связи.

В описанной схеме взаимодействия, при необходимости, можно без больших затруднений добавить шифрование пересылаемого сообщения. Как правило это зависит от области применения ЭЦП.

Задание

Необходимо разработать программное решение, позволяющее выполнить следующие сценарии:

Сценарий 1:

1. Клиент подписывает сообщение.
2. Клиент обращается к серверу и передает ему подписанное сообщение.
3. Сервер осуществляет верификацию подписанного сообщения.
4. Клиент проверяет статус верификации, возвращенный ему сервером.

Сценарий 2:

1. Клиент запрашивает публичный ключ сервера.

2. Клиент запрашивает генерацию случайного сообщения на сервере.
3. Сервер генерирует сообщение, подписывает его.
4. Клиент осуществляет верификацию полученного сообщения.

Для ЭЦП следует использовать RSA + SHA256.

Задание 3. Комбинированное

Чешский национальный банк предоставляет возможность отслеживать валютный курс чешской кроны.

Ежедневный курс доступен по адресу:

https://www.cnb.cz/en/financial_markets/foreign_exchange_market/exchange_rate_fixing/daily.txt?date=27.07.2019

Исторические данные доступны по адресу:

https://www.cnb.cz/en/financial_markets/foreign_exchange_market/exchange_rate_fixing/year.txt?year=2019

Задание

Необходимо разработать программное решение, обладающее следующей функциональностью:

1. синхронизация данных по чешской кроне в БД по расписанию. Должна быть возможность сконфигурировать время/интервал запуска. Например: запускать синхронизацию каждый день в 0:01. Период запуска должен задаваться конфигурации приложения. Валюты, по которым синхронизируются данные, должны быть конфигурируемыми.
2. синхронизация данных по чешской кроне за период времени. На вход подается startDate и endDate, приложение синхронизирует в БД данные за этот период. Валюты, по которым синхронизируются данные, должны быть конфигурируемыми.
3. предоставляет web-API, с помощью которого можно получить отчет по курсу кроны за период времени. В отчете необходимо вывести минимальное, максимальное и среднее значение каждой из выбранных валют отдельно. Валюты, по которым строится отчёт, передаются в запросе. Показатели в отчёте необходимо рассчитывать для валюты в количестве 1 условная единица, т.е. для Amount = 1. Формат отчета – JSON.

Будет плюсом, если:

Есть тесты на логику формирования отчета.

Необходимо учесть, что в данных, предоставляемых API, могут быть аномалии. Например, для некоторых временных интервалов может не быть курсов определенных валют и т.п.