Содержание

Введение…….................................................................................................2

План работы………………………………………………………………...3

1 Общая часть …………………...................................................................4

1.1 Предметная область ЭКГ …………………………………...................4

1.1.1 Отведения ЭКГ…………………….....................................................5

1.2 Процедура ЭКГ………………………………………………………..11

1.2.1 Снятие ЭКГ………………………………………………………….12

1.2.2 Рекомендация перед плановой процедурой ЭКГ…………………13

Введение

Совершенствование приборного обеспечения современной медицины — задача, связанная с повышением качества жизни граждан РФ. В некоторых медицинских учреждениях находятся приборы ЭКГ, которые не соответствуют современным тенденциям развития информационных технологий, в частности, сетей передачи данных, цифровых архивов хранения информации — электронной кары больного, а также унификации протоколов и форматов хранения данных. Основная идея разработки нового прибора ЭКГ — это создание аппарата, информация с которого будет интегрирована с информационными системами медицинского учреждения, что также находится в общей тенденции развития современных технологий «интернета вещей». Передача данных на экран прибора из информационной системы и результата измерений обратно автоматически позволило бы значительно сократить время приема пациента, а также его нахождения в медицинском учреждении. Для решения этой задачи необходимо разработать недорогой прибор ВКР на доступной элементной базе и модулей.

Актуальность задачи

- Оцифровка ЭКГ в память микроконтроллера с последующей передачей на сервер (Orange Pi miniPC)

- Компактность устройства

Цели и Задачи

Цели:

- разработать прототип аппарата ЭКГ, позволяющего обеспечить доступ к данным по протоколам интернета.

Задачи:

1. Изучить предметную область ЭКГ-исследований
2. Спроектировать схему ЭКГ прибора
3. Разработать программный код подсистем
4. Демонстрация устройства

Требования к результату

1. Разработать систему для ЭКГ устройства
2. Передача данных через Orange Pi PC H3.

Используемые ПО

STM32 конфигуратор;

EasyEDA;

Язык программирования Forth;

Rest Interface (Python) и JSON.

1 Общая часть

* 1. Предметная область ЭКГ

Электрокардиография – одна из основополагающих методик исследования, не теряющая своей актуальности и по сей день. Она позволяет врачу любого профиля определять функциональное состояние человеческого сердца, выявлять наличие возможных патологии и затем, фиксировать это на кардиограмме. Методика ЭКГ проста, но для ее использования предусмотрен широкий спектр определенных технологий. В данной категории, можно найти основные и редко используемые варианты подключения ЭКГ, правила наложения электродов для записи электрокардиограммы.

Для получения полной картины происходящего электроды прикрепляются на грудь и конечности в зависимости от выбранного исследования, с которых снимаются сигналы электрической активности сердца.

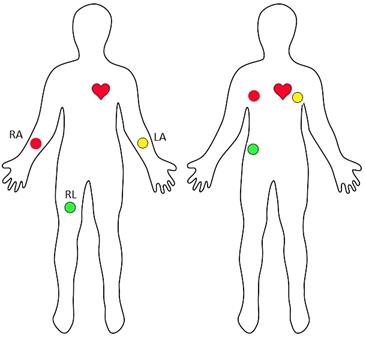


Рис.1 Подкрепление электродов

Электрическая система сердца заправляет генерацией и распространением электрических сигналов по сердечной мышце, само сердце периодически сокращается и расслабляется, перекачивая кровь. В течении цикла работы сердца происходит упорядоченный процесс деполяризации. Деполяризация – это резкое изменение электрического состояния клетки, когда отрицательный внутренний заряд клетки становится на короткое время положительным. В сердце деполяризация начинается в специализированных клетках в синусно-предсердном узле. Далее волна распространяется через предсердно-желудочковый узел вниз и приводит к сокращению желудочков. В отличие от других нервных клеток, которые неспособны генерировать электрический сигнал в автоматическом режиме, клетки синусно-предсердного узла способны создавать ритмичный электрический сигнал без внешнего воздействия. Точнее, внешние воздействия, такие как: физическая нагрузка, которая влияет только на частоту колебаний. При этом происходит периодическая деполяризация и реполяризация. Реполяризация - это когда электрическое состояние клетки имеет положительный внутренний заряд клетки становится на короткое время отрицательным. В электрокардиостимуляторе также имеется генератор стабильной частоты, выполняющий роль синусно-предсердного узла. Из-за того, что процессы в клетках электрохимические, а не электрические, деполяризация и реполяризация в них происходят намного медленнее.

Все это необходимо знать и учитывать для того, чтобы правильно расположить на теле пациента электроды, которые будут обнаруживать небольшие изменения колебаний на коже, при каждом сокращении сердечной мышцы.

1.1.1 Отведение ЭКГ

В Электрокардиографии записано не сколько отведений:

3 стандартных отведения:

I - левая рука (+) и правая рука (-),

II - левая стопа (+) и правая рука (-),

III - левая нога (+) и левая рука (-);

3 усиленных униполярных отведения от конечностей:

aVR - усиленное отведение от правой руки,

aVL - усиленное отведение от левой руки,

aVF - усиленное отведение от левой ноги;

6 грудных назначений:

V1, V2, V3, V4, V5, V6;

Также возможно удаление дополнительных отведений:

3 дополнительных грудных отведения диагностика по целям где присутствует очаг изменений миокарда в задней области ЛЖ:

V7, V8, V9;

3 биполярных отведения. Дополнительная диагностика очаговых изменений миокарда в заднем, переднебоковом и верхнем отделах передней стенки ЛЖ:

D - дорсальный, I - нижний, A - передний.

Также встречаются крайне редкие варианты отведений:

Отведение S5 - используется для плохо выраженного предсердного комплекса ЭКГ, это помогает в диагностике желудочковых и над желудочковых нарушений сердечного ритма.

Ортогональные отведения - ЭКГ снимается в трех грудных отведениях. Самыми простыми отведениями являются X, Y, Z. Ось этих отведений перпендикулярна друг другу.

Пищеводные отведения - используются для выявления предсердного комплекса ЭКГ[14]. Для регистрации электрода, который подключен к кардиографу, вводят в пищевод с помощью зонда для определения возбуждений предсердий. Эта методика помогает в диагностике различных аритмий.

Внутрисердечные отведения - используются для регистрации ЭДС(электродвижущая сила) сердца в полости предсердия или желудочка. Для этого в процессе зондирования в полость сердца вводят специальный электрод-зонд.

Отведения по Арриги.

Ось отведений по Арриги расположена в плоскости, что образует треугольник, в центре которого расположено сердце.

При любом варианте расположения сердца в грудной клетке одна из осей остается параллельной задней стенке левого желудочка и улавливает признаки инфаркта миокарда несколько лучше, чем в стандартных отведениях III и aVF.

ЭКГ снимают в отведениях по Арриги в следующих положениях: в первом положении регистрируется отведение А1, во втором положении – отведение А2, в третьем-А3.

Обозначения:

- RCA - Right Coronary Artery (правая коронарная артерия);

- SVC - Superior Vena Cava (верхняя полая вена);

- IVC - Inferior Cava (нижняя полая вена);

- RA - Right Atrium (правое предсердие);

- RV - Right Ventricle (правый желудочек);

- ПМЖВ - левая передняя нисходящая артерия (передняя нисходящая артерия);

-LV - Left Ventricle (левый желудочек);

-LCX - левая артерия (огибающая артерия).

Если рассмотреть расположение органов в теле человека относительно скелета, а точнее где находится сердце, то на переднюю поверхность грудной клетки проецируется 2/3 правого сердца и 1/3 левого желудочка. Поскольку левый желудочек электрически активен и сильнее, топография ЭКГ воспринимается несколько иначе: 2/3 передней стенки занимает левый желудочек, а 1/3 правого края занимает правый желудочек. Таким образом нижняя и левая боковые стенки представлены левым желудочком. Проще говоря, что первые два грудных электрода (V1, V2) стоят на границе правого и левого желудочков, то есть на перегородке. Таким образом, именно они демонстрируют как электрофизиологические характеристики левого желудочка, так и активность правого.

Отведя от конечностей, если вы посмотрите сердце в вертикальной плоскости, но только нижнюю и боковую стенки то обнаружим следующие отведения.

Боковая стенка: отводы l и aVL.

Нижняя стенка: III, aVF и II.

Отведения на Груди будет показывать сердце в горизонтальной плоскости, своеобразным полукругом. Первые четыре отведения показывают переднюю стенку, а два последних-боковые.

-V1-V2 - раздел;

-V3-V4 - собственно передняя стенка;

-V4 - верх.

-V5-V6 - боковая стенка.

Дополнительные грудные отведения: V7-V9 показывают заднюю стенку, а дополнительные правые грудные отведения: V3R и V4R показывают правый желудочек.

1.1.2 Расположение электродов. Точки наложения электродов для ЭКГ.

В стандартных отведениях и в усиленных отведениях от конечностей расположены электроды таким образом:

Красный-правая рука,

Желтый-левая рука,

Зеленый-левая нога,

Черный - правая ступня.

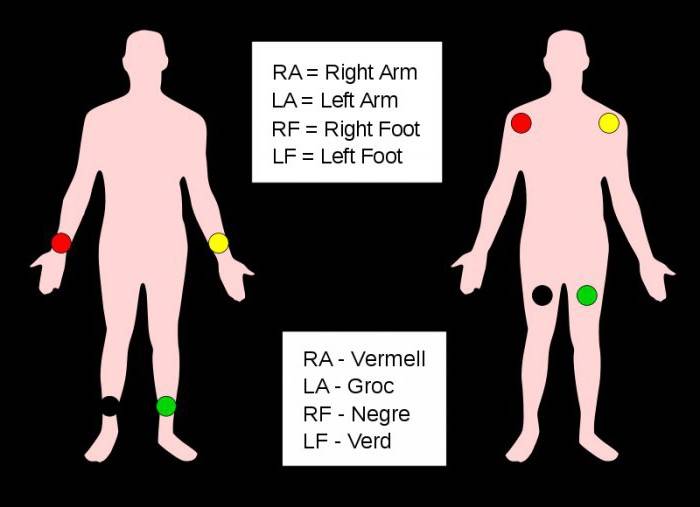


Рис.2 Подключение электродов

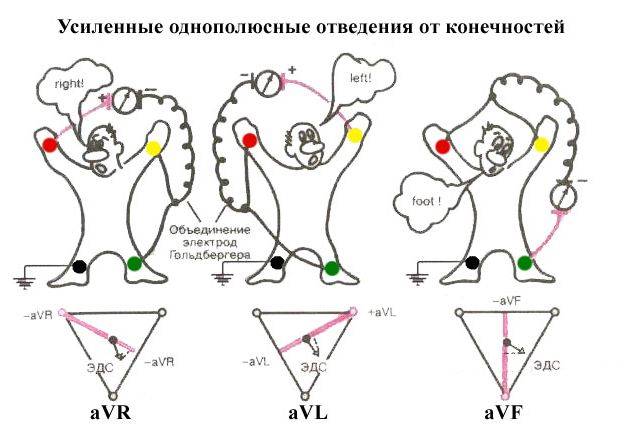


Рис.3 Усиленные однополюсные отведения от конечностей

В грудных отведениях расположены электроды:

V1 (красный) - в четвертом межреберье по правому краю грудины,

V2 (желтый) - в четвертом межреберье по левому краю грудины,

V3 (зеленый) – на уровне пятого ребра по левому краю над грудинной, между четвертым и вторым электродами,

V4 (коричневый) - в пятом межреберье по левой срединно-ключичной линии,

V5 (черный) - по горизонтальной линии V4 по левая передняя подмышечная линия,

V6 (синяя) - на горизонтальной линии V4-V5 по левой средней подмышечной линии.

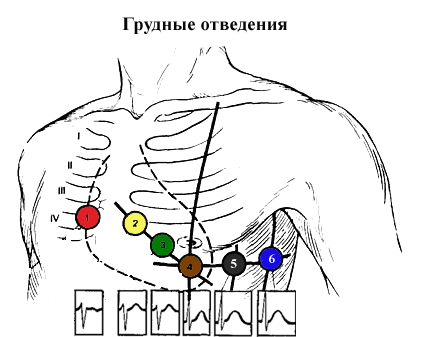


Рис.4 Грудные отведения

В дополнительных грудных отведениях расположены электроды:

V7 - на уровне V4-V6 по левой задней подмышечной линии,

V8 - на уровне V4-V6 по левой лопаточной линии,

V9 - на уровне V4-V6 подлопаточной линии, левая околопозвоночная линия.

В отведениях по Небу расположены электроды:

Красный стандарт - во втором межреберье по правому краю грудины,

Зеленый стандарт - в пятом межреберье по левой срединно-ключичной линии,

Желтый стандарт - на горизонтальной линии с зеленый электрод по задней подмышечной линии.

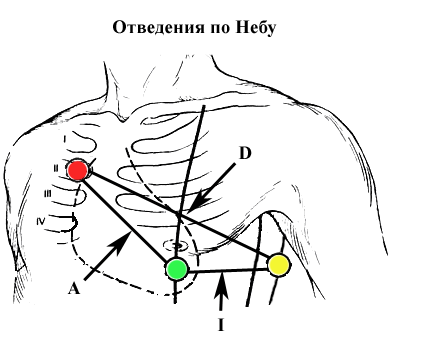


Рис.5 Отведения по Небу

В ортогональных отведениях по Франку расположены:

Грудные электроды устанавливают на уровне пятого межреберья, когда пациент сидит, и на четвертом уровне в положении лежа на спине. Расположение электродов, следующее:

Точка E расположена по средней линии грудины;

Точка М - на позвоночнике, симметрично точке Е;

Точка А - от левой средней подмышечной линии; точка С - между электродами Е и А;

Точка I - по правой средней подмышечной линии;

Точка H находится на затылке или на голове, а точка F - на левой ступне.

Полярность, предложенная Франком, следующая:

Отведение X - горизонтальная пространственная составляющая получается переключением электродов E, C и A положительный полюс и I отрицательный полюс;

Отведение Z (срединно-пространственная составляющая) - электроды А и М с положительным полюсом и 1, Е.

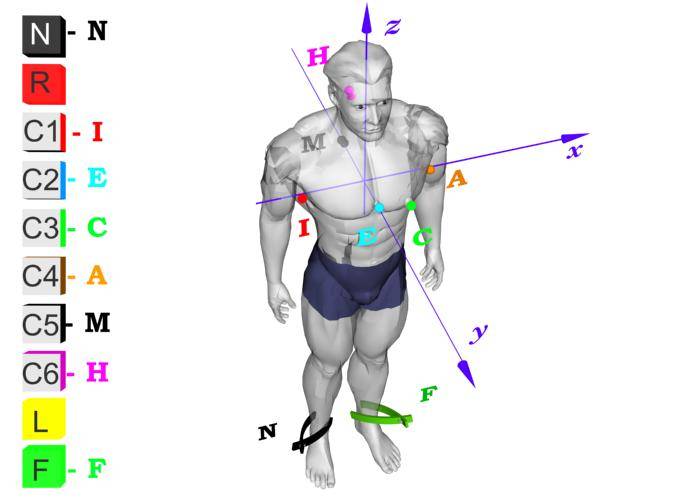


Рис.6 Отведения по Арриги

В отведениях Арриги расположены электроды:

Желтый положительный электрод с плоской пластиной укреплен под углом левой лопатки,

Красный отрицательный электрод на присоске, который находится выше середины левой ключицы,

Зеленый – на левая голень.

1.2 Процедура ЭКГ.

Прежде чем научиться снимать ЭКГ, подумайте, какие действия нужно предпринять, чтобы подготовить пациента.

Аппарат ЭКГ есть в каждом лечебном учреждении, он находится в отдельной комнате с кушеткой для удобства пациента и медперсонала. Помещение должно быть светлым и уютным, с температурой воздуха + 22 ... + 24 градуса Цельсия. Поскольку правильно снять ЭКГ можно только в том случае, если пациент полностью спокоен, такая среда очень важна для данной манипуляции. Поместите испытуемого на медицинскую кушетку. В положении лежа тело легко расслабляется, что важно для будущей записи кардиографа и оценки работы самого сердца. Перед наложением электродов для ЭКГ следует обработать ватным тампоном, смоченным медицинским спиртом, необходимые участки рук и ног пациента. Обработка этих мест проводится физиологическим раствором или специальным лечебным гелем, предназначенным для этих целей. Пациенту необходимо сохранять спокойствие во время записи кардиографа, дышать ровно, умеренно, не беспокоиться.

Вам необходимо знать, в какой последовательности нужно накладывать электроды. Для удобства персонала, проводящего эту манипуляцию, изобретатели аппарата ЭКГ выделили электроды четырех цветов: красный, желтый, зеленый и черный. Они накладываются именно в таком порядке и никак иначе, иначе делать ЭКГ не рекомендуется. Спутать их просто недопустимо. Поэтому медперсонал, работающий с аппаратом ЭКГ, проходит специальную подготовку с последующей сдачей экзамена и получением допуска или сертификата на работу с данным аппаратом. Медицинский работник в кабинете ЭКГ согласно своей рабочей инструкции должен четко знать места наложения электродов и правильно выполнять последовательность.

Итак, электроды для рук и ног выглядят как большие зажимы, но ничего страшного, зажим располагается на конечности совершенно безболезненно, эти зажимы разного цвета и накладываются на определенные места на теле следующим образом:

* Красный - запястье правой руки.
* Желтый - запястье левой руки.
* Зеленая - левая нога.
* Черная - правая нога.

Рисунок с отведением указан выше (рис 2)

Алгоритм действий, следующий:

1. Подготовка пациента к будущим манипуляциям. Укладывая его на кушетку, медработник просит расслабиться и не напрягаться. Удалите все ненужные предметы, если они есть, которые могут помешать записи кардиографа. Освободите необходимую кожу от одежды.
2. Приступайте к наложению электродов строго в определенной последовательности и порядке наложения электродов.
3. Подключайте устройство к работе.
4. После того, как устройство подключено и готово к работе, начните запись.
5. Удалите бумагу с записанной электрокардиограммой сердца.
6. Результат ЭКГ передается пациенту или врачу на руки для последующей расшифровки.

1.2.1 Снятие ЭКГ

Запись проанализирует, насколько регулярны сердечные сокращения, покажет частоту сердечных сокращений, фокус, проводящую способность сердечной мышцы, определение сердца по отношению к осям и состояние сердечных зубов.

Сразу после прочтения кардиограммы врач сможет поставить диагноз и назначить лечение, либо даст необходимые рекомендации, что значительно ускорит процесс выздоровления или избавит от серьезных осложнений, а главное, вовремя проведенная ЭКГ может сэкономить жизнь человека.

1.2.2 Рекомендация перед плановой процедурой ЭКГ

Перед снятием ЭКГ пациента необходимо проинструктировать о том, какие условия необходимо соблюдать накануне и в день удаления.

* Накануне рекомендуется избегать нервного напряжения, а продолжительность сна должна быть не менее 8 часов.
* Исключите продукты, влияющие на работу сердца, на 1 день, например, крепкий кофе или чай, острые приправы, алкогольные напитки, курение.
* Не наносите крем и лосьоны на кожу рук, ног, груди, жирные кислоты, которые могут впоследствии ухудшить проводимость медицинского геля на коже перед наложением электродов.
* Абсолютное спокойствие необходимо перед сдачей ЭКГ во время самой процедуры.
* Обязательно исключите физические нагрузки в день процедуры.
* Перед самой процедурой нужно спокойно посидеть минут 15-20, дыхание спокойное, равномерное.

Если у обследуемого наблюдается сильная одышка, то ему нужно делать ЭКГ не лежа, а сидя, так как именно в этом положении тела аппарат может четко фиксировать сердечную аритмию.

Есть определенные условия, при которых проводить ЭКГ категорически нельзя, а именно:

* При остром инфаркте миокарда.
* Нестабильная стенокардия.
* Сердечная недостаточность.
* Некоторые виды аритмий неясной этиологии.
* Тяжелые формы стеноза аорты.
* Синдром тромбоэмболии легочной артерии (тромбоэмболия легочной артерии).
* Расслоение аневризмы аорты.
* Острые воспалительные заболевания сердечной мышцы и мышцы перикарда.
* Тяжелые инфекционные заболевания.
* Тяжелое психическое заболевание.

Особенность ЭКГ

Электрокардиограмма проводится в отдельном кабинете медицинского учреждения, хотя при необходимости ее можно записать дома, в палате пациента или в машине скорой помощи. Обозначенное помещение должно находиться на достаточном расстоянии от возможных помех в виде источников электричества. Кушетка располагается на расстоянии 1,5-2 м от шнура питания. Также рекомендуется экранировать кушетку, для чего нужно использовать одеяло с металлической сеткой, имеющей заземление.

Запись ЭКГ обычно проводится в положении пациента, лежащего на кушетке. При наличии противопоказаний пациент может оставаться в сидячем положении во время электрокардиографии.

Перед процедурой пациенту следует избегать чрезмерных физических нагрузок, употребления напитков и еды, активизирующих работу сердца.

2 Специальная часть.

2.1 Что такое Rest interface?

Representational State Transfer — это архитектурный стиль взаимодействия компонентов распределённого приложения в сети. Архитектурный стиль – это набор согласованных ограничений и принципов проектирования, позволяющий добиться определённых свойств системы.

Назначение REST в том, чтобы придать проектируемой системе такие свойства как:

* Производительность,
* Масштабируемость,
* Гибкость к изменениям,
* Отказоустойчивость,
* Простота поддержки.

Принципы REST

Давайте рассмотри 6 принципов REST — ограничения, которые и помогают нам добиться этих нефункциональных требований.

6 принципов REST:

1. Клиент-серверная архитектура
2. Stateless
3. Кэширование
4. Единообразие интерфейса
5. Layered system
6. Code on demand

Далее мы рассмотрим поподробнее данные принципы.

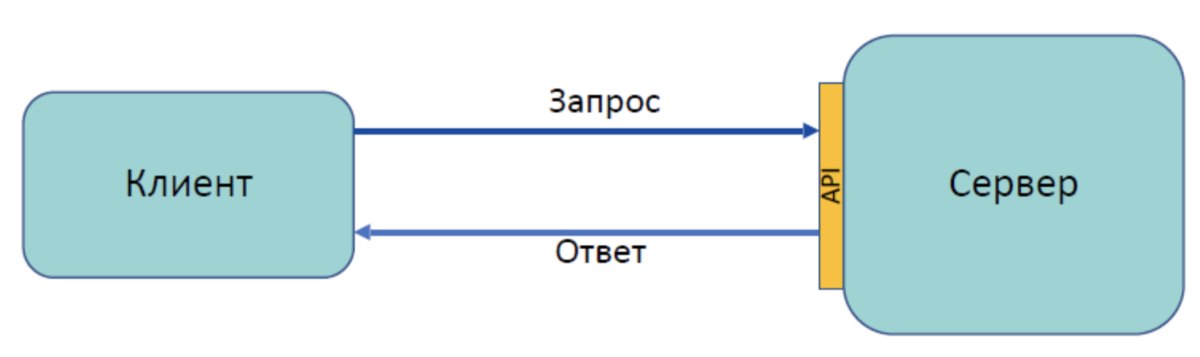
Принцип 1. Клиент-серверная архитектура

Сама концепция клиент-серверной архитектуры заключается в разделении некоторых зон ответственности: в разделении функций клиента и сервера. Что это означает?

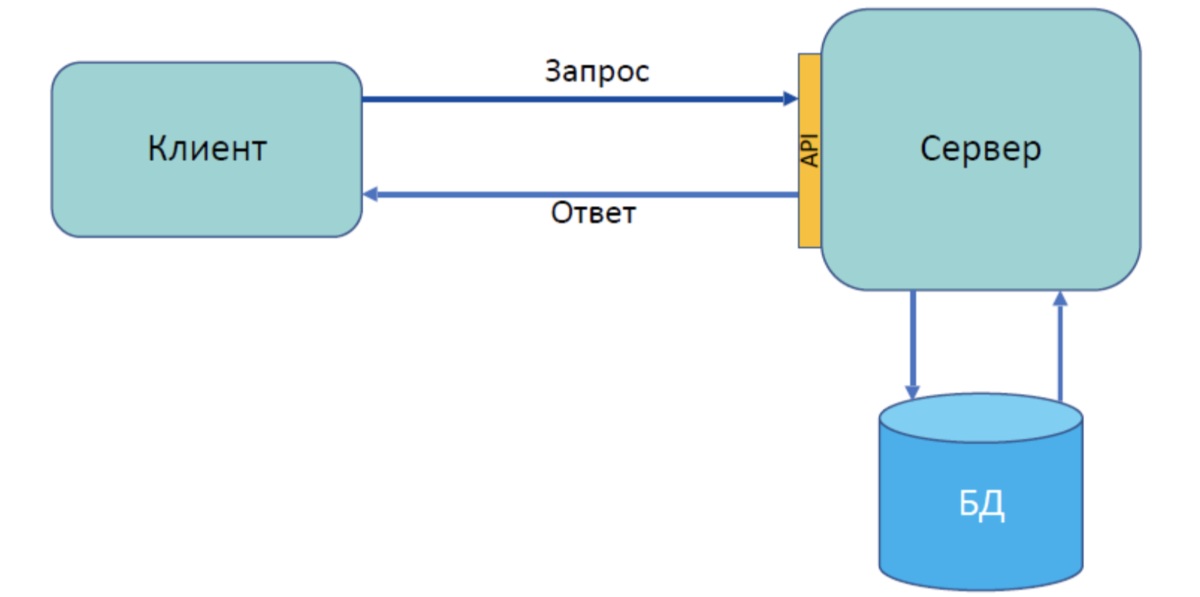
Например, мы разделяем нашу систему так, что клиент (допустим, это мобильное приложение) реализует только функциональное взаимодействие с сервером. При этом сервер реализует в себе логику хранения данных, сложные взаимодействия со смежными системами и т.д.

Что мы этим добиваемся и как могло бы быть иначе? Давайте представим, что клиент и сервер у нас объединены. Тогда, если мы говорим о мобильном приложении, каждое мобильное приложение каждого клиента должно было бы быть абсолютно самодостаточной единицей. И тогда, поскольку у нас единого сервера нет для получения/отправки информации, у нас получилась бы какая-то сеть единообразных компонентов – например, мобильные приложения общались бы друг с другом – такая распределённая сеть равноценных узлов.

Например, в блокчейне. Тем не менее, в случае с REST мы говорим о том, что разделяем ответственность. Например, отображение информации, её обработку и хранение.

Клиент-серверная архитектура

Также сервер может иметь базу данных (см. рисунок ниже). В данном случае надо понимать, что пара «сервер и БД» тоже будет парой «клиент-сервер». Только в данном случае сервером будет БД, а сам сервер — клиентом.

Трёхзвенная архитектура

Что дает клиент-серверная архитектура и зачем она нужна?

Во-первых, клиент-серверная архитектура дает нам определённую масштабируемость: есть сервер, есть единая точка обработки запросов. При необходимости выдерживать большую нагрузку мы можем поставить несколько серверов. Также к нему можно подключать достаточно большое количество клиентов (сколько сможет выдержать). Таким образом, клиент-серверная архитектура позволяет добиться масштабируемости.

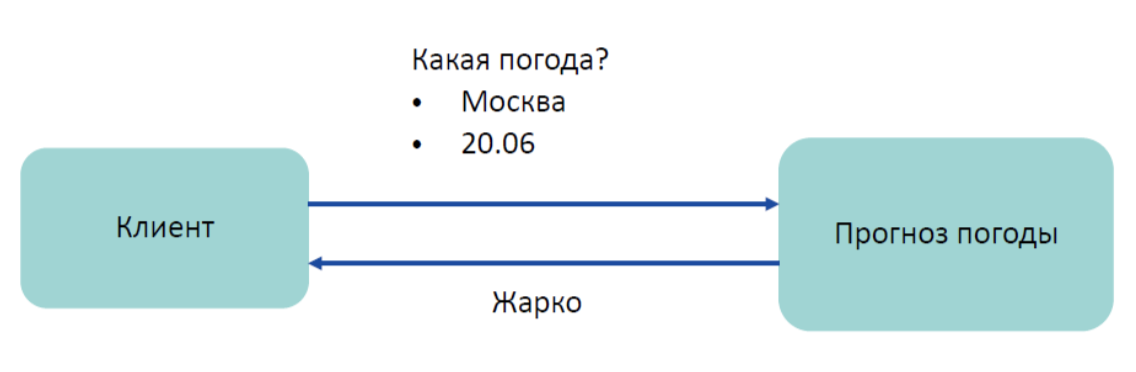
Во-вторых, REST даёт определённую простоту поддержки. Если мы хотим изменить логику обработки информации на сервере, то выполним эти изменения на сервере. В данном случае мы можем и не менять каждого клиента, как если бы они были абсолютно равноценной сетью.

Конечно, есть и минусы. В случае с клиент-серверной архитектурой мы понимаем, что у нас есть единая точка отказа в виде сервера. Если отказал сервер и у нас нет дополнительных инстансов, то для нас это будет означать неработоспособность системы.

Также потенциально может увеличиться нагрузка, поскольку часть логики мы вынесли с клиента на сервер. Клиент будет совершать меньше каких-либо действий самостоятельно, соответственно, у нас возрастёт количество запросов между клиентом и сервером.

Принцип 2. Stateless

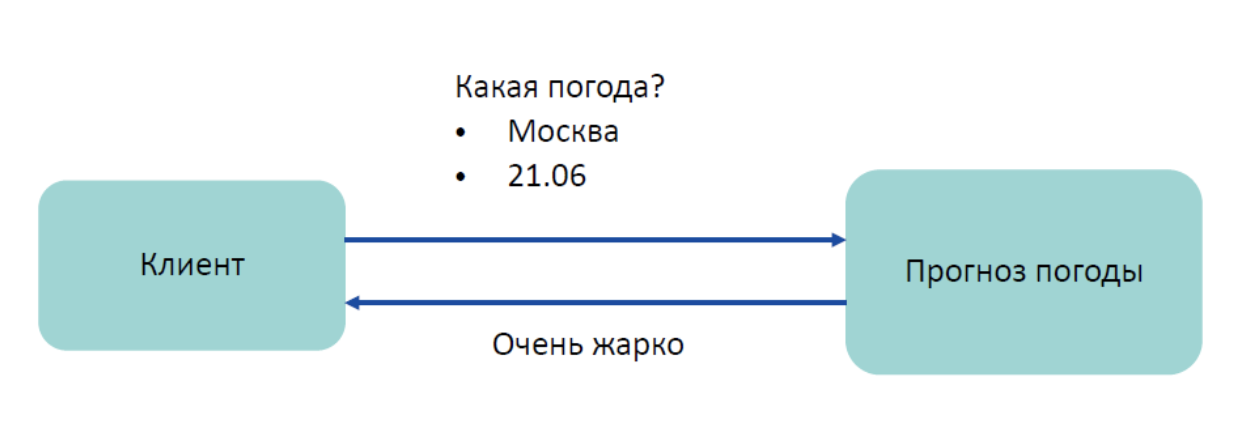
Принцип заключается в том, что сервер не должен хранить у себя информацию о сессии с клиентом. Он должен в каждом запросе получать всю информацию для обработки.

Пример реализации принципа Stateless.

Представим, что у нас есть некоторый сервис прогноза погоды, в котором уже реализована клиент-серверная архитектура, и мы хотим получить сообщение о прогнозе погоды на завтра.

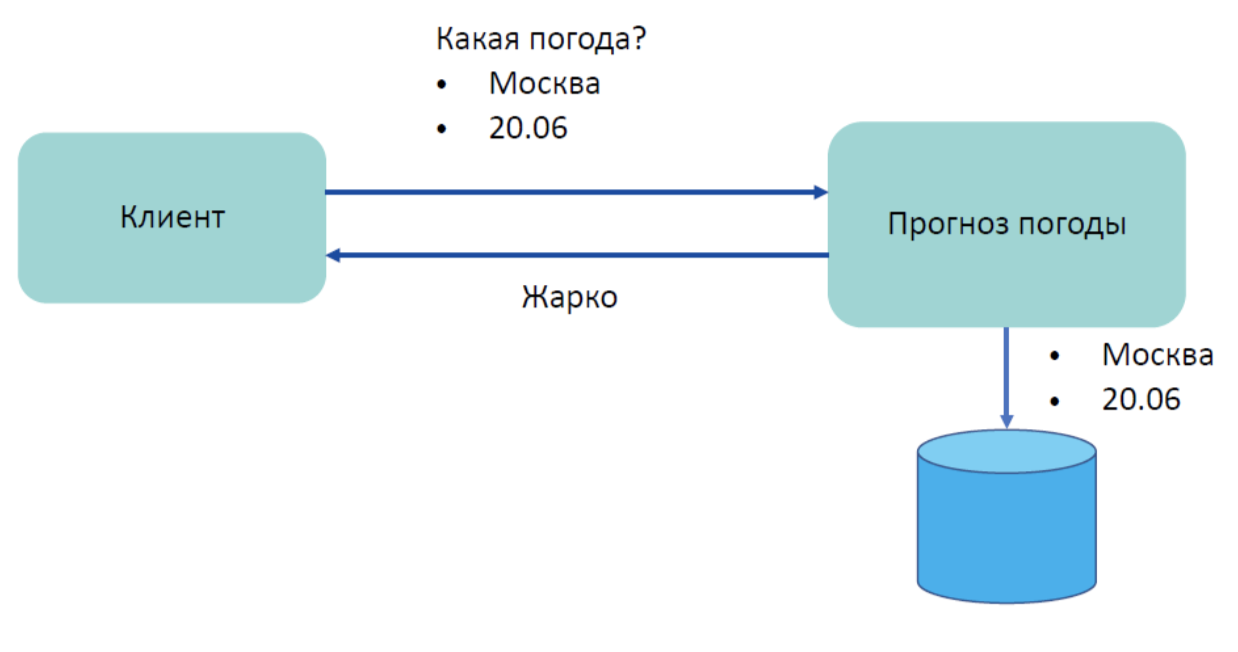
Что мы делаем в случае, если мы работаем с Stateless? Мы отправляем запрос «Какая погода?», отправляем место, где хотим погоду узнать, и дату. Соответственно, прогноз погоды отвечает нам — «Будет жарко».

Если я захочу узнать, какая будет погода через день, то опять укажу место, где хочу узнать погоду, укажу другую дату. Сервер получит этот запрос, обработает и сообщит мне, что там уже будет очень жарко.

Пример реализации принципа Stateless.

Рассмотрим ситуацию: что было бы, если бы у нас не было Stateless? В таком случае у нас бы был Stateful. В этом случае сервер хранит информацию о предыдущих обращениях клиента, хранит информацию о сессии, какую-то часть контекста взаимодействия с клиентом. А затем может использовать эту информацию при обработке следующих запросов.

Приведём пример на рисунке:

Пример реализации принципа Stateful

В этом случае у сервера хранится некоторый контекст. Он понимает, что я у него спрашиваю про 21-е число и могу дать ответ на основе информации, хранимой у него в БД или в кэше. Один из примеров, где можно встретить подход Stateful в жизни — это работа с FTP-сервером.

Вернёмся к Statless-подходу. Почему в REST-архитектуре мы должны использовать именно Statless-подход?

Какие он даёт плюсы?

* Масштабируемость сервера,
* Уменьшение времени обработки запроса,
* Простота поддержки,
* Возможность использовать кэширование.

В первую очередь, это масштабирование сервера. Если каждый запрос содержит в себе абсолютно весь контекст, необходимый для обработки, то можно, например, клонировать сервер-обработчик: вместо одного поставить десять таких. Мне будет абсолютно неважно, в какой из этих клонов придёт запрос. Если бы они хранили состояние, то либо должны были синхронизироваться, либо мне нужно было бы умело направлять запрос в нужное место.

Помимо этого, появляется простота поддержки. Каждый раз я вижу в логах, какое сообщение приходило от клиента, какой ответ он получал. Мне не нужно дополнительно узнавать о том, какое состояние хранил сервер.

Также подход Stateless позволяет использовать кэширование.

Какие проблемы может создать Stateless-подход?

* Усложнение логики клиента (именно на стороне клиента нам нужно хранить всю информацию о состоянии, о допустимых действиях, о недопустимых действиях и подобных вещах).
* Увеличение нагрузки на сеть (каждый раз мы передаём всю информацию, весь контекст. Таким образом, больше информации гоняем по сети).

Принцип 3. Кэширование

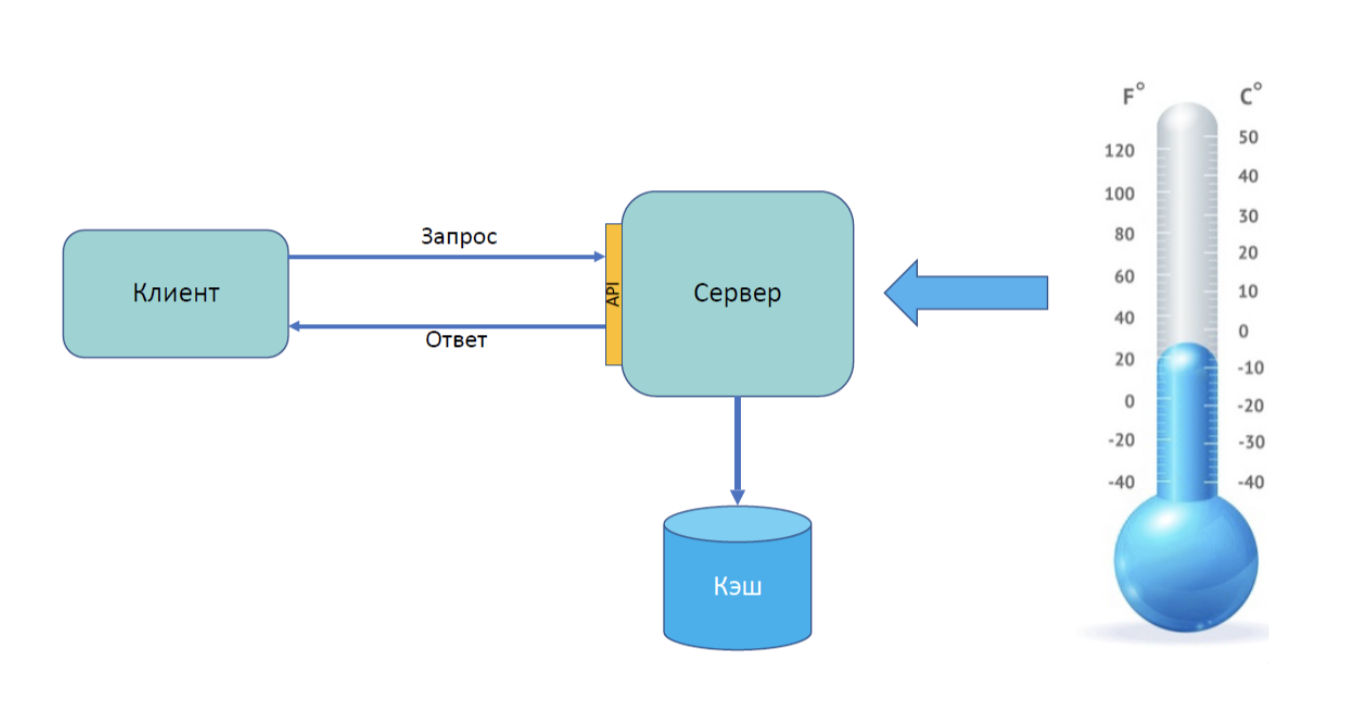
В оригинале этот принцип говорит нам о том, что каждый ответ сервера должен иметь пометку, можно ли его кэшировать.

Представим, что у нас всё так же есть сервис по прогнозу погоды, есть клиент, с которым взаимодействуют. Сам по себе этот сервис погоду не определяет. Погоду определяет метеостанция, с которой он связывается с помощью специальных удалённых вызовов. Что происходит, когда мы используем кэширование?

Например, клиент обратился к серверу с запросом «Хочу узнать погоду». Что делает сервер?

Если мы его только запустили и используем кэширование или если мы не используем кэширование вообще — сервер обратится к метеостанции, а она вернёт ему ответ. Перед тем, как сервер ответит клиенту, он должен сохранить эту информацию в кэше. И только потом вернуть ответ. Для чего?

Когда клиент в следующий раз отправит ровно такой же запрос, сервер сможет не обращаться к метеостанции. Он сможет извлечь прогноз из кэша и вернуть ответ клиенту.

Пример реализации архитектуры с использованием кэширования

Чего мы добились? Мы убрали одну часть взаимодействия между сервером и метеостанцией. Зачем нам это нужно? Это нужно и полезно, если у сервера часто запрашивают одинаковую информацию. Например, кэширование активно используется на новостных сайтах или в соцсетях (на веб-ресурсах, к которым происходит много обращений).

Какие у кэширования плюсы?

* Уменьшение количества сетевых взаимодействий.
* Уменьшение нагрузки на системы (не грузим их дополнительными запросами).

В каких-то случаях одинаковых обращений будет не так много. Тогда кэширование использовать нет смысла.

При этом важно понимать, что кэширование — это совсем не простая штука. Она бывает достаточно сложна и нетривиальна в реализации.

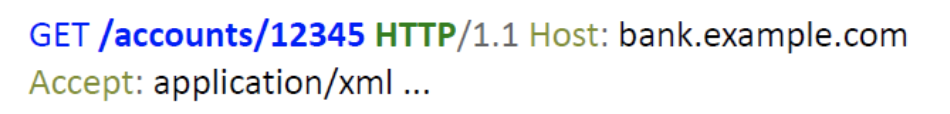
Также мы должны учитывать, что если отдаём какие-то данные, которые сохранили раньше, то важно помнить, что эти данные могли уже устареть.

В каких-то случаях это может быть приемлемо, но в каких-то случаях — абсолютно недопустимо. Соответственно, стоит ли использовать кэширование — всегда нужно обдумывать на конкретном примере.

Принцип 4. Единообразие интерфейса. HATEOAS

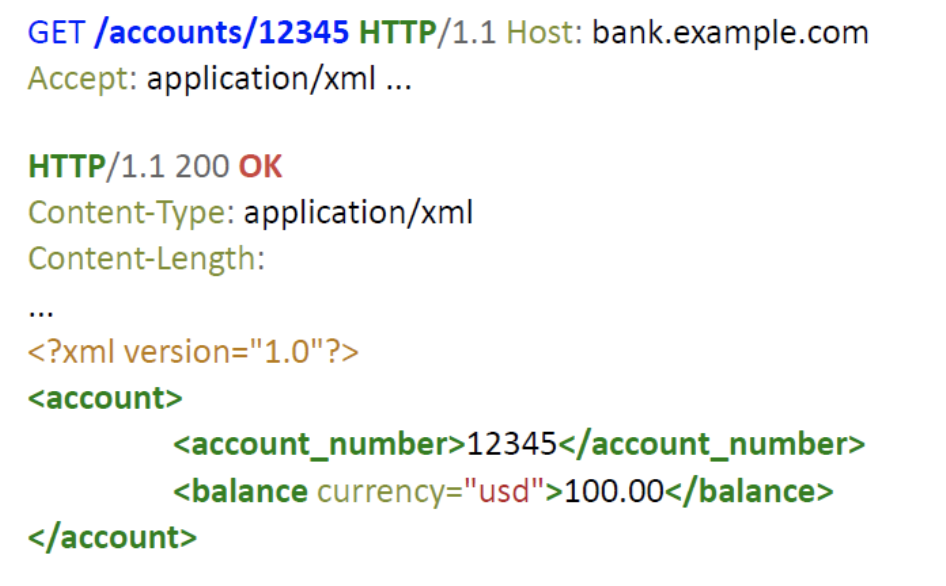
Hypermedia as the Engine of Application State (HATEOAS) — одно из ограничений REST, согласно которому сервер возвращает не только ресурс, но и его связи с другими ресурсами и действия, которые можно с ним совершить.

Рассмотрим пример. Возьмём HTTP-запрос, в котором я хочу получить определенный ресурс:

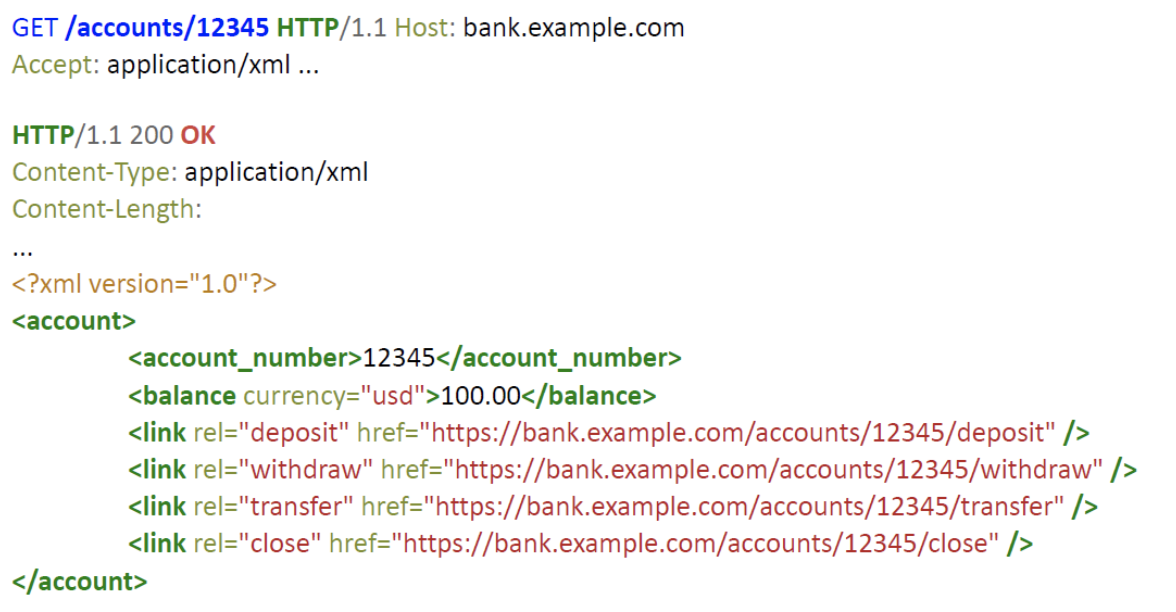
Пример запроса ресурса

Здесь мы используем HTTP-глагол GET, то есть хотим получить ресурс. Обращаемся к некоторому счёту с номером 12345.

Если бы мы не использовали подход HATEOAS, то получили бы примерно такой XML-ответ:

Пример ответа без использования принципа HATEOAS

Что же предлагает HATEOAS? Если бы мы с учётом этого ограничения выполняли бы этот запрос, то в ответе получим не только информацию об этом объекте, но и все те действия, которые мы можем с ним совершить. И, если бы у него были бы какие-то важные связанные объекты, мы получили бы ещё и ссылки на них.

Пример ответа с использованием принципа HATEOAS

Получая такие ответы, клиент самостоятельно понимает, какие конкретные действия он может совершать над этим объектом и какую ещё информацию о связанных объектах он может получить. Мы даём клиентскому приложению намного больше информации и свободы действий. Логика клиента становится более гибкой, но при этом и более сложной.

Главный плюс этого подхода — клиент становится очень гибким в плане изменений на сервере с точки зрения изменения допустимых действий, изменения модели данных и т.д.

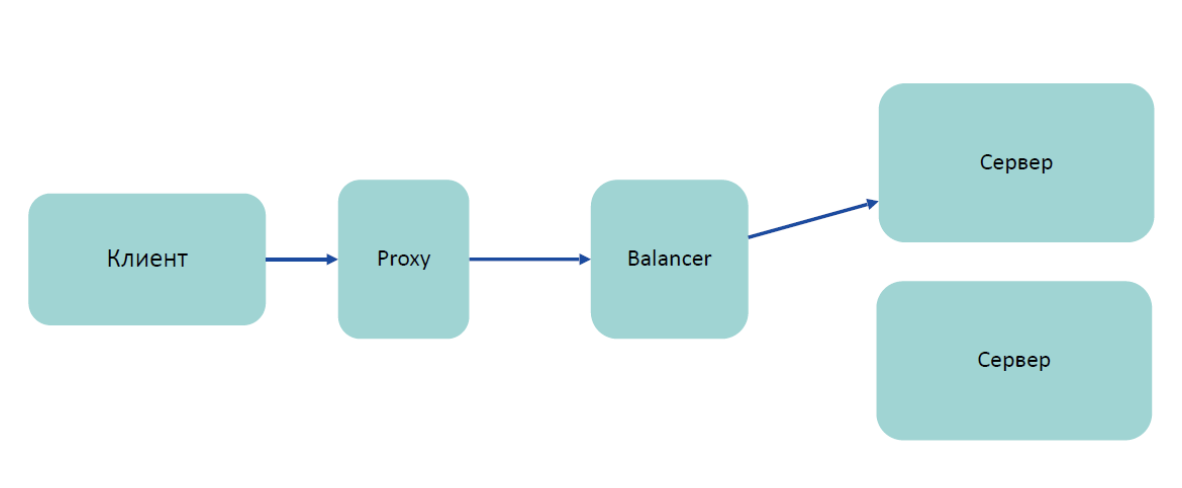
В качестве обратной стороны медали мы получаем сильное усложнение логики, в первую очередь, клиента. Это может потянуть за собой и усложнение логики на сервере, потому что такие ответы нужно правильно формировать. Фактически ответственность за действия, которые совершает клиент, мы передаём на его же сторону. Мы ослабляем контроль валидности совершаемых операций на стороне сервера.

Принцип 5. Layered system (слоистая архитектура)

В предыдущих схемах мы рассматривали сторону клиента и сторону сервера, но не думали, что между ними могут быть посредники.

В реальной жизни между ними могут быть, к примеру, proxy-сервера, роутеры — все, что угодно. И то, по какому пути запрос проходит от клиента до сервера, мы часто не можем знать.

Концепция слоистой архитектуры заключается в том, что ни клиент, ни сервер не должны знать о том, как происходит цепочка вызовов дальше своих прямых соседей.

Модель слоистой архитектуры

Знания в этой схеме об участниках конкретно этой цепочки вызовов должны заканчиваться proxy-сервером слева и сервером справа. О клиенте он уже ничего не знает.

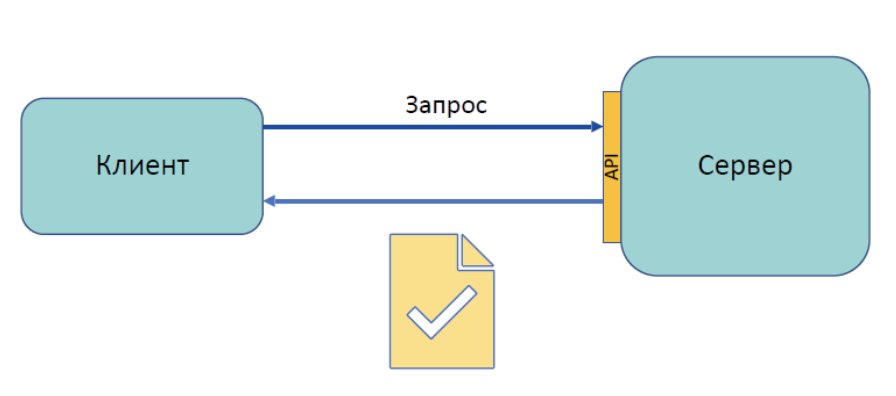
Если изменяется поведение proxy-сервера (роутера или чего-то ещё), это не должно повлечь изменения для клиентского приложения или для сервера. Помещая их в эту цепочку вызовов, мы не должны замечать никакой разницы. Это позволяет нам изменять общую архитектуру без доработок на стороне клиента или сервера.

Минусы:

* Увеличение нагрузки на сеть (больше участников и больше вызовов, чем если бы мы шли один раз от клиента до сервера напрямую).
* Увеличение времени получения ответа (из-за появления дополнительных участников).

Принцип 6. Code on done (код по требованию)

Идея передачи некоторого исполняемого кода (по сути какой-то программы) от сервера клиенту.

Модель архитектуры, реализующей принцип "Код по запросу"

Представьте, что клиент — это, например, обычный браузер. Клиент отправляет некоторый запрос и ждёт ответа — страницу с определённым интерактивом (например, должен появляться фейерверк в том месте, где пользователь кликает кнопкой мышки). Это всё может быть реализовано на стороне клиента.

Либо клиент, запрашивая данную страницу приветствия, получит в ответ от сервера не просто HTML-код для отображения, а ещё программу, которую он сам и исполнит. Получается, что сервер передаёт исходный код клиенту, а тот его выполняет.

Что мы за счёт этого получаем? Отчасти, это схоже с принципом HATEOAS. Мы позволяем клиенту стать гибче. Если мы захотим изменить цвет фейерверка, то нам не нужно вносить изменений на клиенте — мы можем сделать это на сервере, а затем передавать клиенту. Пример такого языка — javascript.

2.2 Как связан Rest с HTTP и JSON?

Давайте рассмотрим наиболее частые заблуждения, которые вы можете встретить относительно концепции REST.

1. Ограничения REST опциональны (необязательны)

С точки зрения создателя этой концепции существует ровно одно необязательное ограничение — код по требованию. Все остальные ограничения должны выполняться. Если одно из них не выполняется — это уже не REST-подход.

2. REST — протокол передачи данных

REST — это не протокол передачи данных. Он не определяет правила о том, как мы должны передавать запросы, какая у них должна быть структура, что мы должны возвращать в ошибках. Единственное, что косвенно можно было бы приписать — это указание на то, что каждый ответ сервера должен содержать информацию о том, можно ли его кэшировать.

Но, в целом, REST — это концепция, парадигма, но не протокол. В отличие от HTTP, который действительно является протоколом.

3. REST — это всегда HTTP

С одной стороны, ни один из архитектурных принципов REST не говорит нам о том, какой транспорт мы должны использовать — HTTP или очереди.

Но при этом в жизни очень часто встречаются люди, для которых REST и HTTP — это аксиома.

Поэтому, если сказать человеку, что REST — это необязательно HTTP, то вас могут посчитать сумасшедшими.

Почему же все считают, что REST — это HTTP? Здесь нужно сделать ремарку, что одним из главных авторов протокола HTTP — это Рэй Филдинг, автор концепции REST. Рэй Филдинг стремился спроектировать HTTP так, чтобы с помощью него концепцию REST было максимально удобно реализовывать.

4. REST — это обязательно JSON

Почему так сложилось? Главная причина в том, что какое-то время назад сервисы вида JSON over HTTP стали противопоставлять SOAP. JSON одновременно стал популярным и стал антагонистом XML, как SOAP подходу. JSON использовался, потому что это не SOAP.

Модель зрелости REST-сервисов

Модель зрелости REST-сервисов Ричардсона
Модель зрелости REST-сервисов Ричардсона

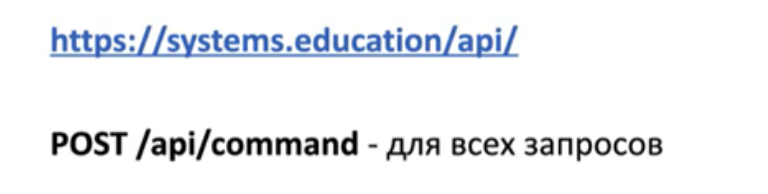
Ричардсон выделил [уровни зрелости](https://en.wikipedia.org/wiki/Richardson_Maturity_Model) REST-сервисов. Выделение происходило исходя из подхода, что REST — это, с точки зрения протокола, всё-таки HTTP. Соответственно, он спроектировал модель, по которой можно понять: насколько сервис REST или не REST.

Уровень 0

В первую очередь, он выделил нулевой уровень. К нему относятся любые сервисы, которые в качестве транспорта используют HTTP и какой-то формат представления данных. Например, когда мы говорим про JSON over HTTP – мы говорим про нулевой уровень.

Если более наглядно «пощупать ручками» с точки зрения использования протокола HTTP, то можно представить, что мы выставляем некоторый API. Мы начинаем с того, что объявляем единый путь для отправки команд и всегда используем один и тот же HTTP-глагол для совершения абсолютно любых действий с любыми объектами. Например, создай вебинар, запиши вебинар, удали вебинар и т.д. То есть мы всегда используем один и тот же URL и всегда используем один и тот же HTTP-метод, обычно POST.

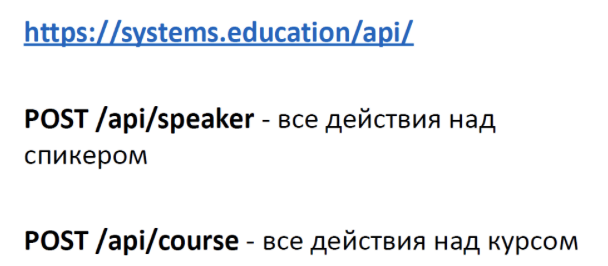
Как один из примеров:

Пример 0-го уровня соответствия REST

Уровень 1

Следующий уровень — первый. Мы уже научились использовать разные ресурсы и делаем это не по одному URL. Но при этом всё ещё игнорируем HTTP-глаголы.

Мы просто разделяем явно наши объекты, как некоторые ресурсы. Например, спикер, курс, вебинар. Но, независимо от того, что мы хотим сделать — удалить, создать, редактировать, мы всё равно используем один и тот же HTTP-глагол POST.

Пример 1-го уровня соответствия REST

Уровень 2

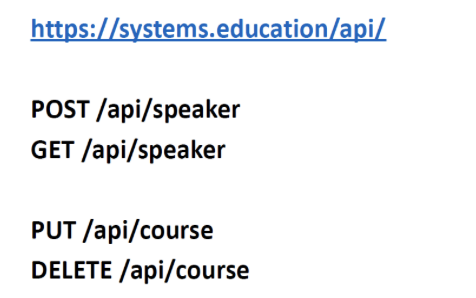
Второй уровень — это когда мы начинаем правильно с точки зрения спецификации HTTP-протокола использовать HTTP-глаголы.

Например, если есть спикер, то, чтобы создать спикера и получить информацию о нём, я использую соответствующий глагол: GET, POST. Когда хочу создать или удалить спикера — я использую глаголы: PUT, DELETE.

По сути, второй уровень зрелости — это то, что чаще всего называют REST.

Надо понимать, что, с точки зрения изначальной концепции, если мы дошли до второго уровня зрелости, то это еще не означает, что мы спроектировали REST-систему/ REST-сервис. Но в очень распространённом понимании соответствие 2-ому уровню часто называют RESTfull сервисом.

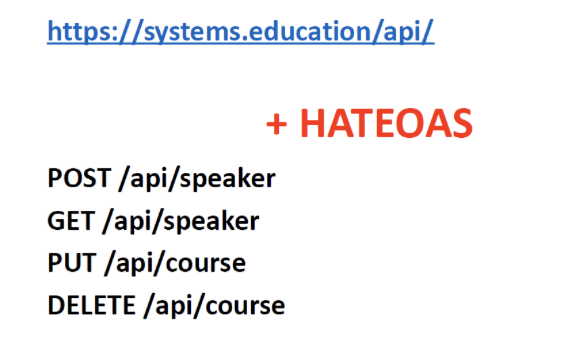
RESTfull-сервис — это такой сервис, который спроектирован с учётом REST-ограничений. Хотя, в целом, правильнее сервис такого уровня зрелости называть HTTP-сервисом или HTTP-API, нежели REST-API.



Пример 2-го уровня соответствия REST

Уровень 3

Третий уровень зрелости — это уровень, в котором мы начинаем использовать концепцию HATEOAS. Когда мы передаём информацию, ресурсы, мы сообщаем потребителям (клиентам) о том, какие ещё действия необходимо совершить ресурсу, а также связи с другими ресурсами.

Пример 3-го уровня соответствия REST

2.3. Почему выбран именно Rest interface?

Заключение

В ходе выполненной работы изучена предметная область ЭКГ-исследований, исследована проблема фильтрации белого шума, спроектирована схема ЭКГ прибора, разработан программный код подсистем, а также проведена демонстрация устройства.

Основным результатом ВКР является прототип прибора ЭКГ, позволяющий производить съемку ритмов сердца в двух точках, для этого решены следующие задачи:

- изучена предметная область ЭКГ,

- подобраны и реализованы алгоритмы и программное обеспечение для съема данных и первичной обработки,

- спроектирована схему ЭКГ прибора,

- разработан программный код подсистем,

- продемонстрирована работа устройства.

К сожалению, не удалось реализовать функции сервера в Orange PI.

Несмотря на это предложенная схема может быть расширена на 12 стандартных отведений за счет использования микросхем, позволяющих одновременно снимать по шесть сигналов и в реальном времени передавать отчеты в Orange PI.