# Задание 1. Трехзвенная (клиент-серверная) архитектура

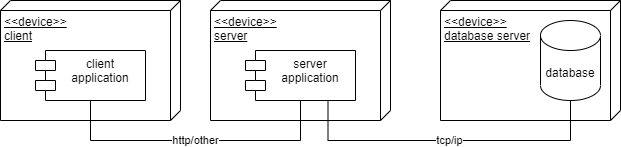


Рис. 1 Обобщенная диаграмма развертывания приложения на основе трехзвенной архитектуры

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 web-API можно подробнее рассмотреть, например, в музыкальном сервисе [Spotify](https://developer.spotify.com/documentation/web-api/reference/#/operations/get-an-album).

Graphical user interface

Description automatically generated

Рис. 2. Описание http-endpoint'а 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:

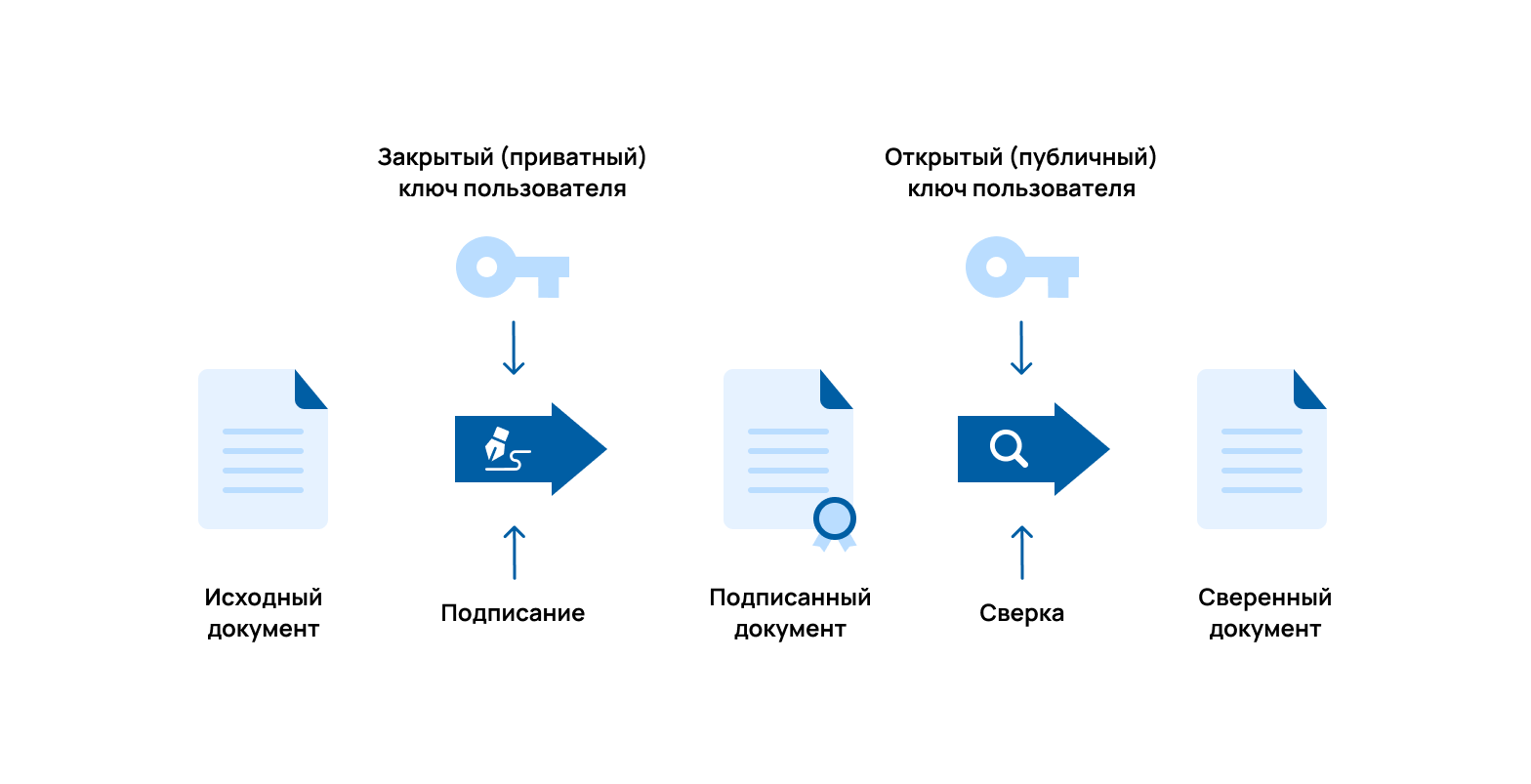


Рис. 3 Общий принцип работы ЭЦП

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 – A 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 сообщение (или его хэш) своим закрытым ключом. В результате этих действий пользователь А подписывает сообщение.
2. Затем пользователь А, шифрует то, что получилось на шаге 1 открытым ключом пользователя Б. На этом этапе сообщение шифруется, чтобы никто посторонний не мог его прочитать.
3. Полученная последовательность передается к пользователю Б.
4. Пользователь Б получает последовательность и расшифровывает ее, используя свой закрытый ключ.
5. Затем Б, используя открытый ключ пользователя А, расшифровывает оригинальное сообщение.

В результате *абонент* Б получает исходное сообщение и убеждается в том, что его 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.

Данная схема позволяет избежать многих конфликтных ситуаций. Иногда нет необходимости зашифровывать передаваемое сообщение, но нужно его скрепить электронной подписью. В этом случае из приведенного выше протокола исключаются шаги 2 и 4, то есть текст шифруется закрытым ключом отправителя, и полученная последовательность присоединяется к документу. Получатель с помощью открытого ключа отправителя расшифровывает прикрепленную подпись, которая, по сути, является зашифрованным повторением основного сообщения. Если расшифрованная подпись совпадает с основным текстом, значит, подпись верна.

## 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. Клиент проверяет статус верификации, возвращенный ему сервером.

Сценарий 2:

1. Клиент запрашивает публичный ключ сервера.
2. Клиент запрашивает генерацию случайного сообщения на сервере.
3. 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 за текущую дату 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.
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 в конфигурации приложения.
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.
4. Было бы неплохо иметь тесты, хотя бы на базовые сценарии 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.