**Описание требований к интеграции (часть 1). Файловый обмен**

Пост обновлен апр. 10

Описание требований к интеграции систем – стандартная задача для системного или бизнес-аналитика. Но вы можете столкнуться со сложностями, если раньше не решали ее.

Я хочу поделиться своим опытом и рассказать о том, чему нужно уделить внимание, если перед вами стоит задача по написанию требований к интеграции систем. Эта статья будет посвящена файловому обмену.

(О требованиях к интеграции через API вы можете почитать тут: [Часть 2.](https://www.artofba.com/post/systems-integration-api))

**Настройки подключения и расписание запуска**

Как правило написание требований аналитиком начинается с того, как будет осуществляться интеграция. Начните описание со следующих пунктов:

* **Тип взаимодействия**. В нашем случае - файловый обмен.
* **Протокол взаимодействия**. Это может быть обычный открытый FTP протокол либо защищенные протоколы: FTPS (FTP через защищенный криптографический протокол SSL), SFTP (с использованием защищенного протокола SSH).
* **Настройки подключения.** Путь к FTP-серверу, учетные записи для подключения к нему. Также можно указать, чьими силами будет осуществляться настройка доступа: Заказчика или Исполнителя.
* **Условия запуска обмена**. Это могут быть разные события: изменился статус сущности, наступило время, заданное в расписании, пользователь нажал кнопку и т.д.
* **Расписание запуска обмена**. Если условием запуска механизма информационного обмена является расписание, то указывается периодичность запуска (например, ежедневно, еженедельно и т.д.), осуществляющего разбор и обработку, а также время запуска. Также можно указать, должна ли быть возможность отключения запуска, возможность принудительного запуска в любой момент, вне зависимости от настроенного расписания. В некоторых случаях могут быть установлены дополнительные правила для расписания. Например, если на момент заданного времени старта задачи нет данных, то повторять запуск каждые X минут Y раз. Если файл не появился, то записать в лог ошибку.

***Пример***

*Интеграция с внешней системой [Название внешней Системы] должна выполняться с помощью задачи (job) [Название задачи] в соответствии с указанным в [Место настроек расписания, у разных систем свое] расписанием.*

*При запуске задачи Система обращается к папке на FTP-сервере по указанному в настройках задачи пути и принимает XML-файл установленного формата [здесь может быть любой другой формат: CSV, JSON, BIN, структурированный TXT, и др.].*

*Для загрузки данных на стороне [Название принимающей Системы] необходимо создать техническую доменную учетную запись (см. таблицу 1), выполняющую получение данных из FTP-каталога и заполнение [указать что заполняем, таблицу БД, список и т.п.]. Для создаваемой учётной записи (УЗ) должен быть установлен запрет интерактивного входа [зависит от требований].*

*Хранение учетных данных доменной УЗ предполагается с использованием хранилища учетных данных авторизации в службе [Название службы].*

*Для подключения к FTP серверу и проверки корректности загрузки данных необходимо предоставить доступ к FTP серверу для УЗ, перечисленных в таблице 1 (должно быть выполнено силами [Ответственная сторона]).*

*Таблица 1. Перечень УЗ для доступа к FTP - серверу*

|  |  |  |
| --- | --- | --- |
| ***Учетная запись*** | ***ФИО*** | ***Уровень доступа*** |
| *Домен\Учетная запись* | *Иванов И.И.* | *Чтение и запись* |
| *Домен\Учетная запись* | *Техническая УЗ* | *Полный доступ* |

**Где и как забираем данные**

Далее необходимо описать требования к тому месту, откуда будут забираться данные.

* Т.к. мы говорим о файловом обмене, то может возникнуть потребность проанализировать структуру папок, например, на FTP-сервере, а также перемещать или удалять файлы после обработки.
* Если названия папок сформированы по шаблону, то этот шаблон необходимо описать. В названии папок не должны присутствовать пробелы и специальные символы, которые могут быть приняты парсером как разделители.

Кстати, может возникнуть необходимость забирать файлы не только из папок. Они могут приходить по почте, или пользователи могут загружать файлы вручную. В этом случае порядок описания тот же, просто описываются требования не к FTP, а к почтовому ящику или Exchange серверу.

***Пример***

*Необходимо организовать объем доступного дискового пространства на FTP-сервере не менее [Объем] Гб.*

*Путь к FTP-серверу: [Путь]*

*XML-файлы, предназначенные для загрузки в [Название принимающей Системы] должны помещаться в папку «XXX\_Download».*

*После передачи XML-файла, Система должна:*

* *Переместить успешно обработанный файл в папку «XXX\_Loaded».*
* *Если файл обработан неуспешно, то переместить неуспешно обработанный файл в папку «XXX\_Error». Неуспешно обработанными должны считаться файлы:*

o *пустые,*

o *файлы, с некорректной структурой данных,*

*т.е. не соответствующие XSD-схеме, приведенной ниже.*

* *Файл должен быть удален из папки «XXX\_Loaded» через [Период времени] с момента обработки.*
* *Из папки «XXX\_Error» файлы не должны удаляться автоматически.*

Кстати, про удаление данных. Есть смысл предусмотреть хранение исходных данных, перемещение их в архив после обработки или временные таблицы. Т.е. то место, где будут хранится исходные полученные файлы. Не всегда для этого есть возможность, но если есть, то они могут выручить при разборе конфликтных ситуаций и ошибок.

**Предварительные работы до старта интеграции**

Не всегда требуется отдельно описывать, что должно быть сделано до старта интеграции. Но если должны быть проведены некие работы, например, выверка справочника, заполнение реквизитов, настройка, то необходимо прописать состав работ и указать ответственную сторону.

***Пример***

*Перед началом синхронизации необходимо выполнить сопоставление элементов справочника [Название справочника] в системе [Название Системы-источника] с элементами справочника [Название справочника] в системе [Название Системы-приемника].*

*Сопоставление должно быть проведено в системе [Название Системы-приемника] силами [Ответственная сторона].*

**Анализ данных файлов**

Далее приступаем к анализу и описанию содержимого файла.

Заказчик может передать пример файла и, если повезет, спецификацию с описанием структуры. В любом случае, необходимо проверить пример и документ на актуальность и достоверность.

Как проверить на актуальность? Если доступа к данным нет, имеет смысл попросить провести для вас демонстрацию Систем и данных, которые будут передаваться или получаться.

После этого необходимо описать:

* Если название файла формируется по шаблону, то описать сам шаблон. По аналогии с названием папок, в названии файлов не должны присутствовать пробелы и специальные символы, которые могут быть приняты парсером как разделители или стоп-символы.
* Описать атрибуты и их тип, обязательность, допустимые значения, размер полей и т.д.
* Необходимо приложить в спецификацию заполненный пример файла и схему данных. В нашем случае это будет XSD-схема.
* Если требуется создание новых сущностей, также описать их.

Для описания информации об атрибутах можно воспользоваться таблицей 2.

Таблица 2. Описание атрибутов

| **№** | **Атрибут**  **Системы-источника** | **Атрибут**  **Системы -приемника** | **Тэг** | **Тип атрибута** | **Обяза- тельность** | **Описание** |
| --- | --- | --- | --- | --- | --- | --- |
| 1. |  |  |  |  |  |  |

Не забываем о проверках на ошибочные ситуации и поведение нашей системы в них:

* Если название файла не соответствует шаблону названия.
* Если файл не соответствует формату.
* Если нарушена структура и порядок следования атрибутов.
* Если файл пуст.
* Если обязательные атрибуты не заполнены.

Если вам нужно возвращать результат обработки в передающую Систему, то все то же самое необходимо описать для ответного файла.

***Пример***

*Название XML-файла должно формироваться по шаблону:*

*NNN\_[Дата формирования файла в формате YYYY-MM-DD]\_[Время формирования файла в формате HH-MM-SS в 24-часовом формате].*

*Например, NNN\_2017-07-04\_04-02-15*

*Пример XML-файла с данными из [Название Системы-источника]: [Пример XML-файла]*

*Для формирования текста XML-сообщения из [Название Системы-источника] должны использоваться XDTO-пакеты.*

*XSD-схема: [XSD-схема]*

*Таблица 3. Описание атрибутов, пример*

| ***№*** | ***Атрибут***  ***Системы-источника*** | ***Атрибут***  ***Системы -приемника*** | ***Тэг*** | ***Тип атрибута*** | ***Обяза- тельность*** | ***Описание*** |
| --- | --- | --- | --- | --- | --- | --- |
| *1.* | *ID* | *ID* | *ID* | *Строка (255)* | *Да* | *-* |
| *2.* | *Объект* | *Название* | *Title* | *Строка (255)* | *Да* | *-* |

**Алгоритм обработки файлов, правила валидации**

Далее приступаем к описанию алгоритма обработки.

**Для каждой строки интеграции, необходимо описать проверки:**

* Наличие сущности в Системе и что делать, если сущность не найдена. Как минимум, тут есть два варианта: создать сущность или вывести в лог ошибку.
* Проверки на целостность данных, например, есть ли у дочерней сущности родитель, соответствуют ли передаваемые значения внутренней логике Системы?

**Для каждой строки, необходимо описать непосредственно сам алгоритм**

Что должно происходить в системе? Создание, изменение, удаление сущностей? Что происходит с сущностью при этих действиях, что записывается в атрибуты сущности.

Здесь могут быть выявлены дополнительные проверки при выполнении каждого из указанных действий. Например, мы пытаемся удалить сущность, у которой уже есть связанные сущности. В этом случае мы должны проанализировать эти связи и принять решение, можно что-то удалять или нет. Или, например, при корректировке суммы договора получилось отрицательное значение и согласно требованиям, это ошибка. Необходимо прописать и согласовать с Заказчиком правила обработки таких ситуаций.

***Пример***

*При загрузке информации Система должна выполнить для каждой карточки объекта ИС, передаваемой в XML-файле:*

*1. Найти в [Название сущности] элемент, содержащий в поле [Атрибут сущности]*

*значение тега [тег] передаваемой карточки XML-файла;*

*2. Если элемент найден, то обновить поля [Название сущности] следующим образом:*

*2.1. [Описание алгоритма при обновлении сущности]*

*2.2. Записать в лог информацию «Обновлен элемент «[Title]», ID [ID]».*

*3. Если карточка не найдена, то создать новый элемент в [Название сущности], при этом:*

*3.1. [Описание алгоритма при создании сущности]*

*3.2. Записать в лог информацию «Создан элемент «[Title]», ID [ID]».*

**Если есть загрузка из нескольких источников одной и той же сущности**

В этом случае желательно разделить потоки данных на уровне структуры. Т.е. явно указывать в одном из служебных атрибутов сущности, откуда пришла та или иная запись. Даже если нет необходимости явно различать источник данных в пользовательском интерфейсе, такое разделение поможет при разборе ошибок.

**Системные и архитектурные ограничения**

К таким ограничениям может относиться как несоответствие структуре базы данных, так и системные ограничения на получение файлов больших размеров. Ограничения могут быть разной степени тяжести. В одном случае достаточно будет прописать перечень форматов и размер передаваемых файлов. В другом случае могут возникнуть серьезные сложности, если структура передаваемых файлов, и это понятно уже на берегу, не ляжет на структуру данных в Системе-приемнике. Сможете ли вы решить эту ситуацию на уровне алгоритма, зависит от уровня сложности задачи. Но такие ограничения всегда лучше продумать заранее.

**Проверки, излишне требовательные к данным**

Такие проверки могут обернуться проблемами при приемке и тестировании интеграции на промышленных данных.

***Пример проверки***

*Необходимо проверить формат значения в строке «Test»:*

*[ID1 в формате XXX]-[ID2 в формате ZZZ]-[Date в формате DD/MM/YYYY]-[CityName].*

*Параметр CityName в этой строке содержит название города. Если мы попытаемся проанализировать строку с учетом названия города, то столкнемся с практически непредсказуемым содержанием пробелов и тире в названии городов. Это может привести к тому, что часть данных при загрузке данных окажется невалидной из-за этой проверки.*

*Поэтому одно из решений такой ситуации – не анализировать CityName, а проверять формат двух первых ID и даты. Содержание и количество символов в этих параметрах более предсказуемо.*

**Журналирование**

Далее нужно описать, есть ли у Администратора возможность просмотреть информацию по:

* Запланированным заданиям с временем следующего запуска задания.
* Выполняющимся на текущий момент заданиям с временем запуска задания и статусом задания.
* Истории выполненных заданий со временем последнего выполнения, длительностью и статусом выполнения.

Хорошо, когда журнал системных событий по каждому запуску интеграции содержит:

* Дату и время запуска.
* Инициатор запуска (система, либо пользователь).
* Длительность выполнения интеграции.
* Наименование внешней системы.
* Количество обработанных записей,

а также детализацию в разрезе каждого запуска:

o Время начала обработки записи.

o Время окончания обработки записи.

o Результат обработки (успешно, предупреждение, ошибка).

o Сведения об ошибке.

Имеет смысл указать, где Администратор может посмотреть эту информацию.

Необходимо продумать, какие события на каком уровне будут храниться. Например, на уровне Information должны логироваться действия по созданию и обновлению всех сущностей и полей. А в случае возникновения ошибок, сообщения о них должны заноситься в логи на уровне Warning.

Не пишите в требованиях бездумно фразу «записать в логи ошибку». Коллега-разработчик может записать в логи все типы событий (и ошибки, и информацию, и предупреждения) на уровне Error. Это может неприятно закончится при приемке интеграции Заказчиком.

**Разграничение прав**

Не всегда требуется описывать, кто и что видит при отработке сценариев интеграции систем. Чаще всего доступом к запуску и отключению механизма интеграции обладает Администратор.

Но если у вас решение конфликтов при интеграции ложится на плечи выделенного пользователя или роли, то нужно указать, куда у этого пользователя есть доступ и какими правами он обладает.

**Заключение**

Итак, эта статья была посвящена файловому обмену.

О моментах, на которые стоит обратить внимание для других типов взаимодействия между системами, я расскажу в следующих статьях.

**Описание требований к интеграции (часть 2). API**

В этой статье, я и мои коллеги хотим поделиться опытом по интеграции систем используя API.

Как и в [предыдущей статье](https://www.artofba.com/post/%D0%BE%D0%BF%D0%B8%D1%81%D0%B0%D0%BD%D0%B8%D0%B5-%D1%82%D1%80%D0%B5%D0%B1%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B9-%D0%BA-%D0%B8%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B0%D1%86%D0%B8%D0%B8-%D1%87%D0%B0%D1%81%D1%82%D1%8C-1-%D1%84%D0%B0%D0%B9%D0%BB%D0%BE%D0%B2%D1%8B%D0%B9-%D0%BE%D0%B1%D0%BC%D0%B5%D0%BD), стоит сразу сказать, что описание требований к интеграции систем - стандартная задача для системного или бизнес-аналитика. Но вы можете столкнуться со сложностями, если раньше не решали ее.

API - это программный интерфейс системы, который позволяет любой другой заинтересованной системе получить доступ к данным или бизнес-функциям этой системы.

Для начала разберемся в простой ситуации, когда необходимо интегрироваться с уже существующим API, предоставляемым сторонней системой.

Предположим, вам необходимо повысить уровень безопасности вашего приложения путем внедрения двухфакторной аутентификации в системе. На этапе стратегического анализа, учитывая множество факторов, заинтересованные лица приняли решение, что вторым типом аутентификационных данных будет являться числовой код, передаваемый в личном сообщении через Viber.

Стоит отметить, что это одна из многих задач, которую можно решить путем интеграции систем с использованием API, но данный пример считаю простым и достаточным для понимания принципов интеграции систем с помощью API.

Итак, с чего стоит начать решение данной задачи. В первую очередь необходимо разобраться в наборе функциональных возможностей и логике их работы предоставляемой API. Для этого необходимо запросить документацию API, которая может быть представлена в разном виде: свободное описание на сайте компании, [swagger схема](https://swagger.io/tools/swagger-editor/), [Postman коллекция](https://www.postman.com/use-cases/api-first-development/) и т.д., но должна содержать следующую информацию: способ аутентификации и авторизации, набор конечных точек с описанием формата запроса и ответа, описание зависимостей между методами, а также перечень возможных ошибок при работе с API. В нашем случае мы можем найти всю необходимую информацию в публичном доступе [на сайте компании](https://developers.viber.com/docs/api/rest-bot-api/).

Давайте разберемся что же все это значит на конкретном примере.

**Аутентификация** – проверка пользователя на наличие доступа в систему. В нашем случае используется токен - уникальный идентификатор пользователя, сгенерированный в момент создания пользователя и чаще всего неизменный на протяжении всего жизненного цикла пользователя. Изменение токена пользователя приведет к потере доступа к API и необходимости его обновления в нашей системе для возобновления доступа.

**Авторизация** - проверка пользователя на наличие доступа к определённому действию. В нашем случае в зависимости от [настроек доступа](https://developers.viber.com/docs/api/rest-bot-api/#webhooks) в приложении пользователя, которому необходимо отправить сообщение, для нашей системы может быть ограничен доступ к получению персональных данных. Выявление и анализ прав доступа необходимы для валидации требований к нашей системе в будущем.

**Набор конечных точек.** Перечень функциональных возможностей API, реализованный набором [действий разных типов](https://ru.w3docs.com/uchebnik-html/metody-http-zaprosa.html): получить, создать, изменить, удалить и т.д.

**Получить(GET)** – для получения данных об аккаунте нашего пользователя мы можем использовать GET метод [get\_account\_info](https://developers.viber.com/docs/api/rest-bot-api/#get-account-info) .

**Создать(POST)** – для отправки сообщения пользователю мы можем использовать POST метод [send\_message](https://developers.viber.com/docs/api/rest-bot-api/#send-message).

**Изменить(PUT)** – используется для редактирования существующего объекта.

**Удалить(DELETE)** – используется для удаления существующего объекта.

В описании конечной точки, помимо типа действия, указывается путь, по которому эта конечная точка доступна, формат тела запроса и ответа, а также набор входящих и исходящих параметров.

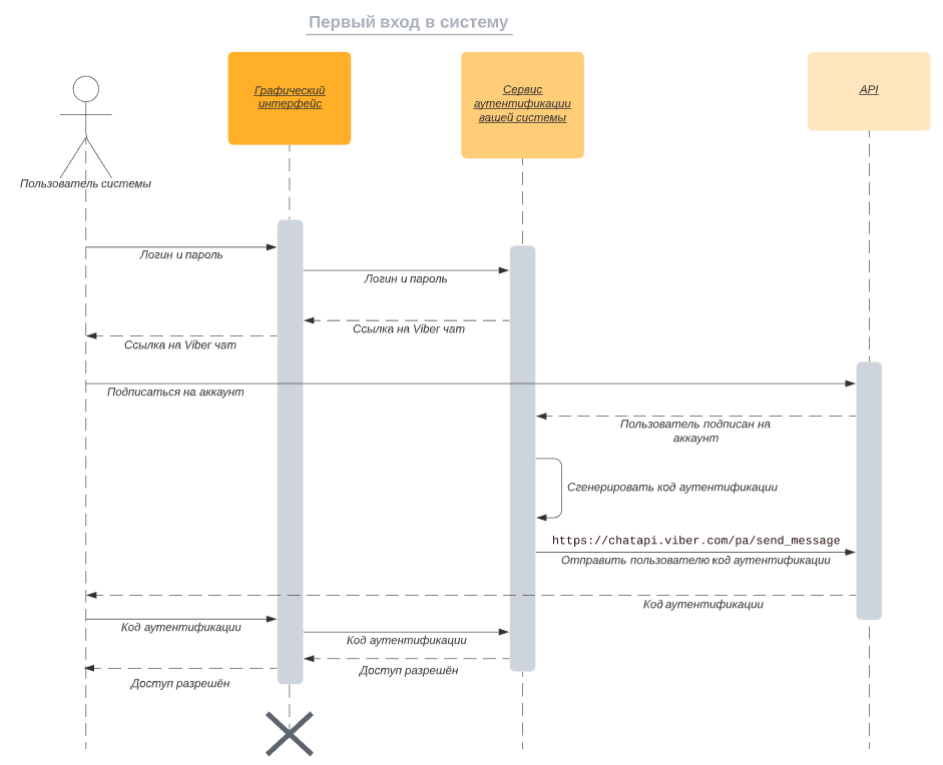
**Путь конечной точки** – это URL, по которому доступен метод.

**Формат тела запроса.** Чаще всего в АПИ используются JSON и XML, подробнее о разнице между ними можно почитать [здесь](https://habr.com/ru/post/31225/). В нашем примере используется JSON.

**Набор входящих и исходящий параметров** – это набор данных необходимый для выполнения метода и возвращаемых в результате работы метода.

**Последовательность вызовов конечных точек.** Так как АПИ - это набор простых действий, при работе с конечными точками необходимо понимать, в какой последовательности их необходимо вызывать. Например, для отправки сообщения пользователю необходимо чтобы пользователь предварительно подписался на наш Viber аккаунт. При нарушении последовательности вызовов конечных точек ответ будет содержать описание и код ошибки. Понимание данных зависимостей необходимо для моделирования процессов взаимодействия вашей системы с API.

После детального изучения логики работы API можно приступать к моделированию процессов взаимодействия систем. Для моделирования можем использовать диаграмму последовательностей UML, на которой необходимо отобразить все взаимодействующие системы или компоненты и смоделировать весь бизнес-процесс в разрезе обмена сообщениями. Принципиально не нужно выделять на диаграмме процессы и подпроцессы, сейчас нет задачи их декомпозировать. Наша задача - смоделировать все возможные потоки данных и определить перечень систем и/или компонентов, необходимых для предоставления необходимых данных или доступа к бизнес-логике. Например, рассмотрим процесс первого входа пользователя в систему:



При моделировании процесса необходимо учитывать, что на выполнения запросов и обработку результатов будет затрачено какое-то время, что может привести к невозможности выполнить последующий запрос, подробнее о race conditions можно почитать [здесь](https://habr.com/ru/post/460339/).

После моделирования всех процессов, необходимо составить таблицу соответствия параметров используемых в интеграции. Т.к. интегрирующиеся системы чаще всего создаются разными командами или организациями для решения различных задач, то одинаковые по своей природе сущности реализовываются разными объектными моделями, учитывающими бизнес-задачу и контекст системы: и в нашей системе, и в Viber конечный пользователь один и тот же, но для реализации его в системе необходим разный набор параметров. Для сопоставления параметров можно использовать следующую таблицу:



**Название параметров** позволит выявить недостающие данные, определить источник заполнения запроса и способ хранения информации, полученной в ответе.

**Формат данных** позволит выявить несоответствия форматов параметров в интегрирующихся системам и связанные с этим возможные проблемы при конвертации форматов либо потере данных из-за ограничения длинны строки.

**Пример данных** необходим для валидации соответствия реальных данных заданному формату. При заполнении необходимо использовать крайние значения данных: самую длинную строку, даты или целочисленные значения, хранящиеся в формате строки и т.д.. В результате анализа данных с продакшена могут появиться дополнительные требования к данным в вашей системе, что в свою очередь приведет к дополнительным трудозатратам.

**Проверка обязательности заполнения параметра** на уровне системы позволит провалидировать возможность заполнения всех обязательных в запросе полей и возможность сохранения данных из ответа с учетом обязательных полей в нашей системе.

**Логика заполнения** позволит явно определить способ обработки данных при наличии какого-либо несоответствия в системах, либо наличии сложной логики заполнения в одной из систем. Например, в вашей системе у одного пользователя может быть более одного телефонного номера, но только один из них помечен как «верифицирован в Viber».

На данном этапе, мы уже понимаем как и какие данные необходимо обрабатывать в процессе интеграции систем для достижения поставленной цели. Но в работе системы могут возникать ошибки, которые нужно научиться корректно обрабатывать. Если вы интегрируетесь с уже готовым API, то в спецификации должен быть определён перечень кодов и описание ошибок возможных при нарушении правил пользования АПИ. В нашем примере такой [список уже есть](https://developers.viber.com/docs/api/rest-bot-api/#error-codes), поэтому мы определяем какие ошибки потенциально могут быть воспроизведены пользовательскими действиями и описываем логику обработки дынных ошибок нашей системой. Также ошибки могут воспроизводиться из-за большого количества запросов, поэтому необходимо составить прогноз количества запросов в единицу времени для каждой конечной точки на весь жизненный цикл вашей системы.

После определения зачем и как будет работать будущая интеграция необходимо задуматься о том, как после реализации мы будем выполнять отладку, оценивать и анализировать работу интеграции. Для решения этих задач можно использовать журналирование вызовов конечных точек, с сохранением URL, тела запроса, тела ответа и даты и времени вызова.

Описанные выше артефакты считаю достаточными для начала создания технического дизайна интеграции техническими специалистами вашей команды.

В следующей части статьи мы рассмотрим задачу проектирования API.