Navegando na Web da API: Conceitos da Web

Erik Wilde API
Academy, CA Technologies Zürich, Suíça
erik.wilde@ca.com

RESUMO

A Web é baseada em vários padrões que, juntos, constituem a superfície da Web: conhecendo e suportando esses padrões, os problemas podem ser resolvidos de maneiras bem conhecidas. Esse padrão geral de design na Web se aplica às APIs da mesma forma que se aplica à Web humana: ao usar um conjunto (evolutivo) de padrões, os desenvolvedores de API se beneficiam por não terem que reinventar a roda, e os desenvolvedores se beneficiam pela mesmo problema sendo resolvido da mesma maneira em uma variedade de APIs. O conjunto de padrões em evolução para APIs da Web pode ser considerado como um conjunto de blocos de construção ou vocabulários para design de API. O Web Concepts é um site (webconcepts.info) e um repositório (github.com/dret/webconcepts) que pode ser usado para gerenciar como esses blocos de construção são usados dentro das organizações, ajudando assim a estabelecer uma cultura de design de API da Web. A ideia principal do Web Concepts é promover a reutilização de padrões e tecnologias existentes e, portanto, tornar mais fácil para as equipes entender quais opções estão disponíveis em geral e talvez quais sejam escolhas populares em sua organização.

CONCEITOS CCS

• **Sistemas de informação** ÿ **Serviços Web; • Software e sua engenharia** ÿ **Documentação; • Geral e referência** ÿ
Padrões de computação, RFCs e diretrizes;

Formato de Referência ACM:
Erik Wilde. 2018. Navegando na API da Web: Conceitos da Web. Em WWW '18 Companion: The 2018 Web Conference Companion, 23 a 27 de abril de 2018, Lyon, França. ACM, Nova York, NY, EUA, 6 páginas. https://doi.org/10.1145/3184558.3188743

1. INTRODUÇÃO

As APIs da Web tornaram-se uma pedra angular importante da Web: elas expõem serviços de maneira programável, da mesma forma que as páginas da Web comuns expõem serviços de maneira acessível por humanos.

Uma das principais diferenças entre as APIs da Web e a Web humana está em termos de diversidade e complexidade. Na Web humana, o acesso é (quase) sempre por meio de um navegador da Web. Embora o HTML5 e seu suporte para aplicativos de navegador muito mais poderosos tenham resultado em um aumento substancial da complexidade técnica1 , a diversidade ainda é relativamente modesta, com a maioria das páginas da Web assumindo um certo suporte mínimo de CSS e scripts em navegadores e, idealmente, tendo alguma estratégia alternativa quando esses pressupostos não são atendidos.

A situação é bastante diferente para APIs da Web, onde não há padrões universalmente aceitos e suportados, exceto o HTTP sendo usado como protocolo para conectar clientes e servidores. A divergência às vezes é muito profunda (como a diferença de opinião se os serviços da Web deveriam usar melhor RPC ou REST como o princípio arquitetônico fundamental). No entanto, enquanto os detalhes de como as APIs da Web são projetadas frequentemente divergem, o fato é que existe algum fundamento compartilhado por causa dos princípios fundamentais universais de identificação de recursos via URI [1] e interação de recursos via HTTP [6].

Além dos próprios fundamentos do URI/HTTP, a “linguagem” das APIs torna-se mais variada. Considere a simples questão de como os dados estruturados são representados: Metamodelos populares para dados estruturados atualmente são provavelmente XML [3], JSON [2] e RDF [4]. Estes já requerem implementações para empregar diferentes conjuntos de ferramentas, para que possam analisar e processar dados estruturados nessas representações . Mas, além disso, também existem modelos padronizados em jogo , seja como formatos de contêiner ou como blocos de construção que estão sendo usados para construir vocabulários maiores. De uma forma ou de outra, esses modelos também se tornam parte da “linguagem” de uma API: os usuários da API precisam entendê-los e trabalhar com eles como destinada.

Além dessas linguagens de representação, a Web também possui um grande conjunto de vocabulários adicionais que as APIs da Web podem usar para projetar e expor seus serviços. Um exemplo popular é o cache HTTP [5], que após a recente revisão do HTTP/1.1 não faz mais parte da especificação principal, mas uma especificação separada que define 5 campos de cabeçalho HTTP e dentro de sua própria “ linguagem de cache” um 12 diretivas de cache HTTP adicionais (que são usadas com o campo de cabeçalho HTTP Cache-Control) e 7 códigos de aviso HTTP (que são usados com o campo de cabeçalho HTTP Warning).

Este exemplo de cache HTTP mostra que existe toda uma “linguagem” adicional sendo definida por uma especificação (neste caso , a RFC 7234 [5] da IETF) que as APIs da Web podem utilizar se quiserem melhorar a maneira como seus recursos podem ser usados com recursos avançados de cache. O uso dessa “linguagem de cache” tem as duas principais vantagens típicas do uso de padrões:

- Não há necessidade de reinventar a roda se houver um padrão que já cobre funcionalidades/recursos que deveriam ser expostos por uma API da Web.
- Os desenvolvedores que trabalham com a API da Web não precisam aprender a maneira específica como a API resolveu esse problema. Se eles conhecerem o padrão, entenderão esse aspecto da API e poderão usar as ferramentas e bibliotecas existentes para oferecer suporte a esse padrão.

Por meio do design de URIs e HTTP e muitos padrões adicionais, a Web é capaz de fornecer esse tipo de reutilização em muitos lugares diferentes. Isso também destaca que, embora a Web seja ótima porque é construída sobre as bases compartilhadas de URI/HTTP, há

1O conjunto de especificações HTML5 e o conjunto de especificações CSS agora são bem mais de 200 especificações combinadas.

WWW '18 Companion, 23–27 de abril de 2018, Lyon, França
ACM ISBN 978-1-4503-5640-4/18/04.
https://doi.org/10.1145/3184558.3188743

ainda existem “linguagens” adicionais usadas em cima dessas que podem ajudar a projetar melhores APIs da Web, permitindo mais reutilização e, assim, tornando mais fácil para os designers de API se concentrarem nos aspectos específicos de suas APIs que realmente precisam de design específico.

2 O MITO DA INTERFACE UNIFORME

Conforme mencionado na introdução, URI/HTTP é a base compartilhada de todas as APIs da Web e, como tal, há uma interface uniforme fundamental que é exposta por todas as APIs da Web: quando você conhece uma URI , deve ser capaz de “seguir seu nariz” sondando o recurso via HTTP OPTIONS e continuando a partir daí. Embora isso seja verdade na teoria, na prática das APIs da Web, muitos padrões diferentes entram em ação, conforme ilustrado na introdução.

Assim, para Web APIs pode ser muito útil para desenvolvedores saber quais padrões uma API usa (ou, usando a noção de padrões como “blocos de construção de linguagens de API”, quais “vocabulários são combinados para definir uma API específica”). O que isso significa é que, além do fato de que os usuários podem de fato explorar uma “linguagem da API” em tempo real (por exemplo, interagir com todos os recursos dela e acompanhar os vários campos de cabeçalho HTTP nas respostas para ter uma ideia do “HTTP linguagem do campo de cabeçalho” dessa API), isso é ineficaz e torna mais difícil para os desenvolvedores entender a linguagem da API.

Por exemplo, os campos de cabeçalho HTTP mencionados acima são uma faceta complexa da arquitetura da Web: existem cerca de 200 campos de cabeçalho diferentes que são especificados e registrados (semi-)oficialmente, e há muitos outros que foram inventados por APIs da Web específicas e são usado apenas nessas APIs. Portanto, qualquer API da Web tem muitos campos de cabeçalho HTTP diferentes que ela pode usar, e quais deles ela usa com frequência não é muito fácil de descobrir.

3 CULTURA DE DESIGN DE API WEB

Esse problema fica mais difícil em cenários de API. As organizações modernas têm muitas equipes diferentes trabalhando em muitas APIs diferentes, e cada uma dessas APIs é o resultado de um exercício de design. Com o surgimento dos microsserviços, as equipes são explicitamente incentivadas a criar as APIs que funcionam melhor para os produtos que estão desenvolvendo.

Os microsserviços tentam equilibrar a independência das equipes para projetar e construir produtos, com o objetivo organizacional maior de acabar com um cenário de API que tenha pelo menos algum nível mínimo de coerência. A abordagem usual hoje em dia é diferente dos primeiros dias de APIs e SOAP, onde o objetivo principal era criar uniformidade padronizando tecnologias e conjuntos de ferramentas. A experiência mostrou que essa abordagem de comando e controle era muito limitante para as equipes e resultava em desenvolvimento de API lento e projetos abaixo do ideal.

Os microsserviços tentam encontrar um equilíbrio entre mais independência para aumentar a velocidade da equipe, mas ainda fornecer às equipes o suporte de que precisam para ter sucesso. Por exemplo, um exemplo típico de uma equipe projetando um serviço seria que eles podem estar interessados em como problemas e APIs semelhantes foram criados dentro da organização. Isso é pedir a cultura de design dentro de uma organização e é útil tanto para a organização (as equipes podem se basear em práticas estabelecidas) quanto para as equipes (as equipes podem se basear em experiências anteriores).

Além disso, esse tipo de cultura de design compartilhada permite que a organização crie ferramentas e suporte em torno de “linguagens de API” compartilhadas. Por exemplo, se as APIs da Web usarem as mesmas representações para documentos iniciais ou relatórios de status, será muito mais fácil para a organização criar ferramentas para pontos de entrada ou relatórios de status que funcionem em APIs. Estabelecer essa cultura de design de API da Web ajuda tanto a organização quanto as equipes individuais.

Observe, no entanto, que essa cultura de design é fluida: é o conjunto compartilhado de práticas (ou seja, blocos de construção da API) que são usados pelas equipes e, como qualquer coisa em TI, essas coisas mudarão com o tempo. Assim, uma representação ideal da cultura de design não seria apenas um instantâneo estático dos blocos de construção em uso, mas capturaria a tendência de seu uso ao longo do tempo.

4 CONCEITOS DA WEB

O assunto principal deste artigo é Web Concepts. Web Concepts é uma coleção de conceitos que juntos formam uma parte considerável da cultura de design e da prática de APIs da Web. No momento em que escrevo, ele captura 730 valores diferentes para 32 conceitos. Um exemplo disso é o conceito de campos de cabeçalho HTTP, dos quais existem 191 tipos diferentes. Outros exemplos de conceitos são códigos de status HTTP, esquemas de URI e tipos de mídia.

A coleção é necessariamente incompleta, por um lado porque para alguns dos conceitos existe um conjunto muito grande de valores registrados que provavelmente muito raramente serão usados em APIs da Web (por exemplo, é o caso de tipos de mídia), e por outro por outro lado, porque a coleção atual é baseada em especificações publicadas ou de rascunho e, em muitos casos, as APIs da Web adicionam seus conceitos privados, mas nunca os registram ou padronizam (como campos de cabeçalho HTTP específicos).

Conceitos da Web estão disponíveis em webconcepts.info e no GitHub como github.com/dret/webconcepts, mas o principal objetivo dos dados não é tanto fornecer uma visão geral confiável de todos os conceitos e valores relevantes, mas servir como ponto de partida para estabelecer a cultura e a prática do design nas organizações.

Examinaremos mais detalhadamente como reutilizar os conceitos da Web na Seção 7 e como usá-los para rastrear a evolução do cenário da API na Seção 8. Mas primeiro é importante observar como os dados são organizados e como eles representam a maneira como em que os conceitos da Web são estabelecidos.

5 MODELO DE CONCEITOS DA WEB

O modelo fundamental do Web Concepts é muito simples: assume que os conceitos e valores para eles são estabelecidos por especificações.
Como o site existe agora, as especificações foram retiradas de algumas organizações selecionadas e, como mostram os números das especificações, com um viés bastante pesado no IETF e W3C:

- Organização Internacional de Padronização (ISO): 3 Especificação ficações
- Internet Engineering Task Force (IETF): 214 Especificações • Java Community Process (JCP): 1 Especificação • Organização para o Avanço da Informação Estruturada Padrões (OASIS): 4 Especificações • World Wide Web Consortium (W3C): 40 Especificações Essa

combinação específica foi influenciada pelo fato de que IETF e W3C criam muitas das especificações fundamentais da Internet e da Web. Isso poderia parecer diferente se alguém olhasse para as APIs da Web em uma vertical específica, que pode usar várias especificações de uma organização com foco nessa vertical. Esta imagem seria

também parecerá diferente se tentar representar também os conceitos e valores que podem não ser "especificações adequadas", mas estão sendo usados em uma variedade de APIs de qualquer maneira.

Para dar conta disso, a Web Concepts trata essas organizações e suas especificações como uma configuração. É fácil adicionar novas organizações, e depois disso, podem ser adicionadas especificações dessas organizações, que definem novos conceitos e valores. A configuração principal das especificações diz respeito às convenções de nomenclatura e às formas como elas são acessíveis on-line e podem ser identificadas por URI.

Da mesma forma que as organizações podem ser facilmente configuradas, o mesmo vale para os conceitos. No momento em que escrevo, existem 32 conceitos, e o número de valores para eles varia muito, como pode ser visto aqui:

• Esquemas de autenticação HTTP (10) • Diretivas de cache HTTP (15) • Codificações de conteúdo HTTP (10) • Parâmetros encaminhados HTTP (4) • Campos de cabeçalho HTTP (191) • Preferências HTTP (4) • Unidades de intervalo HTTP (3) • HTTP Métodos de solicitação (39) • Códigos de status HTTP (62) • Codificações de transferência HTTP (7) • Códigos de aviso HTTP (7) • Reivindicações de token da Web JSON (8) • Métodos de confirmação JWT (4) • Relações de link (98) • Tipos de mídia (104) • Tipos de token de acesso OAuth (2) • Tipos de resposta de terminal de autorização OAuth (2) • Metadados dinâmicos de registro de cliente OAuth (20) • Erros de extensões OAuth (4) • Parâmetros OAuth (26) • Métodos de autenticação de terminal de token OAuth (3) • Respostas de introspecção de token OAuth (12) • Dicas de tipo de token OAuth (2) • URIs OAuth (5) • Métodos de desafio de código PKCE (2) • URIs de perfil (1) • Sufixos de sintaxe estruturada (10) • Esquemas de URI (32) • Namespaces URN (14) • URIs conhecidos (21) • Namespaces XML (5) • Esquemas XML (3)

Acrescentar valores de Web Concept combina especificações e conceitos. Para cada especificação, um recurso separado é criado. Este recurso descreve a própria especificação (com metadados simples, como um título e um resumo) e, em seguida, contém um conjunto de valores. Para cada valor, é especificado a que conceito se destina, qual é o valor , como é descrito e onde a documentação sobre ele pode ser encontrada online.

Isso significa que o estado atual do Web Concepts é representado por 262 recursos de especificação (combinando as especificações de as organizações listadas acima), e que combinados esses 262 recursos de especificação contêm 730 valores (combinando os valores para os 32 conceitos listados acima).

A adição de valores é feita adicionando especificações: Um novo recurso é criado que representa a especificação e os valores que ela define. Em seguida, o site da Web completo (que atualmente é um site Jekyll estático) é regenerado e as novas especificações e valores são adicionados nos vários locais onde são listados e vinculados.

Este modelo descreve como os Web Concepts são gerenciados e publicados como um site. No entanto, o objetivo dos conceitos da Web é também servir como uma fonte legível por máquina de informações sobre conceitos, valores e especificações da Web. Para atingir esse objetivo, a geração do site da Web também gera dados JSON que representam especificações e conceitos/valores da Web. Esses dois documentos JSON estão interligados e qualquer um pode servir como ponto de partida, dependendo se o interesse é principalmente em especificações ou em conceitos.

5.1 Especificações A

Listagem 1 mostra a representação JSON das especificações. É uma parte de um documento JSON maior que representa todas as especificações e mostra apenas um exemplo (do total atual de 262 especificações).

A estrutura geral é uma das organizações e séries de especificações, neste caso a organização é o IETF e a série de especificações é seu fluxo de publicações RFC . Tanto a organização quanto a série possuem identificadores únicos que podem ser usados para referências cruzadas de especificações e conceitos, e também servem como pontos de entrada legíveis por humanos na Web.

Uma especificação individual possui um identificador dentro da série e, mais uma vez, também possui um URI para referência cruzada e uma página de destino legível por humanos. A especificação tem alguns metadados associados a ela, principalmente um título e um nome (e um resumo). Ele também possui um identificador URI atribuído pela própria organização e um URL (possivelmente diferente) onde a especificação pode ser acessada online.

As principais informações sobre a especificação são uma matriz de conceitos. Cada um desses conceitos é definido por um identificador de conceito (para referência cruzada com as informações de conceito detalhadas descritas na seção a seguir) e um identificador de valor que também serve para fins de referência cruzada. Tanto o conceito quanto o identificador de valor devem ser usados em conjunto com o documento JSON que representa todos os conceitos e valores (mas também servem como páginas de entrada legíveis por humanos para representações da Web).

5.2 Conceitos Todos

os conceitos e valores são representados em um segundo documento JSON, que semelhante à especificação JSON que descreve todas as especificações, descreve todos os conceitos e valores. A Listagem 2 mostra um trecho deste documento JSON descrevendo um valor (do total atual de 730 valores de conceito).

A partir dos 32 conceitos da Web atualmente configurados, o documento JSON é estruturado por conceito. Cada conceito tem alguns metadados associados a ele, que são seu nome no singular e no plural, bem como um id (que é um recurso legível por humanos,

Listagem 1: Estrutura JSON representando um conceito da Web

```
{{
  "IETF": {
    "id": "http://webconcepts.info/especificações/IETF/",
    "nome": "Força-Tarefa de Engenharia da Internet",
    "curto": "IETF",
    "série": [{{
      "RFC": {
        "id": "http://webconcepts.info/especificações/IETF/RFC/",
        "nome": "Solicitação de comentários",
        "curto": "RFC",
        "especificações": [{
          "2648": {
            "id": "http://webconcepts.info/specs/IETF/RFC/2648",
            "título": "Um espaço de nomes URN para documentos IETF",
            "nome": "RFC 2648",
            "URL": "urn:ietf:rfc:2648",
            "URL": "http://tools.ietf.org/html/rfc2648",
            "resumo": "
              Um sistema para Nomes Uniformes de Recursos (URNs) ...",
            "conceito": [{
              "id": "http://webconcepts.info/conceitos/urna-namespace/ietf"
            }]
          }
        }]
      }
    }]
  }
}}
```

Listagem 2: Estrutura JSON representando um conceito

```
{
  "conceito": "http://webconcepts.info/conceitos/http-método",
  "nome-singular": "Método de solicitação HTTP",
  "nome-plural": "Métodos de solicitação HTTP",
  "registro": {
    "id": "http://webconcepts.info/conceitos/http-método/http-métodos",
    "valores": [{
      "id": "http://webconcepts.info/conceitos/método-http/GET",
      "detalhes": [{
        "descrição": "O método GET...",
        "documentação": "http://tools.ietf.org/html/rfc7231#section-4.3.1",
        "especificação": "http://webconcepts.info/specs/IETF/RFC/7231",
        "spec-name": "RFC 7231"
      }]
    }]
  }
}
```

mas também se correlaciona com o identificador que é usado nos dados de especificação). Para aqueles conceitos que possuem um registro mantido oficialmente, o link para ele está incluído.

(Neste ponto vale a pena mencionar que o registro oficial em todos os casos não contém exatamente as mesmas entradas listadas em o site atual da Web Concepts. A razão para isso é que o site também lista muitos valores que foram propostos, como em rascunhos da Internet, mas ainda não foram oficialmente adicionados ao registro. Situações semelhantes ocorrerão quando as organizações usarem conceitos da Web como ponto de partida, mas depois adicionarem conceitos e valores que usam e, portanto, documentam para seu próprio cenário de API, mas que nunca devem ser inseridos nos registros oficiais.)

Cada conceito é representado por uma matriz de valores (que são colhidos de todas as especificações). Cada um deles tem um valor real e, em seguida, lista o identificador do conceito , bem como o id do valor (que é um URI que pode ser cruzado com os dados de especificações). O valor também contém detalhes, que são representados como uma matriz, pois alguns valores são definidos em mais de uma especificação. Os detalhes contêm uma descrição, um link para a documentação on-line e, para fácil referência cruzada, também identificam a especificação, tanto pelo identificador quanto pelo nome da especificação.

6 CONCEITOS DE USO DA WEB

Uma maneira óbvia de usar o Web Concepts é servir como informações auxiliares em ferramentas e documentação. Por exemplo, quando uma ferramenta se concentra em APIs REST e possui abstrações de design ou programação que usam conceitos fundamentais da arquitetura da Web , como métodos de solicitação HTTP e códigos de status, em vez de compilar essas informações manualmente, elas simplesmente podem ser reutilizadas conforme fornecidas pela Web Conceitos.

Ao usar o Web Concepts, três opções são possíveis:

(1) Uma opção é baixar o JSON do site, criando assim um instantâneo das opções existentes no momento. Isso poderia ser combinado com atualizações ocasionais, mas essencialmente ainda permaneceria um modelo instantâneo.
(2) Uma segunda opção é carregar dinamicamente o JSON, idealmente combinado com algum mecanismo de cache para que todas as atualizações feitas no site sejam refletidas dinamicamente quando o site for atualizado.
(3) Uma terceira opção é clonar o site e controlar sua evolução separadamente. Isso pode combinar várias técnicas, como aparar conteúdo, adicionar especificações personalizadas e ainda obter alterações periódicas do upstream para que quaisquer alterações no site sejam refletidas no clone.

Nenhuma dessas opções é, por definição, melhor que a outra. É simplesmente uma questão de quanto dinamismo é necessário e quanto controle.

No entanto, usar conceitos da Web para rastrear a evolução do cenário de padrões da API da Web é apenas um ponto de partida. As coisas ficam mais interessantes quando se trata de reutilizar conceitos da Web para criar uma versão personalizada, conforme descrito na seção a seguir.

7 REUTILIZANDO CONCEITOS DA WEB

Embora os conceitos da Web possam ser usados simplesmente para rastrear o cenário dos padrões da API da Web, eles também podem ser usados como ponto de partida para criar uma versão mais personalizada da "linguagem de design" das APIs nas organizações.

Um exemplo são os vocabulários especiais que podem ser usados dentro das organizações. Em alguns casos, eles podem ser cobertos por conceitos de nível da Web, como tipos de mídia. No entanto, pode haver outros casos em que os vocabulários são usados em diferentes níveis que não necessariamente fazem parte do "cenário geral da API da Web", como os vocabulários RDF. Nesse caso, pode ser útil adicionar conceitos adicionais, para que esses vocabulários possam se tornar parte do cenário da API da Web que pode ser rastreado.

Os conceitos da Web facilitam isso ao tratar os conceitos e as especificações como configuráveis. Isso significa que basta adicionar um novo conceito (como "vocabulários RDF") e, em seguida, começar a adicionar valores que identificam vocabulários específicos. Esses valores podem ser identificados e definidos por documentos em nível de organização, acrescentando ao ponto de partida de organizações de padrões que são cobertas pelo atual conjunto de dados de conceitos da Web.

Outro exemplo pode ser um conceito como Kafka topics2 . Em organizações que usam a plataforma de streaming distribuída Kafka, os tópicos são uma parte essencial de como as mensagens são produzidas e consumidas. Em ou No entanto, o próprio Kafka não oferece nenhuma facilidade para definir ou gerenciar tópicos, ele simplesmente organizará a distribuição de mensagens com base nos tópicos de mensagens de produtores e consumidores. Uma organização pode decidir transformar os tópicos Kafka em um conceito adicional da Web, certificando-se de que haja um local onde os tópicos possam ser registrados quando forem definidos e possam ser consultados para saber quais tópicos de mensagem estão sendo usados.

2Kafka é uma plataforma de software de processamento de fluxo de código aberto, onde as mensagens são gerenciadas de maneira PubSub e o consumo de mensagens é baseado em rótulos de mensagens chamados tópicos.

8 ACOMPANHANDO A EVOLUÇÃO DO PAISAGEM DA API

Um nível ainda mais alto de sofisticação pode ser alcançado com o Web Concepts, não apenas rastreando conceitos e valores, mas também rastreando o uso desses valores nas APIs. Isso equivaleria à capacidade de responder à pergunta "quais de nossas APIs estão potencialmente retornando o código de status HTTP 451?"

Essas informações podem ser rastreadas por todas as APIs de uma organização, reivindicando quais conceitos da Web estão usando e, em seguida, tendo algum processo para consolidar essas informações para resultar em informações de uso em todas as APIs. Uma maneira possível de fazer isso seria um processo descentralizado rastrear informações de uso de documentos domésticos [7] publicados por APIs individuais.

Esse processo pode até gerar dados históricos para representar tendências no cenário de APIs, como dizer quais padrões tiveram uma adoção crescente e decrescente nas APIs. Essas informações seriam úteis para equipes de desenvolvedores que procuram informações sobre quais padrões considerar ao projetar sua API: os populares podem ser bons candidatos, enquanto os que estão em declínio podem não ser mais as melhores opções de design. Conforme mencionado anteriormente, no final, isso criaria uma imagem dinâmica de quais blocos de construção estão sendo usados nas APIs de uma organização e como essa visão muda com o tempo. Esse nível de percepção pode ser uma perspectiva interessante das práticas recomendadas em constante evolução de como projetar APIs da Web.

9 TRABALHO RELACIONADO

Web Concepts é antes de mais nada um recurso onde certos conceitos e valores acordados para eles podem ser rastreados. Começou como uma tentativa de rastrear conceitos e padrões e ser capaz de fazer isso em um cenário diversificado de organizações que definem conceitos e valores. Embora a ideia em si não seja única, sua cobertura e extensibilidade são.

Para escopos diferentes, existem outros recursos. Como o recurso mais antigo e muito rico, existe a lista de "Registros de protocolo"3 da IANA , que em parte se sobrepõe aos conceitos da Web. No entanto, há o problema antigo de esses dados não serem expostos em formato legível por máquina e o problema adicional de cobrir apenas valores definidos ou registrados pela IANA. Ele também não possui o modelo de dados mais rico do Web Concepts com descrições para todos os valores e links para a fonte confiável.

Esforços mais direcionados também existem. Existe um repositório HTTP, que mantém listas de codificação, métodos, códigos de status, GitHub *hemi4 chamado "Conheça seu como tipos de mídia e relações de link. Embora tenha o mesmo objetivo de disponibilizar essas informações para fácil consumo como dados estruturados, é menos abrangente nos conceitos que abrange.

É provável que existam outros recursos com objetivos e abordagens semelhantes, mas, pelo melhor conhecimento deste autor, nenhum recurso se aproxima em cobertura de conceitos, variedade de fontes das quais os valores são coletados e abertura em termos de capacidade de estender o dados facilmente com novos valores ou conceitos.

3https://www.iana.org/protocols
4https://github.com/for-GET/know-your-http-well

10 GOVERNANÇA

Uma das questões importantes é como o conjunto de especificações será gerenciado. Provavelmente existem algumas respostas fáceis para novas especificações publicadas por organizações como IETF ou W3C. Eles devem ser adicionados ao repositório principal, e o processo atual é fazer isso por meio de PRs e possivelmente discussões de acompanhamento .

Para padrões definidos por organizações menores, eles podem não ser tão visíveis e amplamente usados, mas, desde que sejam adicionados ao respectivo registro da IANA, eles também são bons candidatos para inclusão. Se eles não estiverem registrados na IANA, ainda pode ser valioso incluí-los, desde que haja uma especificação publicada que possa ser referenciada e consultada por todos.

Para valores que são usados, mas não registrados nem definidos em uma especificação, as coisas ficam mais sutis. Por exemplo, muitas APIs da Web usam campos de cabeçalho HTTP que só eles usam e, embora seja bem conhecido que eles usam esses campos, muitas vezes não há uma especificação real para eles. A estrutura atual do Web Concepts não permite diferentes níveis de autoridade, então eles teriam que ser adicionados como especificações ou teriam que ser deixados de fora. Pode ser interessante adicionar tal diferenciação ao Web Concepts, mas como um sistema binário, é provável que se torne problemático em alguns casos.

Por fim, seguindo o modelo git geral de gerenciamento de dados, o repositório do Web Concepts sempre pode ser clonado e ajustado conforme necessário (conforme discutido na Seção 7). Esses clones podem ficar totalmente desconectados ou seus proprietários podem optar por obter atualizações seletivamente do upstream e, possivelmente, enviar suas atualizações locais para o upstream.

De um modo geral, o Web Concepts pretende ser um dado aberto e é publicado sob o título “The Unlicense”5 . Embora o repositório seja atualmente de propriedade privada, ele pode ser facilmente bifurcado por qualquer pessoa e para qualquer lugar e, com bastante interesse, pode ser facilmente transformado em uma organização GitHub (que é gratuita para código aberto).

11 CONCLUSÕES

Este artigo descreve os conceitos da Web e como eles podem ser usados como ponto de partida para desenvolvedores que procuram entender quais blocos de construção são populares quando se trata de navegar na API da Web. A ideia do Web Concepts é fornecer um conjunto aberto e abertamente reutilizável de blocos de construção, que podem ser facilmente personalizados para representar o modelo de uma organização de quando a interface uniforme da Web é tudo .

Essa visão pode ajudar provedores e consumidores de APIs da Web, permitindo que eles tenham uma visão mais clara da linguagem de design das APIs e certifique-se de que, dentro de uma organização, a reutilização dos blocos de construção da API se torne uma prática estabelecida e bem suportada.

REFERÊNCIAS [1] Tim

Berners-Lee, Roy Thomas Fielding e Larry Masinter. Identificador Uniforme de Recursos (URI): Sintaxe Genérica. Internet RFC 3986, janeiro de 2005.
[2] Tim Bray. O formato de intercâmbio de dados JavaScript Object Notation (JSON). Internet RFC 8259, dezembro de 2017.
[3] Tim Bray, Jean Paoli, C. Michael Sperberg-McQueen, Eve Maler e François Yergeau. Extensible Markup Language (XML) 1.0 (Quinta Edição). World Wide Web Consortium, Recomendação REC-xml-20081126, novembro de 2008.
[4] Richard Cyganiak, David Wood e Markus Lanthaler. RDF 1.1 Conceitos e Sintaxe Abstrata. World Wide Web Consortium, Recomendação REC rdf11-concepts-20140225, fevereiro de 2014.
[5] Roy Thomas Fielding, Mark Nottingham e Julian F. Reschke. Protocolo de transferência de hipertexto (HTTP/1.1): Cache. Internet RFC 7234, junho de 2014.
[6] Roy Thomas Fielding e Julian F. Reschke. Protocolo de Transferência de Hipertexto (HTTP/1.1): Sintaxe e Roteamento de Mensagens. Internet RFC 7230, junho de 2014.
[7] Mark Nottingham. Início Documentos para APIs HTTP. Rascunho da Internet nottingham-json-home-06, agosto de 2017.

5http://unlicense.org/