API

Application Programming Interface или программный интерфейс приложения – это совокупность способов, протоколов, инструментов, с помощью которых различные программы обмениваются своими возможностями, данными, выполняют разные функции.

API это то, что позволяет настроить как разные компоненты программы должны эффективно взаимодействовать. Проще говоря, API объявляет интерфейс для взаимодействия с его логикой без необходимости знать, что происходит внутри.

API создан для удобства. Когда пользователь работает с планшетом или другим девайсом, ему не приходится вникать, как компьютер обрабатывает информацию: он просто нажимает на иконки в интерфейсе. Аналогичная ситуация с API. Благодаря программному интерфейсу разработчик может подключить свой продукт к другим системам для хранения файлов, отрисовки графики, воспроизведения видео или аудио. При этом ему не приходится писать собственный код или разбираться, как именно работает ОС. Такой алгоритм упрощает и ускоряет процессы.

Как использовать API.

Основная функция API – построение эффективной коммуникации между программами. Для разных целей интерфейс выполняет разные задачи.

В контексте интернета

Программный интерфейс позволяет быстро и просто получить доступ к источникам из другого программного приложения. Например: зайти в личный кабинет интернет-магазина или профиль в социальной сети можно и через аккаунт на других веб-ресурсах. Это возможно благодаря API, установленным непосредственно в социальные сети. Код и API этих платформ дает клиентам доступ к другим приложениям.

В партнерском маркетинге

Работа с API в партнерском маркетинге облегчила труд программистов. Ранее они работали в SaaS – интеграции, которая предоставляла ПО как услугу посредством веб-интерфейса. Большую часть работы в сервисе приходилось выполнять вручную: это замедляло развитие партнерских программ и отражалось на стоимости работ. Теперь же программисты используют API как сравнительно быстрый и дешевый аналог.

В бизнесе

API нужны, чтобы:

- проводить транзакции;
- интегрировать потоки данных с клиентами и партнерскими системами;
- повысить безопасность автоматизированных процессов;
- развивать собственные приложения;
- внедрять инновации, например, при работе с клиентами.

В 1990-е годы организация, которая хотела запустить систему управления взаимоотношениями с клиентами (CRM), была вынуждена вкладывать огромные средства в программное обеспечение, оборудование и специалистов. Теперь компании используют облачные службы вроде Salesforce. Доступ на уровне API к функциям Salesforce позволяет бизнесу включить ключевые элементы функциональности CRM-системы — например, возможность просматривать историю клиента.

В разработке

Разработчикам программный интерфейс позволяет:

- упростить и ускорить выпуск новых продуктов, так как можно использовать уже готовые API для стандартных функций;
- сделать разработку более безопасной, выведя ряд функций в отдельное приложение, где они будут скрыты;
- упростить настройку связей между разными сервисами и программами и не сотрудничать для разработки своего продукта с создателями различных приложений;
- сэкономить деньги, так как не нужно разрабатывать все программные решения с нуля.

До появления Windows и других графических операционных систем программистам для создания окон на экране компьютера приходилось писать тысячи строк кода. Когда же Microsoft предоставила разработчикам API Windows, на создание окон стало уходить всего несколько минут работы.

Примеры API в нашей жизни

Кнопки авторизации. На многих сайтах есть кнопки, позволяющие зарегистрироваться через уже существующие аккаунты на популярных площадках и в

соцсетях. Это возможно благодаря API, которые есть у Google, Facebook, Apple, Twitter, «VK» и других компаний.

Заказ авиабилетов. Многие пользуются агрегаторами билетов, такими как Aviasales и SkyScanner. Такие сервисы собирают информацию о стоимости авиабилетов в разных авиакомпаниях и отображают ее в едином окне. Это позволяет реализовать API, встроенный в сайты авиакомпаний, который помогает в реальном времени обновлять информацию о направлениях и стоимости.

Навигация на сайтах и в приложениях. Крупные компании, в том числе Apple, Google, «Яндекс» и другие, разработали API, позволяющие подключить собственный картографический сервис к другим площадкам. Так, в “Яндекс.Карты” встроены сервисы «Транспорт» и «Пробки». Многие приложения на Android, например, по доставке еды или для спорта, используют встроенный в ОС API, чтобы подключить карты Google к своему сервису. На iOS аналогичная ситуация с Apple Maps.

Особенности современного API

API сегодня — это товар. А его покупатели – разработчики ПО. Поэтому API постоянно дорабатывают, делают его доступным и удобным, выпускают регулярные обновления и занимаются поддержкой. Высокий спрос повышает качество продукта. Чтобы привлечь больше пользователей, разработчики создают новые идеи и дорабатывают ошибки, делают связь между приложениями надежнее и безопаснее.

тестирование API

API TESTING — это тип тестирования программного обеспечения, который проверяет интерфейсы прикладного программирования (API). Целью API-тестирования является проверка функциональности, надежности, производительности и безопасности интерфейсов программирования. В тестировании API вместо использования стандартных пользовательских входов (клавиатура) и выходных данных вы используете программное обеспечение для отправки вызовов в API, получения выходных данных и записи ответа системы. Тесты API сильно отличаются от тестов GUI и не будут концентрироваться на внешнем виде приложения. Он в основном концентрируется на уровне бизнес-логики архитектуры программного обеспечения.

Понимание подходов тестирования API

Соглашение о подходе к тестированию API — важная часть стратегии тестирования. Желательно договорится об этом перед началом разработки API. Ниже перечислены

несколько принципов:

- Прозрачное описание области тестирования и хорошее понимание функциональности API;
- Понимание основных методологий тестирования, таких как анализ граничных значений и классов эквивалентности — часть составления тест-кейсов для API;
- Планирование, определение и подготовка входящих параметров, пустых запросов и тестовых примеров для API;
- Определение и сравнение ожидаемых и фактических результатов. Проверка того, что отличий между ними нет.

Подход Большого взрыва (Big Bang Approach)

Все или практически все разработанные модули собираются вместе в виде законченной системы или ее основной части, и затем проводится интеграционное тестирование. Такой подход очень хорош для сохранения времени. Однако, если Test case и их результаты записаны неверно, то сам процесс интеграции сильно осложнится, что станет преградой для команды тестирования при достижении основной цели интеграционного тестирования;

Инкрементальный подход (Incremental Approach)

при таком подходе тестирование выполняется путем объединения двух или более логически связанных модулей. Затем другие связанные модули поэтапно добавляются и тестируются для правильного функционирования. Процесс продолжается до тех пор, пока все модули не будут соединены и успешно протестированы. Осуществляется разными методами:

- **Нисходящий подход (Top-Down Approach)**: Вначале тестируются все высокоуровневые модули, и постепенно один за другим добавляются низкоуровневые. Все модули более низкого уровня симулируются заглушками с аналогичной функциональностью, затем по мере готовности они заменяются реальными активными компонентами. Преимущества: Локализация неисправностей проще. Возможность получить ранний прототип. Основные недостатки дизайна могут быть найдены и исправлены в первую очередь. Недостатки: Нужно много заглушек. Модули на более низком уровне тестируются недостаточно;
- **Восходящий подход (Bottom-Up Approach)**: В восходящей стратегии каждый модуль на более низких уровнях последовательно тестируется с более высокоуровневыми модулями, пока не будут протестированы все модули.

Требуется помощь драйверов для тестирования. Данный подход считается полезным, если все или практически все модули, разрабатываемого уровня, готовы. Также данный подход помогает определить по результатам тестирования уровень готовности приложения. Пример низкоуровневого модуля - модуль, который заведует хранением токенов авторизации. Высокоуровневый - модуль авторизации, в состав которого помимо прочего входит модуль токенов. Преимущества: Локализация ошибок проще. Не тратится время на ожидание разработки всех модулей, в отличие от подхода Большого взрыва. Недостатки: Критические модули (на верхнем уровне архитектуры ПО), которые контролируют поток приложения, тестируются последними и могут быть подвержены дефектам. Ранний прототип невозможен;

- **Гибридный, сэндвич-подход (Sandwich/Hybrid/Bi-Directional Approach)**: Представляет собой комбинацию восходящего и нисходящего подходов. Здесь целью является средний слой, в то время как драйверы заменяют верхний слой, а заглушки нижний пока компоненты этих слоев не будут разработаны.

Список функционала, покрываемый API тестированием

- **Проверка бизнес-логики** - формат и набор данных, которые получают с сервера, с привязкой к бизнес-логике. Работа продукта должна выполнять те функции, которые были разработаны в процессе создания бизнес требований.
- **Тестирование безопасности** - чтобы убедиться, что реализация API защищена от внешних угроз.
- **Тестирование пользовательского интерфейса** - выполняется как часть сквозных интеграционных тестов. чтобы убедиться, что все аспекты пользовательского интерфейса работают должным образом.
- **Интеграционное взаимодействие сервисов** - соглашение между клиентом и сервером или между двумя приложениями. Перед тем, как начать любое тестирование функциональности, важно убедиться в правильности соглашения. Это можно сделать сначала проверив спецификацию (или само соглашение службы) и убедившись, что:

эндпоинты правильно именованы;

ресурсы и их типы правильно отражают объектную модель;

нет отсутствующей или дублирующей функциональности;

отношения между ресурсами правильно отражаются в API.

- **Тесты на дефекты в процессе работы** ПО. Тесты, помогающие наблюдать за выполнением ПО и фиксировать проблемы, связанные с неопределенностью параллелизма (race conditions), исключениями и утечкой ресурсов. Ниже содержится краткая информация об этих факторах.
    - **Тесты мониторинга API:** Проверяют реализацию на различные ошибки, сбои внутренних обработчиков и другие неотъемлемые проблемы в кодовой базе API. Эти тесты гарантируют отсутствие дыр, которые могут привести к проблемам с безопасностью приложения.
    - **Ошибки выполнения:** Проверка позитивных запросов к API на корректность ответа всегда присутствует в плане тестирования, однако проверка негативных сценариев не менее важная часть стратегии тестирования API.
    - **Утечка ресурсов**. Негативные тесты для проверки сбоев в работе основных ресурсов API путем отправки некорректных запросов. Ресурсы здесь — это память, данные, уязвимости, операции тайм-аута и так далее.
    - **Отслеживание сбоев**. Обнаружение сбоев сетевого обмена. Сбои аутентификации из-за предоставления неверных учетных данных являются примером сценария обнаружения ошибок. Эти тесты гарантируют, что ошибки фиксируются, а затем устраняются.

Контрактное тестирование API

В общем случае контрактное тестирование или **Consumer Driven Contract (CDC)** является связующим звеном между модульным и интеграционным тестированием.

Каждый интерфейс имеет поставщика (supplier) и потребителя (consumer). Само собой, сервисы поставщика и потребителя распределены между разными командами, мы оказываемся в ситуации, когда четко прописанный интерфейс между ними (или контракт) просто необходим. Обычно многие подходят к этой проблеме следующим образом:

- Пишут подробное описание спецификации интерфейса - контракт;
- Реализуют сервис поставщика согласно спецификации;
- Передают спецификацию интерфейса потребителю;
- Ждут реализации от другой стороны;
- Запускают ручные системные тесты, чтобы всё проверить

Сегодня многие компании заменили последние два шага на автоматизированные контрактные тесты, которые регулярно проверяют соответствие описания и реализации у поставщика и потребителя определенного контракта. Что является набором регрессионных тестов, которые обеспечивают раннее обнаружение отклонения от контракта.

Можно выделить следующие тезисы для выполнения контрактного тестирования:

- Команда разработчиков (тестировщиков) со стороны потребителей пишет автоматизированные тесты с ожидаемыми параметрами со стороны потребителей.
- Тесты передаются команде поставщика.
- Команда поставщика запускает контрактные тесты и проверяет результат их выполнения. Если происходит падение тестов, то команды должны зафиксировать сбой и перепроверить документацию (согласованность разработки).

Минусы CDC:

1. CDC тесты не заменяют E2E тесты.
2. CDC тесты не заменяют функциональные тесты API.
3. CDC тесты дороже в поддержке, чем функциональные тесты;
4. Для реализации CDC-тестов нужно использовать (изучать) отдельные инструменты тестирования - Spring Cloud Contract, PACT

Инструменты для тестирования API

Postman

Этот инструмент перерос с плагина для Chrome до многофункционального инструмента тестирования, совместимого с платформами Windows и Mac. Postman считается оптимальным выбором для тех, кто хочет использовать язык разработчика, а не работать с кодировками в интегрированной программной среде. Можно выделить следующие преимущества инструмента:

- широкий интерфейс делает тестирование удобным и простым;
- можно использовать технологию автоматического и исследовательского тестирования;
- удобный в работе клиент REST;

- широкий пакет возможностей для интеграции, включая совместимость с форматами RAML и Swagger;
- корректно функционирует в программах, разработанных для Windows, Linux, Mac и Chrome;
- включены функции Document, Monitoring, Run и Test;
- для полномасштабного использования не нужно учить новый язык программирования;
- интегрирована система легкого обмена полученным опытом между участниками команды. Программа систематизирует запросы и ожидаемые ответы, рассылает результаты участникам проекта.

Apache JMeter

Изначально это открытое программное обеспечение разрабатывалось для нагрузочного тестирования. Но сегодня оно активно используется для функционального тестирования.

Основные особенности:

- автоматически поддерживает файлы CSV, что необходимо для быстрого указания уникальных параметров тестирования;
- результаты тестирования воспроизводятся;
- может применяться в рамках динамического и статического API testing для определения производительности технических ресурсов;
- интеграция Jenkins и JMeter открывает возможность включать тесты в конвейерные обработки CI.

Katalon Studio

Это бесплатный API testing инструмент, что особенно нравится начинающим разработчикам. Он предоставляет общую среду для разработки и выполнения UI-функционала, тестирования мобильных продуктов и служб API/Web. Главным преимуществом решения является его способность комбинации уровней Business (службы API/Web) и UI. Инструмент полностью совместим с операционными системами Mac OS, Linux и Windows.

Katalon Studio поддерживает запросы RESTful и SOAP с разными командами (PUT, DELETE, GET, POST). При этом существует возможность настройки параметров команд.

Главные особенности:

- поддержка API testing запросов RESTful и SOAP;
- множество интегрированных ключей для постановки тестовых задач;
- комбинированная поддержка теста между верификациями API и UI;
- поддержка библиотеки проверки утверждений AssertJ;
- возможность применения подхода, управляемого полученными данными;
- использование исследовательского и автоматического типа тестирования;
- хорошее решение как для начинающих разработчиков, так и для опытных.

SoapUI

Это удобный консольный инструмент, дающий возможность быстрого и простого тестирования SOAP, API REST и веб-сервисов. С использованием SoapUI можно открыть исходный документ и интегрировать необходимый набор функций, включая описанные ниже:

- быстрое создание программного кода с использованием технологии Groovy;
- быстрое создание тестов методом «указания клика» и привычного перемещения объектов с помощью мыши;
- мощный API testing с загрузкой данных из Excel, БД и файлов. Такой алгоритм позволяет реалистично симулировать реальное взаимодействие юзера и API;
- присутствует поддержка асинхронного тестирования и возможность запуска комплексных сценариев;
- сканирование безопасности и загрузка тестов могут использоваться повторно, для чего придется сделать всего пару кликов.

Smartbear ReadyAP

В дополнение к тестированию безопасности платформа Smartbear ReadyAPI предназначена для оптимизации использования и производительности API в любой среде. Человек может выполнить анализ безопасности API одним щелчком мыши,

однако инструмент также поддерживает другие важные функции, такие как просмотр того, насколько хорошо или плохо API сможет справиться с неожиданной нагрузкой или внезапным скачком в использовании. Вы можете настроить ReadyAPI для генерации определенных видов трафика, который, как ожидается, будет обрабатывать API. Программа также может записывать трафик API в режиме реального времени, чтобы будущие тесты были более точными и адаптированными к уникальной среде, в которой они будут работать. Кроме того, платформа способна импортировать практически любую спецификацию или схему для тестирования API с использованием самых популярных протоколов. Изначально ReadyAPI поддерживает Git, Docker, Jenkins, Azure DevOps, TeamCity. Его также можно запускать в любой среде (от разработки до обеспечения качества, даже задолго до запуска API).

Assertible

Assertible представляет собой инструмент тестирования API, который в первую очередь акцентируется на автоматизации процессов и надежности.

- Обеспечивает автоматическое тестирование API на каждом этапе процесса интеграции и поставки программного обеспечения.
- Осуществляет поддержку текущих тестов API после внедрения приложений и интегрирует их с привычными инструментами, такими как GitHub, Slack, и Zapier.
- Поддерживает проверку подлинности ответов HTTP при помощи готовых операторов подтверждения отсутствия ошибок, таких как проверка достоверности JSON Schema и проверка целостности данных JSON Path

Apigee

Apigee является кросс-«облачным» средством тестирования API, позволяющим пользователям измерять и тестировать производительность API, обеспечивать техническую поддержку и разработку API при помощи других редакторов, таких как Swagger.

- Данный инструмент является многошаговым и находится под управлением Javascript
- Он позволяет разрабатывать, отслеживать, выполнять разворачивание и масштабирование API
- Идентифицирует проблемы путем отслеживания трафика API, уровня ошибок и времени ответа

- Легко создает прокси API из Open API Specification и выполняет их развертку в «облаке»
- Модели разворачивания в «облаке», локального разворачивания и гибридного разворачивания работают на основе одного кода
- PCI, HIPAA, SOC2, и PII для приложений и API
- Apigee разработан специально для цифрового бизнеса и задач с интенсивной обработкой данных на под управлением мобильных платформ API и приложений, которые управляют ним.

Tricentis Tosca

Tricentis Tosca представляет собой платформу непрерывного тестирования для Agile и DevOps. Среди преимуществ Tricentis Tosca следует отметить:

- Поддержку многих массивов протоколов: HTTP(s) JMS, AMQP, Rabbit MQ, TIBCO EMS, SOAP, REST, IBM MQ, NET TCP
- Интеграцию в циклы Agile и DevOps
- Максимизацию многократного использования и способность к сопровождению средств автоматизации тестирования на основе использования моделей
- Тесты API могут использоваться как на мобильных, так на браузерных и пакетных приложениях и т.д.
- Достигнута автоматизация, поддерживаемая новыми технологиями
- Снижено время, необходимое на проведение регрессивного тестирования