



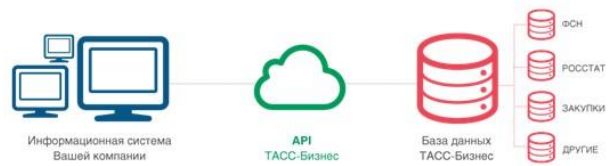
Web service

API

REST

API

Application Programming Interface



Это программный интерфейс, который позволяет двум приложениям взаимодействовать друг с другом без какого-либо вмешательства пользователя. API предоставляют продукт или услугу для связи с другими продуктами и услугами, не зная, как они реализованы.



Аббревиатура **API** расшифровывается как «**Application Programming Interface**» (интерфейс программирования приложений, программный интерфейс приложения).

Чтобы понять, как и каким образом API применяется в разработке и бизнесе, сначала нужно разобраться, как устроена «всемирная паутина».

Всемирная паутина и удалённые серверы

WWW можно представить как огромную сеть связанных серверов, на которых и хранится каждая страница. Обычный ноутбук можно превратить в сервер, способный обслуживать целый сайт в сети, а локальные серверы разработчики используют для создания сайтов перед тем, как открыть их для широкого круга пользователей.

При введении в адресную строку браузера www.facebook.com на удалённый сервер Facebook отправляется соответствующий запрос. Как только браузер получает ответ, то интерпретирует код и отображает страницу.

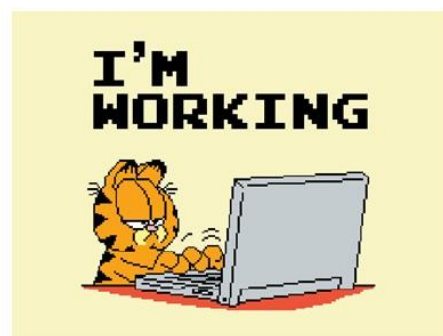
Каждый раз, когда пользователь посещает какую-либо страницу в сети, он взаимодействует с API удалённого сервера. API — это составляющая часть сервера, которая получает запросы и отправляет ответы.

API содержит в себе некие «мостики», позволяющие программе А получить доступ к данным из программы Б или к некоторым ее возможностям. Таким образом, программисты могут расширять функциональность своего продукта и связывать его с чужими разработками.

Все это с разрешения создателей программы А и с соблюдением всех мер безопасности, чтобы разработчики, желающие использовать API, не смогли получить доступ к конфиденциальной информации.

Главный принцип работы API.

Почему его называют интерфейсом



Инкапсуляция сокрытие части функций ради упрощения работы в целом и минимизации участков программного обеспечения, где один из разработчиков мог бы допустить ошибку.

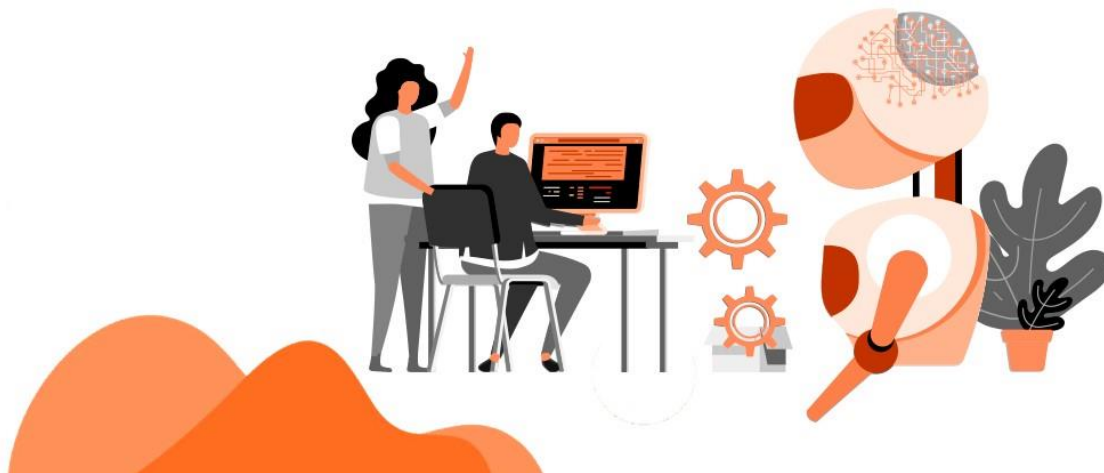
Главный принцип работы API. Почему его называют интерфейсом

Простыми словами, интерфейс – это «прослойка» между приложением А и приложением Б. В ней происходят процессы, которые позволяют двум программам обмениваться информацией и выполнять функции, связанные с обеими сторонами, скрывая «внутреннее строение» программ. Знакомо? Только что таким же образом мы описали API.

Такой подход позволяет наладить взаимодействие между несколькими утилитами, не задумываясь о том, как они устроены, какая программная логика ими движет и каким образом обрабатываются передаваемые данные. Интерфейсы упрощают работу как для простых пользователей, так и для программистов. Первым не приходится задумываться о том, что стоит за привычными функциями в их гаджетах, а разработчикам не нужно изучать код других программистов, чтобы подключить чужой продукт к своему. Это называется инкапсуляцией. Сокрытием части функций ради упрощения работы в целом и минимизации участков программного обеспечения, где один из разработчиков мог бы допустить ошибку.

Зачем нужен API?

Почему разработчики используют API?



Зачем нужен API?

Теперь нам знакомы принципы работы API и задачи, которые они помогают решить. Но они хороши не только этим. Программные интерфейсы используются еще по двум немаловажным причинам.

Во-первых, такой подход позволяет делать программы надежнее. Инкапсуляция в целом заметно упрощает жизнь разработчиков. Отдельные компоненты приложений становятся абстракциями. Создателям нового ПО не приходится лезть в логику низкоуровневых функций и разбираться в их реализации. Так заметно повышается безопасность выполняемых задач, что особенно заметно на уровне таких масштабных программных продуктов, как операционные системы. Программы постоянно выполняют сотни внутренних задач, при этом они проходят незаметно для пользователя и не могут навредить друг другу.

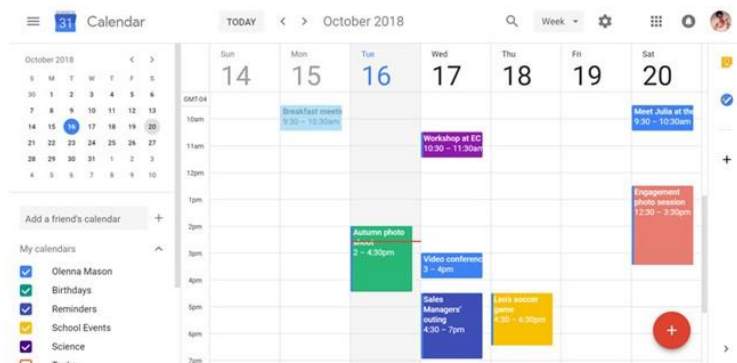
Во-вторых, на API можно заработать. Например, сервисы, предоставляющие информацию с метеовышек, берут плату за каждый запрос актуальной погоды, если их API используется в сторонних приложениях. Аналогичные условия могут предлагать и другие компании, предоставляющие услуги. Будь то навигация, конвертация файлов в другие форматы и прочие возможности, реализуемые через API.

Почему разработчики используют API?

Есть как минимум еще 4 причины, объясняющие интерес программистов к API:

1. API упрощает и ускоряет создание новых продуктов. Разработчикам не приходится каждый раз изобретать велосипед. Можно взять API нейронной сети TensorFlow, к примеру, и внедрить в свое программное обеспечение, а не создавать собственную систему машинного обучения.
2. программный интерфейс увеличивает безопасность разработки. С помощью него можно вынести ряд функций в отдельное приложение, сделав невозможным их некорректное использование. От человеческого фактора это тоже спасает.
3. API упрощает настройку связей между разными сервисами и программами. Интерфейс нивелирует необходимость в тесном сотрудничестве создателей различных приложений. Разработчики могут внедрять поддержку сторонних сервисов, вообще не контактируя с их создателями.
4. Наличие готовых интерфейсов позволяет сэкономить не только время и силы программистов, но и финансы, с которыми часто связано создание новых программных решений.

Google Календарь



Погодное приложение



Примеры API

Работа API представляет собой передачу данных по определенному запросу со стороны клиента или другого приложения. Допустим, нужно выудить информацию с существующего сайта и передать ее в программу.

В браузере будет дан запрос и ожидается ответ в виде HTML-страницы. Если же используется API в стороннем приложении, то ему может быть достаточно фрагмента данных в формате JSON. Более точное техническое описание работы любого из существующих API доступно только их создателям.

На стороне пользователя такая реализация интерфейса будет выглядеть как банальная возможность выполнить действие, связанное с программой А в программе Б. То есть убрать лишний переход в стороннюю программу.

Google Календарь

Те, кто использовал приложения-календари для iOS или Android, знают, что данные в них можно синхронизировать, подключив один из популярных сервисов: Apple iCal или Google Calendar. Обе компании предлагают разработчикам API, позволяющие подключить свой календарь напрямую к сторонним приложениям. Благодаря подобной интеграции люди могут использовать несколько разных программ со схожей функциональностью и иметь на руках актуальную информацию о всех своих делах.

API позволяют создавать новые события и напоминания, удалять уже существующие, редактировать их и т.п.

Погодное приложение

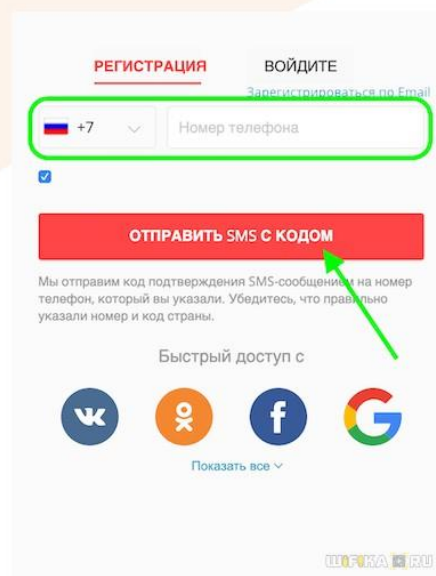
Существующие погодные приложения (встроенные в операционную систему или сторонние из App Store или Google Play) получают информацию о погоде из сторонних источников.

Есть сервисы, взаимодействующие напрямую с метеостанциями и обладающие информацией о текущей погоде. Разработчики приложений для мобильных устройств эту информацию покупают.

А чтобы весь процесс упростить, сервисы, сотрудничающие с метеостанциями, разработали соответствующие API. В них содержится набор функций, помогающий делать запросы о погоде в конкретных местах. Эти запросы через посредника (приложение) отправляются на «метеостанцию», а их результат возвращается пользователю тем же путем.



Сервис по заказу авиабилетов



Кнопки авторизации

Сервис по заказу авиабилетов

Здесь аналогичная ситуация. Помимо сайтов и приложений, принадлежащих авиакомпаниям, есть так называемые агрегаторы. У нас популярен Aviasales, но есть и другие.

Такие сервисы собирают информацию о стоимости авиабилетов в разных авиакомпаниях и отображают ее в едином окне. Чтобы добыть эту информацию, разработчики используют функции сервисов авиакомпаний, которые помогают в реальном времени обновлять информацию о направлениях и стоимости билетов.

Кнопки авторизации

Наверняка вы видели на различных сайтах кнопки, позволяющие зарегистрироваться с помощью уже существующих аккаунтов на популярных площадках. Сейчас такие есть у Google, Facebook, Apple, Twitter, ВКонтакте и т.д. Набор доступных опций на конкретном ресурсе полностью зависит от его хозяев. Это тоже делается через API. Условная Apple создала набор защищенных функций, который можно с минимальными затратами подключить к своему проекту и предоставить пользователям доступ к удобному и безопасному способу авторизации.

При этом жизнь пользователей становится проще, а у владельцев ресурса остается доступ к почтовым адресам и другим персональным данным для взаимодействия с вновь зарегистрировавшимся человеком.

Навигация на сайтах и в приложениях

Тут почти как с погодой. Есть несколько крупных корпораций, предлагающих картографические данные. Те же Apple, Google, Yandex и парочка других. Некоторые из этих компаний разработали API, позволяющие подключить собственный картографический сервис к другим площадкам. Иногда они используются во внутренних продуктах. Яндекс.Транспорт построен на базе Яндекс.Карт, к примеру. Иногда API используются крупными партнерами. Uber использует для навигации сервис компании Google.

Web-сервис (служба) – программа, которая организывает взаимодействие между сайтами. Информация с одного портала передается на другой.



По сути, веб-сервисы — это реализация абсолютно четких интерфейсов обмена данными между различными приложениями, которые написаны не только на разных языках, но и распределены на разных узлах сети.

Именно с появлением веб-сервисов развилась идея SOA — сервис-ориентированной архитектуры веб-приложений (Service Oriented Architecture).

Информация в интернете разнородна. Сайты управляются разными системами. используются разные протоколы передачи и шифрования. Веб-сервисы упрощают обмен информацией между разными площадками.

Можно определить 3 инстанции, которые взаимодействуют между собой: каталог, исполнитель и заказчик. После создания сервиса, исполнитель регистрирует его в каталоге, а там сервис находит заказчик.

Механизм обмена данными формируется в описании Web Services Description. Это спецификация, охватывающая форматы пересылки, типы контента, транспортные протоколы, которые применяются в процессе обмена сведениями между заказчиком и транспортировщиком услуг.

Задачи веб-сервисов



- Интеграция процессов идет сразу, без участия людей
- Если в компании используются корпоративные программы, то веб-сервис поможет настроить их совместную работу

Задачи веб-сервисов

Веб-сервисы могут использоваться во многих сферах.

B2B-транзакции **B2B** (от англ. business-to-business, «бизнес для бизнеса») — это продажи, в которых и заказчиками, и продавцами выступают юридические лица.

Интеграция процессов идет сразу, без участия людей. Например, пополнение каталога интернет-магазина новыми товарами. Их привозят на склад, и кладовщик отмечает в базе данных приход. Автоматически информация передается в интернет-магазин. И покупатель вместо пометки “Нет на складе” на карточке товара видит его количество.

Интеграция сервисов предприятий

Если в компании используются корпоративные программы, то веб-сервис поможет настроить их совместную работу.

Создание системы клиент-сервер

Сервисы используются, чтобы настроить работу клиента и сервера. Это дает преимущества:

- можно продавать не само программное обеспечение, а делать платным доступ к веб-сервису;
- легче решать проблемы с использованием стороннего ПО;
- проще организовывать доступ к контенту и материалам сервера.

Веб-сервис — это приложение, которое упрощает техническую настройку взаимодействия ресурсов.

Веб-сервис Web-API	API
Все веб-сервисы являются API.	Все API не являются веб-сервисами.
Он поддерживает XML.	Ответы форматируются с использованием MediaTypeFormatter Web API в XML, JSON или любой другой заданный формат.
Он может использоваться любым клиентом, который понимает XML.	Может использоваться клиентом, который понимает JSON или XML.
Веб-сервис использует три стиля: REST, SOAP и XML-RPC для связи.	API можно использовать для любого стиля общения.
Он обеспечивает поддержку только для протокола HTTP.	Он обеспечивает поддержку протокола HTTP / s: заголовки запроса / ответа URL и т. Д.



Разница между API и веб-сервисами
Вот важные различия между веб-сервисами и API.

Веб-сервис - это просто API в одежде HTTP.

Веб-сервис	API
------------	-----

Все веб-сервисы являются API.	Все API не являются веб-сервисами.
Он поддерживает XML.	Ответы форматируются с использованием MediaTypeFormatter Web API в XML, JSON или любой другой заданный формат.
Вам нужен протокол SOAP для отправки или получения данных по сети. Поэтому он не имеет легкой архитектуры.	API имеет легковесную архитектуру.
Он может использоваться любым клиентом, который понимает XML.	Может использоваться клиентом, который понимает JSON или XML.
Веб-сервис использует три стиля: REST, SOAP и XML-RPC для связи.	API можно использовать для любого стиля общения.
Он обеспечивает поддержку только для протокола HTTP.	Он обеспечивает поддержку протокола HTTP / s: заголовки запроса / ответа URL и Д.

Преимущества API Сервисов

Вот плюсы / преимущества использования API:

- API поддерживает традиционные действия CRUD (Create Read Update Delete), так как он работает с HTTP-глаголами GET, PUT, POST и DELETE.
- API помогает вам выставлять данные сервиса браузеру
- Он основан на HTTP, который легко определить, предоставить полный REST-способ.

КЛЮЧЕВАЯ РАЗНИЦА

- Веб-сервис — это набор протоколов и стандартов с открытым исходным кодом, используемых для обмена данными между системами или приложениями, а API — это программный интерфейс, который позволяет двум приложениям взаимодействовать друг с другом без какого-либо участия пользователя.
- Веб-сервис используется для REST, SOAP и XML-RPC для связи, в то время как API используется для любого стиля связи.
- Веб-сервис поддерживает только протокол HTTP, тогда как API поддерживает протокол HTTP / HTTPS.
- Веб-сервис поддерживает XML, а API поддерживает XML и JSON.
- Все веб-сервисы являются API-интерфейсами, но все API-интерфейсы не являются веб-сервисами.

СПОСОБЫ РЕАЛИЗАЦИИ ВЕБ-СЕРВИСОВ



XML-RPC (XML Remote Procedure Call)

SOAP (Simple Object Access Protocol)

JSON-RPC (JSON Remote Procedure Call)

REST (Representational State Transfer)

Специализированные протоколы для конкретного вида задач,
такие как **GraphQL**.

Самые известные способы реализации веб-сервисов:

- **XML-RPC** (XML Remote Procedure Call) — протокол удаленного вызова процедур с использованием XML. Прародитель SOAP. Предельно прост в реализации.
- **SOAP** (Simple Object Access Protocol) — стандартный протокол по версии W3C. Четко структурирован и задокументирован. Данный протокол позволяет отправлять и получать сообщения, используя XML в качестве основы.
- **JSON-RPC** (JSON Remote Procedure Call) — более современный аналог XML-RPC. Основное отличие — данные передаются в формате JSON.
- **REST** (Representational State Transfer) — архитектурный стиль взаимодействия компьютерных систем в сети основанный на методах протокола HTTP.
- Специализированные протоколы для конкретного вида задач, такие как GraphQL.
- Менее распространенный, но более эффективный gRPC, передающий данные в бинарном виде и использующий HTTP/2 в качестве транспорта.
- **WSDL**. Это язык, который служит для описания сторонних интерфейсов на базе XML.