

REST API



ПРЕПОДАВАТЕЛЬ



Юрий Костяной

Java/Kotlin backend-разработчик

- 3+ года опыта в коммерческой разработке
- 2+ года опыта в преподавании
- Проекты по интеграции сторонних платформ, CRM
- Проблемно-ориентированный подход в преподавании



ВАЖНО:

- Камера должна быть включена на протяжении всего занятия.
- Если у Вас возник вопрос в процессе занятия, пожалуйста, поднимите руку и дождитесь, пока преподаватель закончит мысль и спросит Вас, также можно задать вопрос в чате или когда преподаватель скажет, что начался блок вопросов.
- Организационные вопросы по обучению решаются с кураторами, а не на тематических занятиях.
- Вести себя уважительно и этично по отношению к остальным участникам занятия.
- Во время занятия будут интерактивные задания, будьте готовы включить камеру или демонстрацию экрана по просьбе преподавателя.

ПЛАН ЗАНЯТИЯ

1. Повторение
2. Основной блок
3. Вопросы по основному блоку
4. Домашняя работа



TEL-RAN
by Starta Institute

1

ПОВТОРЕНИЕ ИЗУЧЕННОГО

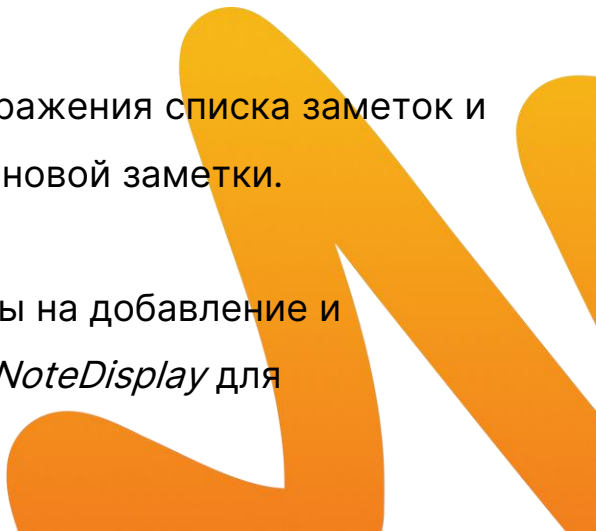
Повторение

Вам представлены три класса для приложения '**Заметки**'. Определите, какой из них является *Model*, какой - *View* и какой - *Controller*.

NoteData: содержит поля *title* и *content*, а также методы для установки и получения этих полей.

NoteDisplay: имеет метод *displayNotes(List<NoteData>)* для отображения списка заметок и метод *showCreateNoteForm()* для отображения формы создания новой заметки.

NoteManager: управляет списком заметок, обрабатывает запросы на добавление и удаление заметок, использует *NoteData* для хранения данных и *NoteDisplay* для отображения информации пользователю.



Повторение

Вам представлены три класса для приложения '**Заметки**'. Определите, какой из них является *Model*, какой - *View* и какой - *Controller*.

Model

NoteData: содержит поля *title* и *content*, а также методы для установки и получения этих полей.

View

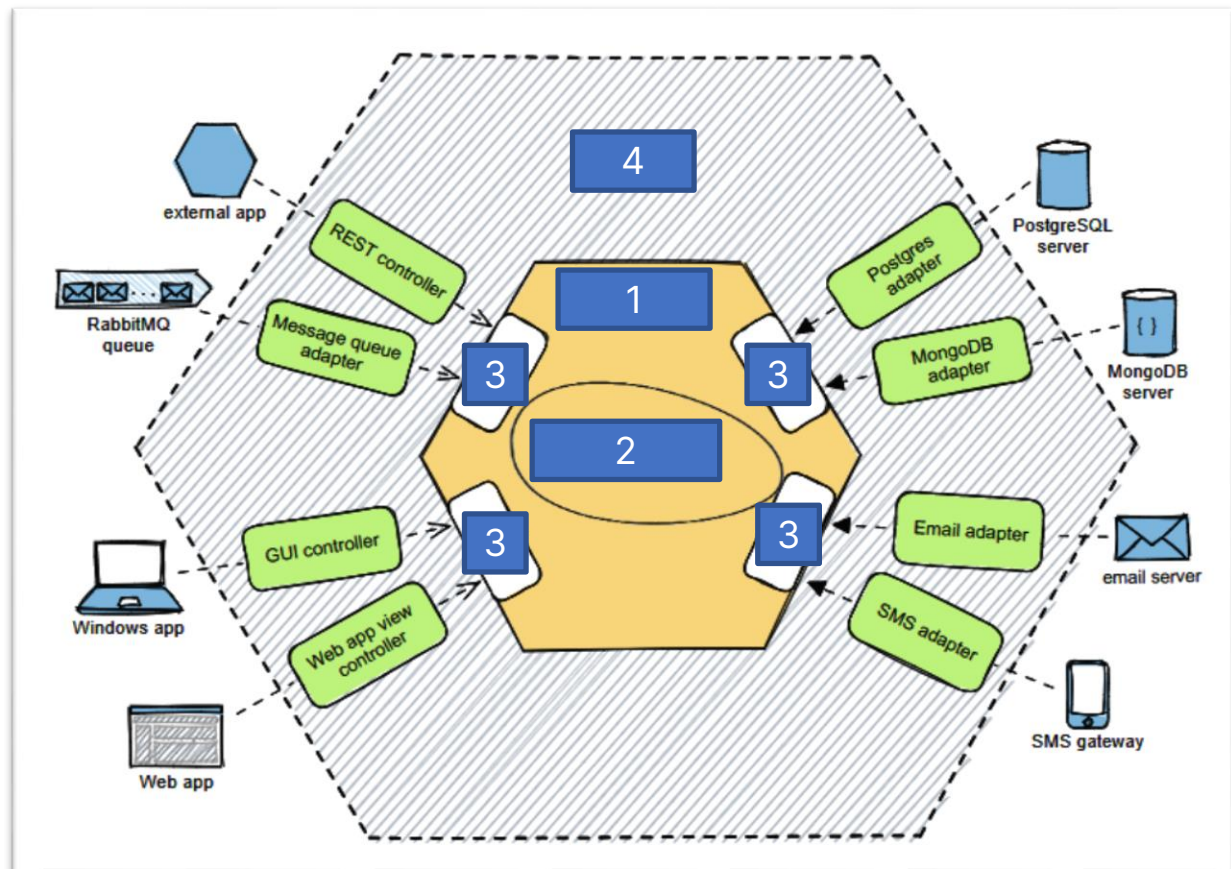
NoteDisplay: имеет метод *displayNotes(List<NoteData>)* для отображения списка заметок и метод *showCreateNoteForm()* для отображения формы создания новой заметки.

Controller

NoteManager: управляет списком заметок, обрабатывает запросы на добавление и удаление заметок, использует *NoteData* для хранения данных и *NoteDisplay* для отображения информации пользователю.

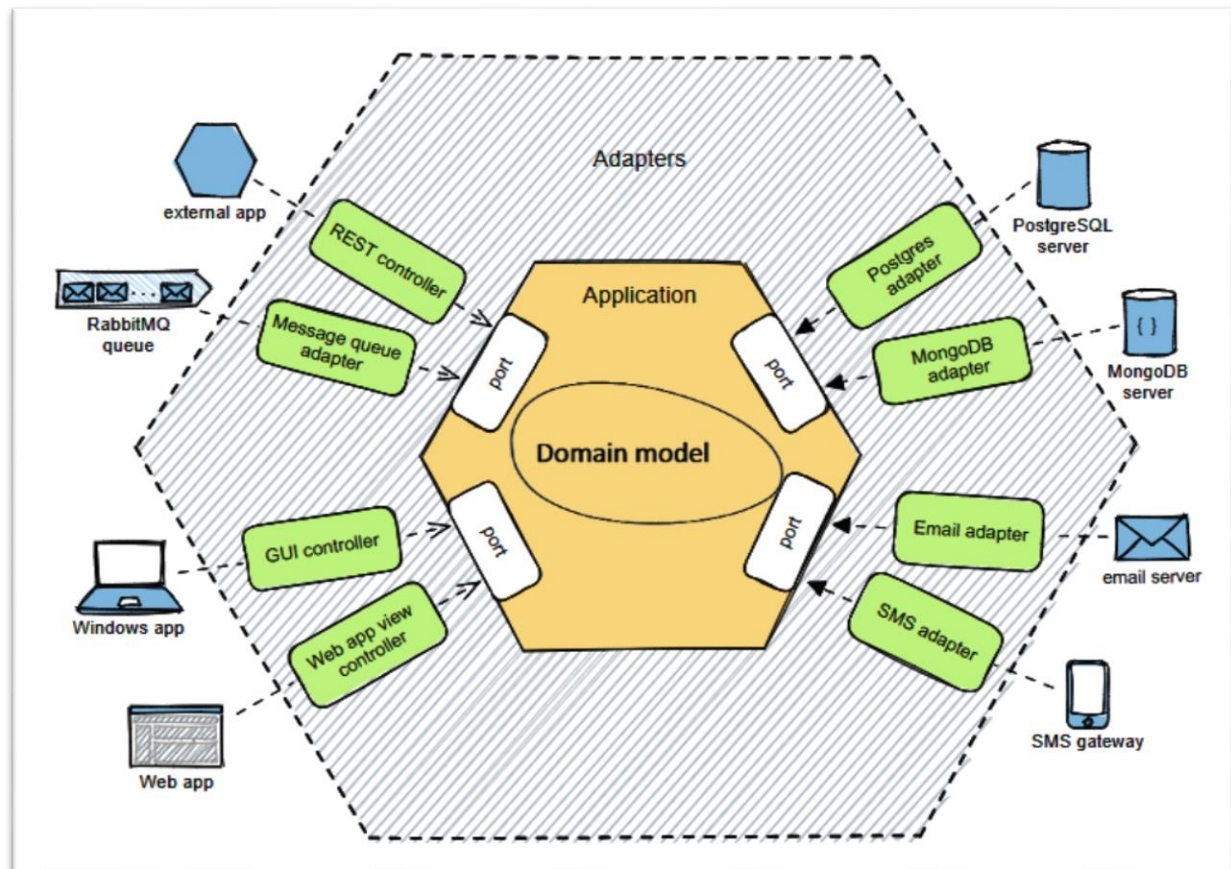
Повторение

Какие части выделяются в
гексагональной архитектуре
приложения?



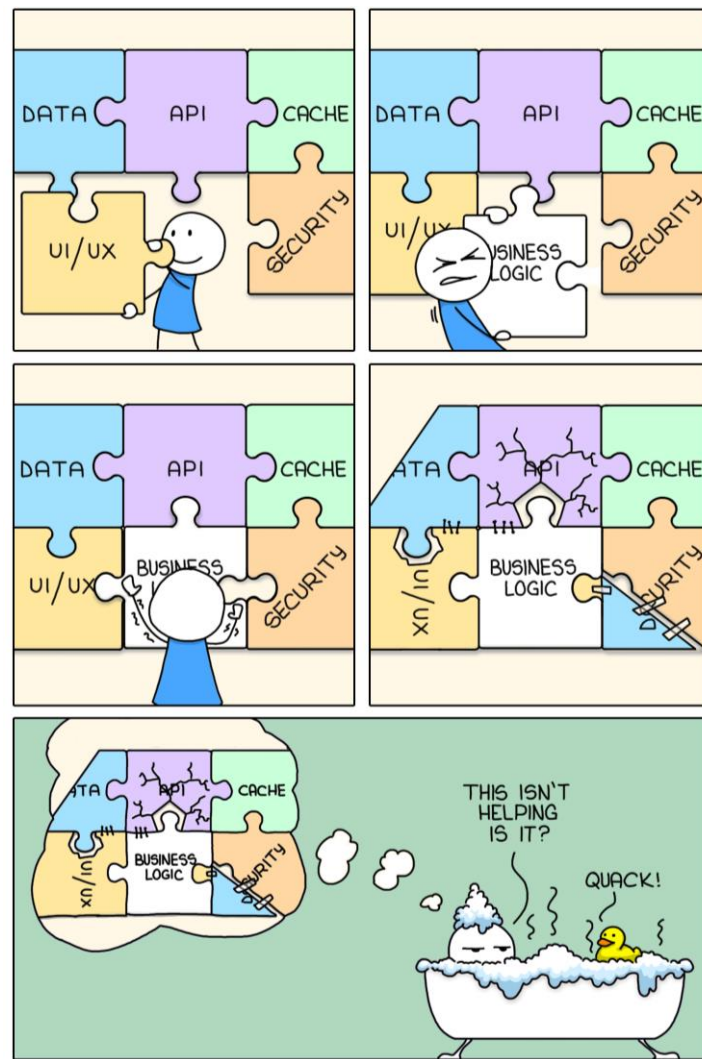
Повторение

Какие части выделяются в
гексагональной архитектуре
приложения?



Повторение

В чём прикол мема?



2

ОСНОВНОЙ БЛОК

Введение

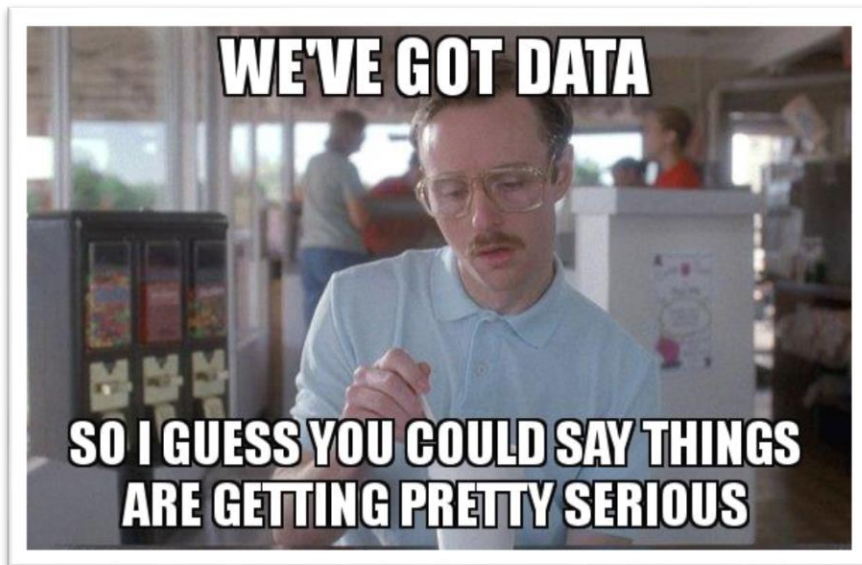
- Rest in Peace



Проблема

Начнём с решения первой задачи, выполняемой приложением – обменом данными с клиентскими приложениями и другими сервисами.

Какие существуют практики организации информационного обмена между клиентом и сервером?



Rest in Peace API

Чтобы клиентское и серверное приложения понимали друг друга им нужен «общий язык для общения». Таким языком является API.

API (application programming interface, программный интерфейс приложения) – спецификации, описывающие способы взаимодействия одной компьютерной программы с другими:

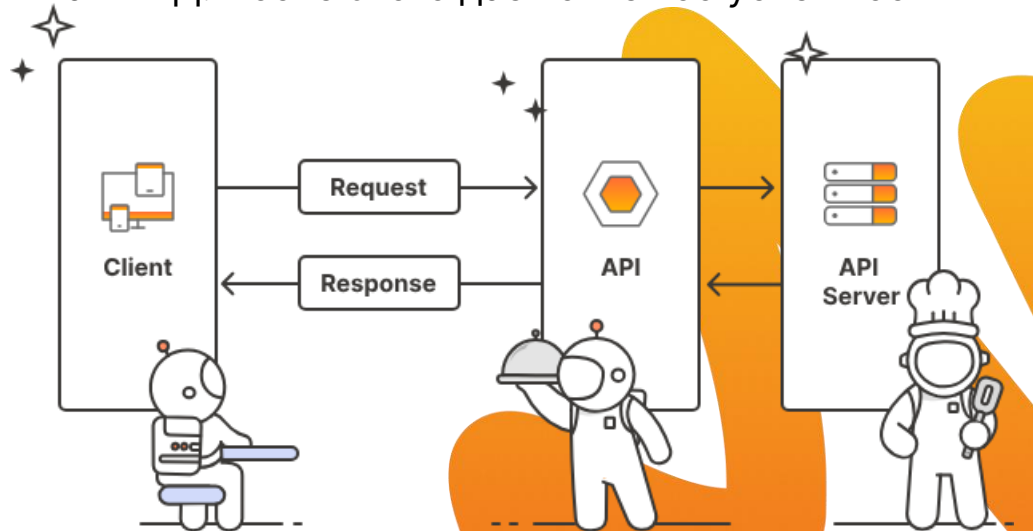
- действия, которые может запрашивать клиентское приложение. Сервер реагирует на эти запросы, выполняя действие и давая ответ
- протоколы и технологии взаимодействия
- состав данных, которыми обмениваются приложения
- способы аутентификации
- расшифровка возможных ошибок
- допустимая последовательность вызовов и др.



Rest in Peace API

Клиентское приложение, используя правила *API*, определённого сервером (обычно оно изложено в документации), направляет запрос к серверу. Сервера, которые обрабатывают запросы пользователей обычно называют **API Server**. Сервер обрабатывает данные и, согласно бизнес-логике, дополняет их запросами от других серверов, реагирует на запрошенное действие, сохраняет результат в БД. После этого даёт ответ об успешности проведённой операции над данными.

API – спецификация, задумка. Для реализации задумки нужна конкретная технология. Наиболее распространёнными технологиями обмена являются *SOAP* и *REST*.



Rest in Peace

SOAP vs REST

SOAP и **REST** – это два механизма обмена данными в Интернете.

Подход *SOAP* отличается высокой степенью структурированности и использует формат данных *XML*.

REST более гибкий и позволяет приложениям обмениваться данными в нескольких форматах (*JSON*, *XML*, *html*, обычный текст, двоичные данные), основной среди которых *JSON*.

Исторически *SOAP* появился раньше, хотя де-факто большая часть приложений сейчас использует *REST* в виду его простоты и краткости передаваемых сообщений. *SOAP* же применяют для приложений, использующих сложную структуру обмениваемых данных.

Подробнее здесь: <https://aws.amazon.com/ru/compare/the-difference-between-soap-rest/>

Rest in Peace **REST API**

REST (REpresentational State Transfer, передача состояния представления или передача «самоописываемого» состояния запрашиваемого ресурса) – это архитектурный стиль для проектирования коммуникационных интерфейсов. *REST* работает только с *http(s)* протоколом в качестве транспорта данных. Системы на основе *REST* не сохраняют состояние, поэтому каждое сообщение обрабатывается независимо от предыдущих.

REST поддерживает шифрование без ущерба для производительности, т.к. этот функционал заложен в протокол *https*.

REST API или **RESTful API** – это API приложения, реализующий принципы *REST*, т.е. обмен *JSON* по протоколу *http(s)* без сохранения состояния.



Rest in Peace Протокол http



http (HyperText Transfer Protocol – протокол передачи гипертекста, т.е. размеченного электронного текста) – протокол прикладного уровня (как *FTP* или *SMTP*), обеспечивающий интерпретацию переданного текстового сообщения как набора частей:

- стартовая строка
- заголовки
- тело

Обмен сообщениями идёт по обыкновенной схеме «*запрос клиента-ответ сервера*». В этой цепочке может быть *прокси* (посредник) – элемент, обеспечивающий работу транспортных служб (пересылка, перенаправление, маршрутизация, балансировка нагрузки и т.д.).

Rest in Peace Протокол http



http не сохраняет своего состояния, но компоненты, использующие *http*, могут самостоятельно осуществлять сохранение информации о состоянии, связанной с последними запросами и ответами (например, «куки» на стороне клиента, «сессии» на стороне сервера). Браузер, посылающий запросы, может отслеживать задержки ответов. Сервер может хранить IP-адреса и заголовки запросов последних клиентов. Однако сам протокол не осведомлён о предыдущих запросах и ответах, в нём не предусмотрена внутренняя поддержка состояния, к нему не предъявляются такие требования.

HTTP is stateless
It does not save the clients request



Rest in Peace Cookie



Cookie (букв. «печенье») – небольшой фрагмент данных, отправленный веб-сервером и хранимый на компьютере пользователя. Веб-клиент (обычно веб-браузер) всякий раз при попытке открыть страницу соответствующего сайта пересылает этот фрагмент данных веб-серверу в составе HTTP-запроса. Применяется для сохранения данных на стороне пользователя, на практике обычно используется для:

- аутентификации пользователя;
- хранения персональных предпочтений и настроек пользователя;
- отслеживания состояния сеанса доступа пользователя;
- хранения сведений статистики о пользователях.

Cookie легко перехватить и подменить в публичных каналах, если не используется шифрование (TLS, SSL)

When you open any website



Rest in Peace

Стартовая строка сообщения http



HTTP

Стартовая строка (Starting line) – определяет тип сообщения. Стартовые строки различаются для запроса и ответа.

Стартовая строка запроса: *Метод URI HTTP/Версия*. Напр., *GET /wiki/HTTP HTTP/1.0*
Метод (Method) – тип запроса, одно слово заглавными буквами (*GET, HEAD, POST, PUT, PATCH, DELETE, OPTIONS, TRACE, CONNECT*). Методы [POST/PUT, GET, PATCH и DELETE](#) соответствуют подходу [CRUD](#) при работе с данными (обычно с базами данных).

URI (Uniform Resource Identifier) – определяет путь к запрашиваемому ресурсу. Основным объектом манипуляции в HTTP является ресурс, на который указывает URI в запросе клиента. Обычно такими ресурсами являются хранящиеся на сервере файлы, но ими могут быть логические объекты или что-то абстрактное (*endpoint*).

Версия (Version) – пара разделённых точкой цифр. Например: 1.0.

Rest in Peace

Стартовая строка сообщения http

HTTP

Стартовая строка ответа: *HTTP/Версия КодСостояния Пояснение*. Напр., *HTTP/1.0 200 OK*

Код состояния (Status Code) – три цифры. По коду состояния определяется дальнейшее содержимое сообщения и поведение клиента;

[Код состояния](#) показывает результат выполнения запроса:

1xx – Информирование о процессе передачи.

2xx – Информирование о случаях успешного принятия и обработки запроса клиента

3xx – Перенаправление. Сообщает клиенту, что для успешного выполнения операции необходимо сделать другой запрос (как правило по другому URI).

4xx – Указание ошибок в запросе клиента.

5xx – Информирование о случаях неудачного выполнения операции по вине сервера.

Пояснение (Reason Phrase) – текстовое короткое пояснение к коду ответа для пользователя. Никак не влияет на сообщение и является необязательным.

Rest in Peace

Код состояния

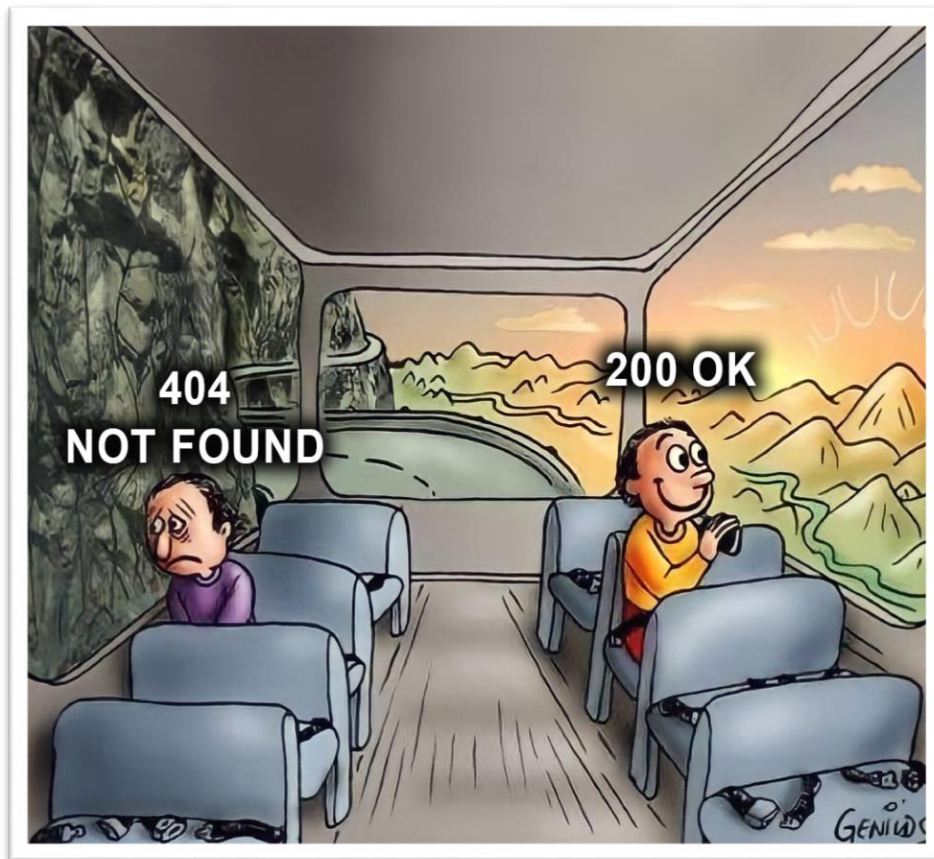
HTTP



Rest in Peace

Код состояния

HTTP



Rest in Peace

Код состояния

HTTP



500

Internal Server Error

Rest in Peace

Код состояния



400 Bad Request



Задание

1 Откройте в браузере консоль (F12 в Google Chrome).

Создайте запрос к странице <https://ru.wikipedia.org/wiki/REST> с помощью браузера (введите URL в адресную строку браузера и нажмите Enter).

Проанализируйте URI и метод запроса, а также код состояния ответа.

2 Установите Postman для своей операционной системы

<https://www.postman.com/downloads/>

Повторите то же действие, создав запрос.



Rest in Peace

Кодировка URL



Кодирование *URL* и просто двоичных данных в последовательность букв, цифр и некоторых специальных знаков латинского алфавита в интернете было связано с ограничением физических устройств на передачу только алфавитно-цифровых символов.

В *URL* такое кодирование обычно применяется для передачи символов в формате *Unicode* (как правило *UTF-8*) в последовательность из двух байт, записанных в шестнадцатичном представлении.

Каждый байт предваряется знаком %. При таком кодировании строка "*корова*" будет иметь вид: `%D0%BA%D0%BE%D1%80%D0%BE%D0%B2%D0%B0`. То есть русской букве *к* будет соответствовать последовательность `%D0%BA` и.т.д. Такое кодирование является общепринятым для путей к файлам или папкам, входящим в *URL*.

В *URL* можно использовать дефис и подчеркивание, но нельзя, например, использовать одинарные или двойные кавычки. Некоторые символы используют для разделения параметров в *URL*, и их кодирование в этом случае будет неправомерным.

Rest in Peace

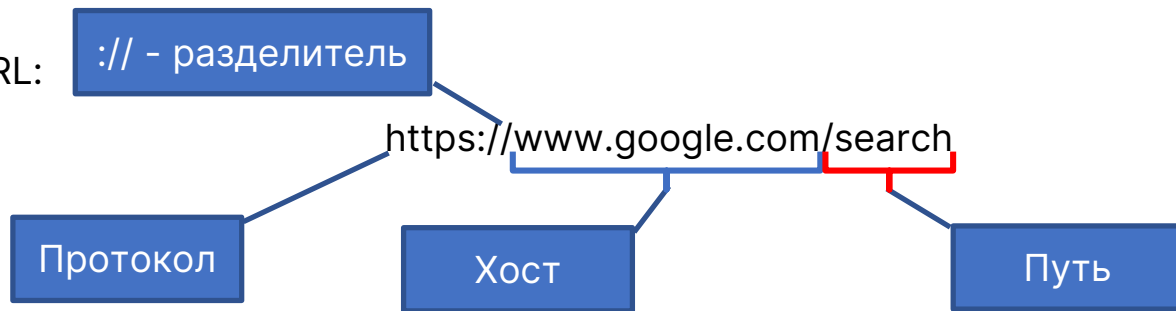
Части URL



Основная часть URL содержит:

- **протокол** (protocol) – протокол передачи данных (http(s), ftp, sftp и т.д.);
- **хост** (host) – доменное имя (протокол DNS), либо IP-адрес;
- **порт** (port) – опциональный числовой параметр, расширяющий возможность обращаться к одному и тому же хосту;
- **путь** (path) – путь к ресурсу. Записывается как набор слов, разделённых знаком /.

Пример URL:



Rest in Peace

Части URL



В составе *URL* можно передавать полезную нагрузку (данные):

- переменные в url. Например, <https://www.ldoceonline.com/dictionary/{word}>, где word – любое слово, искомое в словаре. Скобками {} обозначают переменную часть пути.
- параметры URL. Указываются после ?, представляют пары ключ-значение, разделённые знаком &. Ключ отделяется от значения знаком =. Например, <https://www.google.com/search?q=java&sclient=gws-wiz>
- раздел отображаемой страницы. Указывается после #. Например, https://en.wikipedia.org/wiki/Once_Upon_a_Time_in_America#Cast



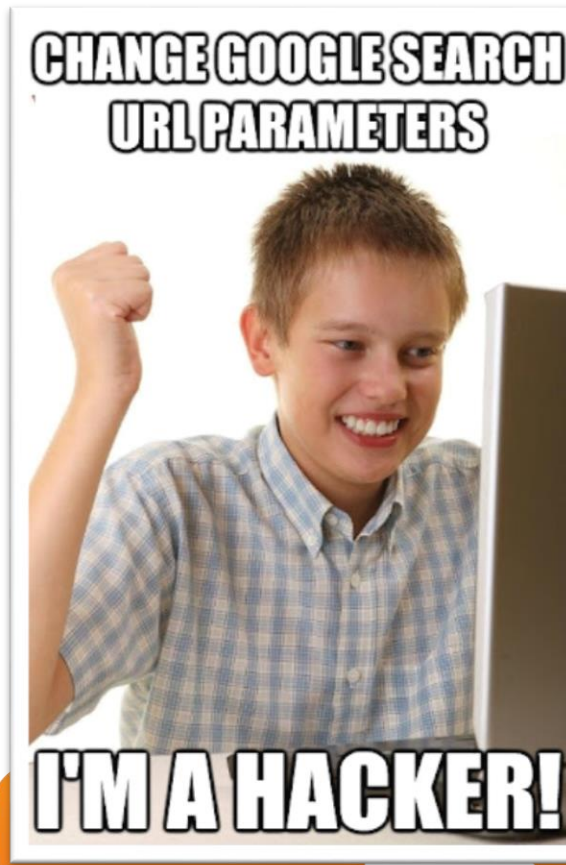
Задание

1 Откройте страницу <https://ru.wikipedia.org/wiki/> в браузере и перейдите в любую русскоязычную статью. Обратите внимание на строку URL в браузере. Скопируйте данную строку и вставьте в текстовый редактор. Почему строки отличаются?

2 Создайте GET-запрос в Postman к странице www.google.com/search. С помощью параметров URL задайте поиск слова *hacker*.

Проанализируйте URI и метод запроса, а также код состояния ответа.

3 Создайте GET-запрос к ресурсу <https://dictionary.cambridge.org/dictionary/english/{word}>. Найдите словарную статью для слова *software*.



Rest in Peace

Заголовки сообщения http

HTTP

Заголовки HTTP (англ. HTTP Headers) – это строки в HTTP-сообщении, содержащие разделённую двоеточием пару имя-значение. Заголовки должны отделяться от тела сообщения хотя бы одной пустой строкой.

Примеры заголовков:

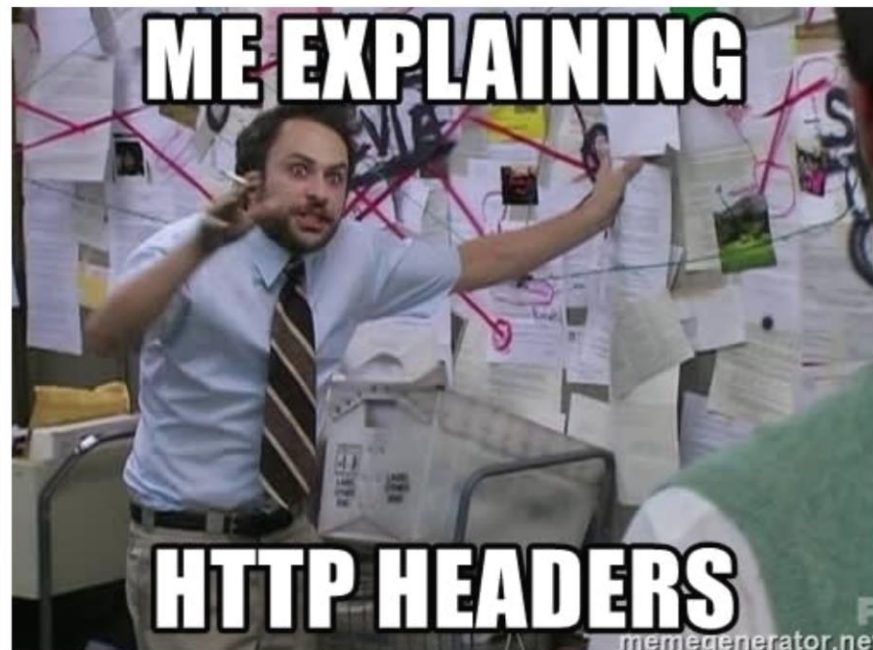
Host: ru.wikipedia.org

Server: Apache/2.2.11 (Win32) PHP/5.3.0

Last-Modified: Sat, 16 Jan 2010 21:16:42 GMT

Content-Type: text/plain; charset=windows-1251

Content-Language: ru



Rest in Peace

Заголовки сообщения http



Все заголовки разделяются на четыре основных группы:

- *General Headers* («Основные заголовки») – могут включаться в любое сообщение клиента и сервера;
- *Request Headers* («Заголовки запроса») – используются только в запросах клиента;
- *Response Headers* («Заголовки ответа») – только для ответов от сервера;
- *Entity Headers* («Заголовки сущности») – сопровождают каждую сущность сообщения.

Именно в таком порядке рекомендуется посылать заголовки получателю.

Все необходимые для функционирования HTTP заголовки описаны в стандартах RFC. Если не хватает существующих, то можно вводить свои. Традиционно к именам таких дополнительных заголовков добавляют префикс «X-» для избежания конфликта имён с возможно существующими. Например, как в заголовках *X-Powered-By* или *X-Cache*.



Заголовки сообщения http

Наиболее распространённые заголовки:

- *Host* – хост сервера;
- *Content-Length* – длина содержимого тела запроса в байтах (*октетах*);
- *User-Agent* – указывает программное обеспечение клиента и его характеристики.
- *Allow* – список поддерживаемых методов всего сервера или конкретного ресурса.

Посылается сервером вместе со статусами *405* и *501*, а также в ответе на метод *OPTIONS*. Пример: *Allow: GET, HEAD, OPTIONS*

- *Content-Type* – формат и способ представления сущности (*mime-type*). *Entity Headers* («Заголовки сущности») – сопровождают каждую сущность сообщения.

Прочие заголовки можно найти [здесь](#).

Rest in Peace

Тип контента

Наиболее распространённые значения заголовка

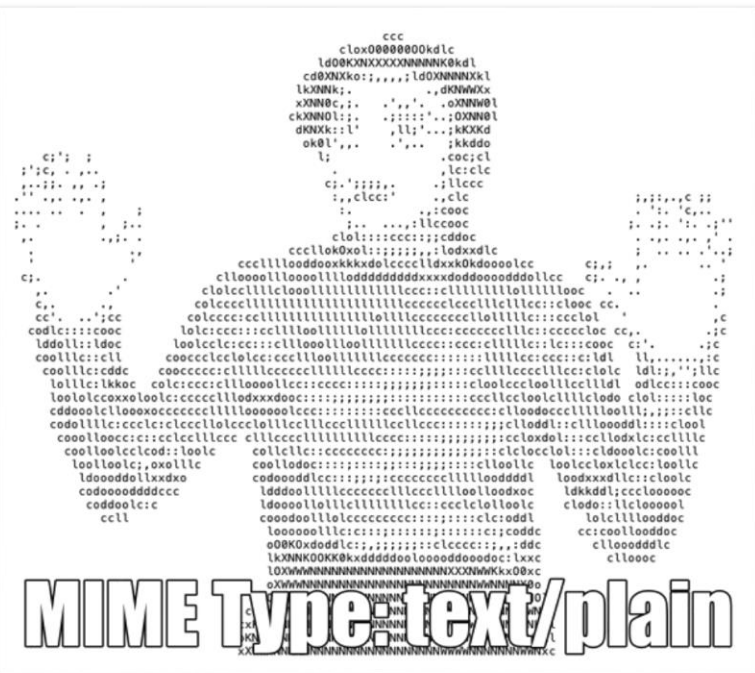
Content-Type (mime):

- text/plain
- text/html
- application/json
- application/xml
- application/x-www-form-urlencoded (тело в кодировке URL)
- multipart/form-data.

Часто к mime-типу дописывают кодировку. Например,

text/html; charset=UTF-8

HTTP



Задание

Повторите ранее созданный запрос в Postman. Проанализируйте заголовки запроса и ответа.



Rest in Peace

Тело сообщения http



Тело HTTP-сообщения (message-body) обычно присутствует в запросах POST, PATCH и PUT и используется для передачи тела объекта, связанного с запросом или ответом.

Правила, устанавливающие допустимость тела в сообщении, отличны для запросов и ответов. Присутствие тела сообщения в запросе отмечается добавлением к заголовкам запроса поля заголовка *Content-Length* или *Transfer-Encoding* (тело закодировано). Тело сообщения может быть добавлено в запрос, только тогда, когда метод запроса допускает тело объекта.

Включается или не включается тело сообщения в сообщение ответа – зависит как от метода запроса, так и от кода состояния ответа:

все ответы на запрос с методом HEAD не должны включать тело сообщения

никакие ответы с кодами состояния *1xx Information*, *204 No Content* и *304 Not Modified* не должны содержать тела сообщения.

Все другие ответы содержат тело сообщения, даже если оно имеет нулевую длину.

Rest in Peace

Тело x-www-form-urlencoded



Изначально *http* задумывался как протокол передачи гипертекста (*html*-страниц), в том числе и данных от *html*-форм. Т.е. пользователь на сайте заполнил форму, нажал кнопку *Отправить* и данные должны выслаться на сервер в виде POST-запроса.

Для передачи тела сообщения было решено использовать уже имеющуюся URL-кодировку.

Данные представляют собой пару ключ-значение (ключ отделён от значение знаком =), пары разделены знаком &. Например,

key1=value1&key2=value2

Такой тип кодирования тела сообщения соответствует Content-Type *application/x-www-form-urlencoded*. Он прост и эффективен, но далеко не все данные можно передать в такой структуре.

Rest in Peace

Текстовые типы тела сообщения

HTTP

В Postman в разделе *raw* можно задать тело одного из следующих типов:

- Обычный текст
- JSON
- XML
- html
- JavaScript



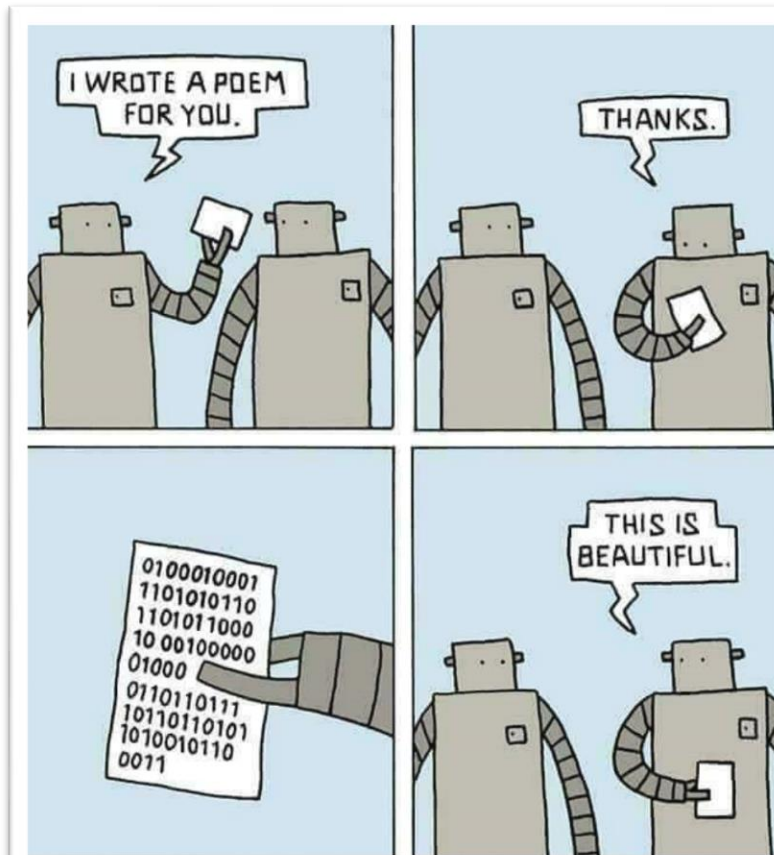
Rest in Peace

Двоичные данные в теле



В тело могут быть помещены двоичные данные (любой файл в виде байт).

В Postman в разделе *binary* достаточно выбрать подходящий файл.



Rest in Peace

Многочастное тело сообщения



Протокол *HTTP* поддерживает передачу нескольких сущностей в пределах одного сообщения. Причём сущности могут передаваться не только в виде одноуровневой последовательности, но и в виде иерархии с вложением элементов друг в друга.

Для обозначения множественного содержимого используются медиатипы *multipart/**.

multipart/mixed – если получателю не известно как работать с типом;

multipart/byteranges – тип для частичного GET-запроса сущности.

multipart/form-data – POST-запрос от html-формы, когда нужно направить текст и приложенные файлы. Параметр *boundary* означает разделитель между различными типами передаваемых сообщений.

[Пример](#) тела запроса.

В Postman тело многочастного сообщения заполняется в разделе *form-data*.

Задание

Перейдите в swagger <https://reqres.in/api-docs/>

Получите данные о запросе авторизации пользователя (*login*). Создайте запрос в Postman и залогиньтесь под пользователем byron.fields@reqres.in. В качестве пароля можно указать любую строку.

Создайте запрос *logout* и выполните его.



3

Домашнее задание

Домашнее задание

1 С помощью Postman создайте GET и POST запросы к одному из известных интернет-ресурсов. Результат выложите в виде Postman-коллекции в Ваш репозиторий.

2 К придуманной ранее архитектуре приложения придумайте REST API для взаимодействия с клиентским приложением. Результат опишите в виде md-файла. Пример описания API:



api_spec.md



ЗАКЛЮЧЕНИЕ

