Rest- архитектурный стиль, Soap – формат обмена сообщениями

* Пример реализации **SOAP**: XML поверх SOAP через HTTP

Специфика SOAP это формат обмена данными это всегда xml в котором :

- envelope(конверт) – корневой элемент, который определяет сообщение и пространство имен

- header- содержит атрибуты сообщения н-р информация о сетевой маршрутизации

- body – содержит сообщение которым обмениваются приложения

-fault – содерждит инфу об ошибках которые произошли при обработке сообщения. И запрос и ответ должен соответствовать архитектуре SOAP

Специфика REST – использование HTTP в качестве транспортного протокола

В SOAP используется только SOAP XML для запросо и ответов

В REST такого нет можно обмениваться сообщениями на основе JSON или XML, JSON является самым популярным

SOAP использует WSDL(Web services description language) – это язык описания веб сервисов и доступов к ним на языке XML

REST ен имеет стандартного языка определения сервиса, юзают OPEN API и SWAGGER

API интерфейс общения между двумя независимыми компонентами по. В по есть некий апи провайдер – часть приложения предоставляющая какие-то функции и/или услуги и консьюмер( клиент) тот, кто хочет воспользоваться этими услугами. От клиента к провайдеру идет запрос на предоставление услуг и получает ответ от провайдера может ли он пользоваться этими услугами. При этом их общение происходит по контракту какой запрос можно делать провайдеру и какой респонз он может отправить.

Классификация апи по типам релизной политики :

-open api это открытые общедоступные в интернете

-partners api покупка апи для его использования но для компании этот апи будет конечным продуктом

-privat api

Чаще юзают открытые апи потому что легче переиспользовать в других проектах

Рест это архитектурный стиль имеющий 6 свойств который в итоге призван сделать легко расширяемые обновляемые и поддрживаемые приложение:

1. Униформ интерфейс т.е например если у приложения несколько апи у них должны быть принципы путей общения, как осуществлять запрос какие ответы будут воспроизводиться
2. Клиент-сервер, четкое разделение клиента и сервера и они должны быть обособлены и не влиять друг на друга
3. Стейтлесс о том, что сервер не должен хранить клиентскую инфу и каждый раз когда клиент осуществляет запрос на сервер его запрос должен содержать все необходимое для предоставление ему ответа
4. Source если он может быть кешируемый то он должен содержать хедеры сообщения

Система должна состоять из лееров-прослоек, накладываются друг на друга, и они не долдны знать что происходит в другом слое но они должны общаться

Опциональный принцип в том что ограничиваемся тем что действительно нужно клиенту, т7е мы должны отдавать ту часть кода клиенту которую он сейчас запросил

НА ЧТО СМОТРЕТЬ ПРИ ТЕСТИРОВАНИИ РЕСТИ АПИ

Рести апи это веб апи который строится на принципах application presentation session transport net-work data link и physical это не протокол в отличае от соап, не строго регламентируемое это просто набор принципов которые если реализуется это рестфул если нет то оно все равно будет работать. Рест работает на самом высоком уровне поверх хттп протокола и полностью привязан к хттп, если соап может использовать другие протоколы, то для реста только хттп.

Апи скрипты трансформируются в хттп запрос который дальше обрабатывается на стороне сервера

Структура хттп запроса :

- метод. Сам хттп метод с которым мы обращаемся на сервер

- урл, куда обращаемся

-параметры( опционально)

-хедеры, т.е сервисная информация

-бади ( опционально) может быть или не быть, зависит от используемого метода

ХТТП МЕТОДЫ

- get запрос который получает информацию, который как правило делает селект запрос из бд

-post записывает в базу данных, т.е на уровне баз данных делает инсерт

-put метод для обновления уже существующих записей, при этом ччитается что пут должен обновлять полностью всю модель, объект, а патч обновляет объект частично. Т.е это запросы апдейт на уровне базы данных

-delete запрос который удаляет запись на уровне базы данных

-patch

-options