Jornada AppSec

Nivelando os conceitos de API

Autor: Felipe Pinheiro

GitHub: https://github.com/pinheiro-felipe/jornada-appsec-criando-api-de-postagem-de-status-do-pedido-para-delivery

4º Dia: Quarta, 12 de junho de 2024

O vento começou a soprar, as nuvens cinzas apareceram e logo o sol foi encoberto

Bom dia pessoal, hoje nós vamos continuar nosso processo de desenvolvimento da plataforma de delivery, mas antes precisamos entender alguns conceitos como: API Design Fisrt, URL, HTTP, Subdomínio, Contexto delimitado e etc. Depois de entendidos, podemos começar a modelar nossa API.

Após o mapeamento dos comandos, eventos e atores vamos começar a desenhar nossa API usando a abordagem API Design First. E por isso precisamos entender alguns conceitos.

9- Nivelamento de conceitos de API: Quais são os conceitos de API que precisamos aprender?

API Design First, é um modelo de desenvolvimento de software onde a criação do software começa primeiro pela API. Se trata de antes mesmo de escrever o código, descrever a api de uma forma que facilite o entendimento humano e mesmo assim computadores consigam entender.

Pontos positivos da abordagem:

Consumidores consultam a API facilmente. As APIs ficam no catálogo de APIs em um portal do desenvolvedor que possibilita o seu uso, compreensão do comportamento e propósito.

Desde o início do desenvolvimento, deixa claro através de contratos o que a API vai receber, o que vai fazer e o que vai retornar a quem a executou.

Cenários com informações incompletas, erros inesperados ou problemas nos disparos de eventos com base em cenários específicos, podem ser previamente detectados com mais facilidade e corrigidos ainda em tempo de design, encurtando assim o ciclo de vida do desenvolvimento.

O contrato é disponibilizado em uma versão inicial para retornar dados fictícios, liberando assim os desenvolvedores que consumirão a API para iniciarem o seu ciclo de desenvolvimento.

Dessa forma não se constroi a API apenas no final do ciclo do desenvolvimento backend e sim no começo do desenvolvimento, para utilizá-la como referência seja para o frontend ou para o backend.

Desenvolvedores frontend não precisarão esperar pelo desenvolvimento backend terminar a API, para aí sim construírem o frontend. Muito menos criar alguma API fake para simular a API que ainda não existe. API Design First, possibilita o paralelismo no desenvolvimento, onde frontend e backend estão construindo olhando para os contratos que foram desenhados.

Qualquer nova alteração que surgir, o frontend já consegue testar seu funcionamento sem esperar pelo backend. Da mesma forma, assim que o contrato for implementado pelo backend, deverá possuir o mesmo resultado, não alterando em nada a vida de quem consome a API.

Para nos ajudar a modelar o design da API, vamos olhar para nossos comandos, eventos, atores e realizar as seguintes ações:

Identificar ator
Identificar subdomínio
Identificar contexto delimitado (Possíveis serviços ou microsserviços)
Identificar agregado
Identificar recursos
Identificar relacionamento entre recursos
Definir o path de cada comando juntamente com seu recurso

Sobre URL:

Utilizamos abaixo grande parte do texto retirado do site da UFSCar - Universidade Federal de São Carlos para explicar o conceito de URL. Fizemos pequenas modificações quando necessário e incluímos trechos do developer.mozilla.org.

3.1.Internet: conceitos básicos. Ufscar, 1997. Disponível em: <https://www.dm.ufscar.br/profs/waldeck/curso/html/internet/basico.html>. Acesso em: 12 Jun. 2024.

"A URL (Uniform Resource Locator) exibe a localização exata de documentos, objetos, ligações, referências e programas executáveis na vastidão da Internet. Em termos simples, qualquer coisa a que referimos na Internet tem uma URL. Também são chamadas de endereços virtuais.

A URL é composta por várias partes:

protocolo://host:porta/caminho?parâmetro=valor#fragmento

Exemplo 1 de URL no padrão descrito acima:

http://www.example.com:80/arquivos/myfile.html?key1=value1&key2=value2#AlgumlugarNaPagina

Exemplo 2 de URL no padrão descrito acima:

https://www.carrosusados.com:443/locais/riodejaneiro?cor=azul&ano=2020#ModeloEsporte

As partes da URL são:

Protocolo (Protocol) - Define o protocolo de acesso ao objeto. Alguns protocolos são:

http - É o protocolo principal da web utilizado para serviços de documentos de hipertexto. Em nossa URL do exemplo 2, o protocolo é : https, versão

segura do protocolo http e que roda na porta 443, enquanto o http por padrão na porta 80.

ftp - É o protocolo usado para transferência de arquivos.

mailto - É o protocolo utilizado pelos serviços de e-mail para enviar mensagens eletrônicas.

telnet - É o protocolo de acesso remoto.

Host (Nome do domínio) - Um nome que identifica um computador na internet (ou seu número de endereço IP).

Na fase de desenvolvimento do software, nosso host acaba sendo local. Localmente o host é representado pela palavra localhost que não é apenas o nome do nosso servidor virtual, mas também o nome do nosso domínio.

Também podemos representar nosso host local, pelo nosso endereço de IP ou pelo endereço IP de loopback 127.0.0.1 que representa nosso próprio computador. Isso para se referir ao computador em que o software está hospedado.

Mas quando publicamos nossa aplicação na internet, possuímos um host online que hospeda nosso domínio para ser acessado na internet.

www.carrosusados.com é o nome de domínio hospedado num host. Ele indica qual servidor web será solicitado. Alternativamente, é possível utilizar um endereço IP, mas isso pode ser menos conveniente e não é muito utilizado na Web.

Ao realizar uma requisição no endereço acima, você adquire dois domínios: o de nível superior e o de segundo nível, o SLD (second-level domain) que em tradução é o domínio de segundo nível, e representa o nome exclusivo da empresa ou companhia. Em nossa URL do exemplo 2, o nome exclusivo da empresa é carrosusados.

O TLD (top-level domain) que em tradução livre é domínio de nível superior (também conhecido como extensão do domínio ou domínio de topo), indica o tipo de organização que o site ou plataforma representa. Se é .com, .org, .gov, .edu e etc.

O Estados Unidos, quem criou o internet, adotou seguintes convenções para domínio maximal:

"." = máximo: Representa a origem de todos os domínios.

".com" = Representa domínios comerciais e industriais. Exemplo: www.amazon.com

".edu" = Representa domínios de instituições educacionais. Exemplo: web.mit.edu

".gov" = Representa domínios governamentais. Exemplos: www.nasa.gov, www.whitehouse.gov, etc.

".mil" = Representa domínios de instituições militares

".org" = Representa domínios de instituições sem fins lucrativos. Exemplo: www.linux.org

".net" = Representa domínios de provedores de acesso. Exemplos: ibm.net, algarnet.net, etc.

Para outros países, são aplicados o código ISO (International Standards Organization) designando o país.

No caso dos países abaixo e outros não exemplificados, o ccTLD em tradução é domínio de nível superior de código do país (country code top level domain) que informa o país ou território.

".br" = Representa domínios do Brasil

".fr" = Representa domínios da França

".pt" = Representa domínios de Portugal

".ne" = Representa domínios dos países baixos

Porta (Port) - Número da porta TCP associado ao protocolo (Quase nunca aparece devido a uma padronização. Por exemplo, convenciona-se que http é dado pela porta 80 e https pela porta 443). Em nossa URL do exemplo 2 que utiliza o protocolo https, a porta é: 443

Caminho (Path) - Caminho de busca e nome de arquivo que permite identificar o objeto dentro do computador.
É o caminho para o recurso no servidor web. Nos primeiros dias da web, um caminho era representado pelo caminho físico correspondente no servidor web. Hoje em dia isso é mais uma abstração tratada pelos servidores web, não sendo necessariamente o endereço físico do arquivo em questão. O

caminho pode possuir parâmetros que descrevem onde procurar recursos específicos. Em nossa URL do exemplo 2, o caminho é: /locais/riodejaneiro, sendo que o rio de janeiro representa o parâmetro no caminho. O caminho seria definido assim: /locais/{nomeDaLocalidade}

Query / Parâmetros (Parameters) - Qualquer informação que o objeto necessita para tornar-se disponível.
São parâmetros extras fornecidos ao servidor Web. Eles são uma lista de pares chaves/valores incluídos depois do símbolo de interrogação "?" e separados com o símbolo de e comercial "&".
O servidor web pode usar esses parâmetros para filtrar dados dentro da solicitação de conteúdo e retornar o recurso para o usuário. Em nossa URL do exemplo 2, os parâmetros são: cor=azul&ano=2020

Fragmento (Anchor) - É uma referência para um subconjunto de um objeto (pode ser uma seção ou subseção de um documento).
É uma âncora para outra parte do próprio recurso. Uma âncora representa uma espécie de "marcador" dentro do recurso, dando ao navegador as instruções para mostrar o conteúdo localizado naquele ponto "marcado". Em um documento HTML, por exemplo, o navegador da scroll para a âncora em um ponto definido; em um vídeo ou áudio, o navegador tenta ir para o tempo que a âncora representa. Vale ressaltar que a parte após o símbolo "#", conhecido como jogo da velha ou hashtag ou como identificador de fragmento, nunca é enviada ao servidor com o pedido. Em nossa URL do exemplo 2, o fragmento é: #ModeloEsporte

Dicas para o path:

Deve refletir o tipo do recurso
Deve ser único
Deve ser explícito
Pode conter id
Deve refletir coleções de itens do tipo do recurso
Deve respeitar a hierarquia

Alguns Métodos HTTP:

POST: Usado para criar um novo recurso enviando os parâmetros no corpo da requisição. Após uma requisição bem sucedida, retorna o status 201 e o recurso criado

PATCH: Usado para aplicar modificações parciais a um recurso. Após uma requisição bem sucedida, retorna o status 200 e o recurso modificado

PUT: Usado para atualizar um recurso existente ou criar um novo recurso se este não existir. Envia-se os parâmetros no corpo da requisição e após uma requisição bem sucedida, retorna o status 200 e o recurso criado ou modificado

GET: Retorna o recurso ou os recursos pesquisados, recebendo os parâmetros quando estes existirem, seja no path ou na query. Após uma requisição bem sucedida, retorna o status 200 e os dados desejados no corpo da resposta

DELETE: Remove um recurso específico. Após uma requisição bem sucedida, retorna o status 204 e o recurso removido