Unidade 11 - Banco de dados e JSON

Apresentação

JSON é o formato utilizado para intercambiar dados que mais cresce na atualidade e, por conseguinte, grande parte dos dados intercambiados, em algum momento, serão armazenados em bancos de dados, por meio de SGBD (Sistemas Gerenciadores de Banco de Dados). Os bancos de dados, por sua vez, colaboram na persistência dos dados que são intercambiados no formato JSON, o que colabora diretamente para o processamento de dados em diversos sistemas distribuídos e sistemas *web*.

Nesta Unidade de Aprendizagem, você irá aprender conceitos sobre os tipos de dados JSON presentes em um SGBD, além de implementar uma base de dados em JSON e, por fim, integrar os elementos JSON utilizando MySQL.

Bons estudos.

Ao final desta Unidade de Aprendizagem, você deve apresentar os seguintes aprendizados:

- Identificar o tipo de dados JSON em SGBD.
- Implementar uma base de dados com JSON.
- Integrar elementos JSON em MySQL.

Infográfico

A utilização do JSON para intercambiar dados de páginas *web* para bancos de dados é uma das principais funções do tráfego de dados na *web*, sobretudo pelo preenchimento de formulários eletrônicos, que por diversas vezes empregam JSON, dada a facilidade de orquestração de sua sintaxe.

No Infográfico a seguir, você vai ver a dinâmica do fluxo de intercâmbio de dados na *web* por meio do JSON, utilizando formulários de preenchimento *web*, até a conversão das informações preenchidas em dados nos bancos de dados distribuídos.

Conteúdo interativo disponível na plataforma de ensino!

Conteúdo do Livro

JSON é utilizado para intercambiar dados de forma mais simples, em contraponto ao XML. Os dados são o fator principal pelo qual se utiliza JSON. O mesmo vale para os SGBDs: a principal razão para utilizá-los é porque existem dados. Os dados, então, são a razão tanto da utilização do JSON, quanto dos SGBDs.

No capítulo Banco de dados e JSON, da obra *Programação back end III*, base teórica desta Unidade de Aprendizagem, você vai estudar os dados da perspectiva tanto do JSON quanto dos bancos de dados. Os dados são as estruturas fundamentais sobre as quais um sistema de informação é construído.

Boa leitura.

PROGRAMAÇÃO BACK END III

Pedro Henrique Chagas Freitas

SAGAH
SOLUÇÕES EDUCACIONAIS INTEGRADAS

Banco de dados e JSON

Objetivos de aprendizagem

Ao final deste texto, você deverá apresentar os seguintes aprendizados:

- Identificar os tipos de dados JSON em SGBD.
- Implementar uma base de dados em JSON.
- Integrar elementos JSON em MySQL.

Introdução

Neste capítulo, abordaremos os tipos de dados JSON presentes em um sistema gerenciador de banco de dados (SGBD), além de implementar uma base de dados em JSON. Por fim, integraremos os elementos JSON utilizando MySQL.

Tipos de dados JSON em SGBD

JSON (*JavaScript object notation*) é utilizado para intercambiar dados de forma mais simples que o XML. Logo, os dados são o principal motivo pelo qual utilizamos JSON. Da mesma forma, a principal razão para existirem SGBDs é por existem dados. Portanto, os dados são a razão tanto da utilização do JSON quanto dos SGBDs. Em suma, o dado é a estrutura fundamental sobre a qual um sistema de informação é construído.

Os dados são armazenados em bancos de dados, por meio de sistemas que irão gerenciar e orquestrar, em tabelas, os dados que serão armazenados. Estes sistemas são os SGBDs. Um banco de dados é uma coleção de dados relacionados, sendo que estes podem ser definidos como fatos, que possuem um significado implícito. A Figura 1 mostra os fatos de uma empresa reunidos como dados em um banco de dados.

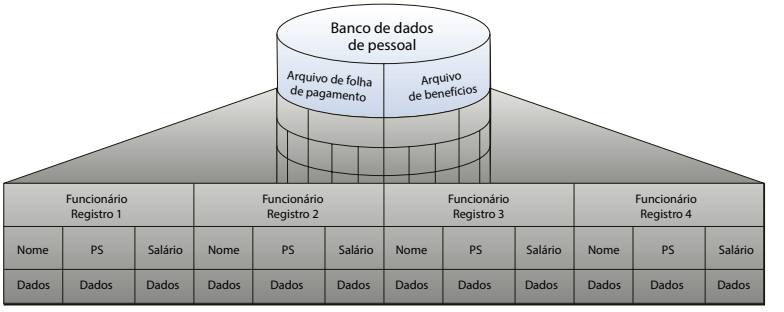

Figura 1. Fatos reunidos em um banco de dados.

Fonte: Adaptada de O'Brien (2004).

O que caracteriza um banco de dados é a forma como os dados estão interrelacionados, representando um universo de discurso, que apresenta um significado relevante a um grupo de usuários. Logo, podemos concluir que um SGBD deverá gerenciar uma coleção de dados interrelacionados, armazenados de forma centralizada ou distribuída, com algum significado inerente, isto é, informações de interesse de uma ou mais organizações.

Um banco de dados pode ser criado e mantido por um conjunto de aplicações desenvolvidas especialmente para esta tarefa ou por um SGBD. Um SGBD é um *software* (conjunto de programas) de caráter geral, que executa os processos de definição, construção, manipulação e compartilhamento de bancos de dados entre vários usuários e aplicações, incluindo módulos para consulta, atualização e as interfaces entre o sistema e o usuário.

Os bancos de dados também podem ser classificados de modo a refletir até que grau seus dados são estruturados e, neste contexto, encaixa-se nosso formato de persistência de dados, o JSON. Temos três tipos de dados, que visam refletir sua estrutura em um banco de dados, a partir do formato destes, como definidos a seguir.

- **Tipos de dados não estruturados:** aqueles que existem em seu estado original (bruto), ou seja, no formato em que foram coletados. Portanto, estão em um formato que não possibilita o processamento, o que produz informações.

- **Tipos de dados estruturados:** resultado da obtenção de dados não estruturados e de sua formatação (estruturação), visando facilitar o armazenamento, a utilização e a geração de informações. A estrutura (formato) é aplicada com base no tipo de processamento que se deseja executar nos dados.

Saiba mais

Alguns dados podem não estar prontos (não estruturados) para determinados tipos de processamento, mas podem estar prontos (estruturados) para outros tipos.

Por exemplo, o valor de dados 37890 pode se referir a um CEP, um valor de vendas ou um código de produto. Se representar um CEP ou um código de produto e for armazenado como texto, não será possível executar cálculos matemáticos com ele. Por outro lado, se esse valor representar uma transação de vendas, será necessário formatá-lo como numérico.

- **Tipos de dados semiestruturados:** aqueles que já foram parcialmente processados. Por exemplo, ao olharmos uma página comum da Web, os dados são apresentados em um formato pré-organizado para transmitir uma informação.

Os tipos de dados mencionados focam no armazenamento e gerenciamento de dados, segundo a perspectiva de estrutura, que é o principal no intercâmbio de dados em JSON, assim como em XML. É importante destacar que várias corporações não se limitam ao uso de dados estruturados, pois estas também utilizam dados semiestruturados e não estruturados. Basta pensar nas informações valiosas que podem ser encontradas em *e-mails* de empresas, memorandos, documentos como procedimentos e regras, conteúdos de páginas da Web, e assim por diante.

Atualmente, há uma nova tendência em que as necessidades de armazenamento e gerenciamento de dados não estruturados e semiestruturados estão sendo atendidas pela nova geração de bancos de dados em JSON e em XML.

A *extensible markup language* (XML), ou "linguagem de marcação extensível", é uma linguagem especial, utilizada para representar e manipular elementos de dados em formato textual. Os bancos de dados em XML dão suporte ao armazenamento e gerenciamento de dados semiestruturados em

XML. No caso do JSON, temos uma imensa vantagem, dada a sua simplicidade. Este pode ser utilizado para intercambiar dados dos três diferentes tipos de dados, para qualquer tipo de banco de dados, não necessariamente precisando ser um banco de dados em JSON.

Essa vantagem é fundamental, pois ajudou e ainda ajuda a dar preferência a utilização do JSON ao invés do XML, uma vez que a maioria dos dados que trafegam hoje na internet serão armazenados em bancos de dados em algum momento. É importante ressaltar, também, que o JSON ganhou muito espaço nos sistemas Web por trabalhar de forma mais simples, sem burocracia, além de integrado ao JavaScript.

Entre suas principais vantagens, destacamos que o JSON:

- possui *parsing* facilitado;
- suporta classes e objetos;
- é mais rápido na execução e no transporte de dados;
- cria arquivos menores;
- não é uma linguagem de marcação, nem um esquema de validação de dados;
- é, atualmente, utilizado por grande parte das empresas que trabalham diretamente com dados, como Google, Facebook, Twitter etc.

Implementando uma base de dados em JSON

Conforme verificamos, as bases de dados são repositórios que visam armazenar dados em uma ordem lógica, para posterior acesso, consulta, edição etc. Já o JSON é um formato de intercâmbio de dados que, em nosso caso, visará intercambiar dados de um sistema para um banco de dados.

Feita essa divisão, é importante ressaltar que uma base de dados é dividida em três dimensões ou níveis. Cada um desses níveis irá cooperar de forma integrada para persistir os dados intercambiados pelo formato JSON. Logo, temos que o grande objetivo de uma base de dados é prover ao usuário uma visão abstrata destes. Dessa forma, o sistema omite certos detalhes de como os dados são armazenados e mantidos. No entanto, para que o sistema possa ser utilizado, os dados devem ser armazenados e buscados. Em nosso caso, esse tráfego ocorrerá pelo JSON nos três níveis de abstração.

Cada um destes níveis corresponderá às abstrações dos dados armazenados na base de dados. Os três níveis são:

- nível de visão do usuário (externo);
- nível lógico (conceitual);
- nível físico (interno).

Teremos, então, as seguintes funções, conforme o nível de abstração da nossa base de dados.

- Nível de visão do usuário (externo) — nível mais alto de abstração, que descreve partes da base de dados, de acordo com as necessidades individuais de cada usuário. Em outras palavras, descreve o modo como os dados são vistos pelos usuários do SGBD.
- Nível lógico (conceitual) — escreve quais dados estão armazenados e seus relacionamentos. Neste nível, a base de dados é descrita por estruturas relativamente simples, que podem envolver estruturas complexas no nível físico.
- Nível físico (interno) — nível mais baixo de abstração. Descreve como os dados estão realmente armazenados, englobando estruturas complexas de baixo nível, que são descritas em detalhe.

A Figura 2 ilustra a arquitetura de três níveis para uma base de dados em JSON.

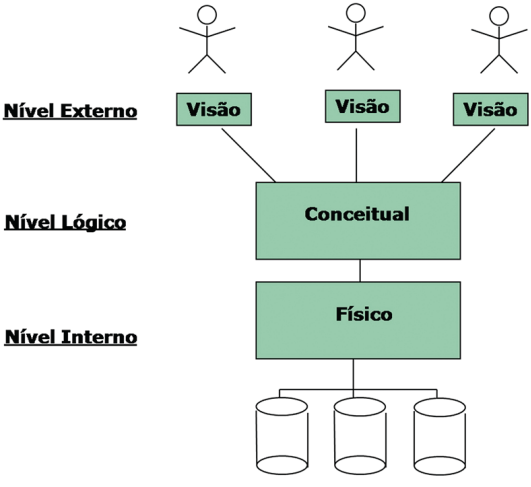

Figura 2. Arquitetura de três níveis para uma base de dados em JSON.

Fonte: Sousa (2013, documento *on-line*).

Como o JSON realizar o intercâmbio de dados, nosso objetivo será inserir dados em uma base de dados pelo JSON. Os dados serão de nomes e datas de acesso, com horário de entrada e saída de um sistema de ponto eletrônico.

Uma das formas de se implementar é criando um *array* em JSON para a inserção automática dos nomes e datas de acesso, com respectivos horários de entrada e saída.

```
[
    {
      "nome": "João Pedro Lisboa Filho",
       "array": [
       "dataentrada": "2019-09-26 09:30",
       "datasaida": "2019-09-26 18:30", ]
    },
[
    {
      "nome": "Carlos Eduardo Silva Castro",
       "array": [
       "dataentrada": "2019-09-26 09:00",
       "datasaida": "2019-09-26 18:00", ]
    },
[
    {
      "nome": "Rodrigo Pereira Araújo Sobrinho",
       "array": [
       "dataentrada": "2019-09-26 08:33",
       "datasaida": "2019-09-26 17:33", ]
    },
[
    {
      "nome": "Diego Henrique Gaspari Maria",
       "array": [
       "dataentrada": "2019-09-26 08:10",
       "datasaida": "2019-09-26 17:08", ]
    },
[
    {
      "nome": "Paulo Junior Oliveira Fernandes",
       "array": [
```

```
        "dataentrada": "2019-09-26 08:00",
        "datasaida": "2019-09-26 17:00", ]
    }
]
```

Teremos, então, a seguinte base de dados, representada na Figura 3, implementada por nosso *array* em JSON.

	nome	dataentrada	datasaida
2	João Pedro Lisboa Filho	26/09/2019 09:30	26/09/2019 18:30
3	Carlos Eduardo Silva Castro	26/09/2019 09:00	26/09/2019 18:00
4	Rodrigo Pereira Araújo Sobrinho	26/09/2019 08:33	26/09/2019 17:33
5	Diego Henrique Gaspari Maria	26/09/2019 08:10	26/09/2019 17:08
6	Paulo Junior Oliveira Fernandes	26/09/2019 08:00	26/09/2019 17:00

Figura 3. Base de dados criada pelo *array* em JSON.

Integrar elementos JSON em MySQL

Para demonstrar a integração dos elementos JSON em MySQL vamos, inicialmente, apresentar o funcionamento da linguagem do SGBD MySQL, a fim de melhor contextualizar a integração que faremos, que consistirá na inserção de dados pelo JSON no banco de dados MySQL. Em razão disso, faremos a consulta dos dados armazenados pela linguagem SQL, oriunda do banco MySQL.

Como faremos o intercâmbio de dados pelo JSON e a consulta dos dados pela SQL, é importante compreendermos algumas estruturas desta linguagem.

A *structured query language* (SQL) é uma linguagem que pode ser empregada no MySQL, bem como em outros SGBDs. Basicamente, a SQL serve para a realização de diferentes tarefas repetitivas no banco, aceitando parâmetros de entrada e retornando valores.

Para isso, utilizamos como instrumento as *trigger* do SQL, que são sub-rotinas, parecidas com uma *stored procedure*, que tem como característica ser executada automaticamente a partir de alguma ação realizada no banco de dados. Geralmente, são utilizadas com um tipo de proteção ao acesso indiscriminado a dados de uma tabela. Quando há uma tentativa de inserir,

atualizar ou excluir os dados de uma tabela, e uma *trigger* tiver sido definida na tabela para essa ação específica, ela será executada automaticamente, não podendo nunca ser ignorada.

Não é necessário realizar a integração toda vez que o dado intercambiado em JSON for acessado. Logo, é natural que utilizemos *triggers* do SQL ou sub-rotinas.

A título de exemplo, temos a *trigger* (sub-rotina) a seguir, que tem como objetivo realizar a atualização salarial de um empregado. Neste caso, a *trigger* vai até a tabela Departamento e atualiza o total de despesas do referido Departamento com o novo salário do empregado.

```
CREATE TRIGGER TotalSal2
AFTER UPDATE OF Salario ON Empregado
FOR EACH ROW
WHEN (NEW.Ndep IS NOT NULL)
UPDATE Departamento
SET TotalSal=TotalSal+NEW.Salario-OLD.Salario
WHERE Dnum=NEW.Ndep;
```

Na integração JSON e MySQL, vamos usar *triggers* para realizar as sub-rotinas. Utilizaremos um *array* para inserção de dados via JSON em nosso SGBD MySQL e, em seguida, faremos as consultas pela SQL, conforme ilustra a Figura 4.

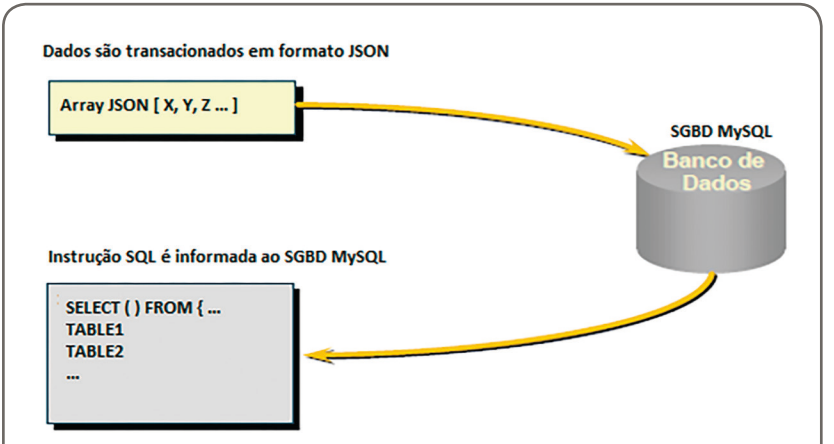

Figura 4. Esquema de consulta JSON–SQL.

Inicialmente, criaremos nossas tabelas no banco de dados pela linguagem SQL, com a sintaxe a seguir.

```
CREATE TABLE nome_da_tabela
(Nome_da_coluna1 tipo_do_dado (tamanho_do_dado)
Nome_da_coluna2 tipo_do_dado (tamanho_do_dado)
...
Nome_da_colunaN tipo_do_dado (tamanho_do_dado));
```

Então, utilizaremos uma *trigger* `Cliente` e faremos nosso `arrayJSON` para inserir os números telefônicos de nossos clientes.

```
CREATE TRIGGER Cliente
AFTER INSERT OF nome ON cliente
AFTER INSERT OF telefone ON cliente
UPDATE cliente;

[
arrayJSON_Cliente:
    {
      "nome": "Carlos Eduardo",
       "array": [
       "telefone": 61984567834, ]
    },
[
    {
      "nome": "Carla Menezes",
       "array": [
       "telefone": 65987564321 ]
    },
[
    {
      "nome": "Rodrigo Caio",
       "array": [
       "telefone": 63998671243, ]
    },
[
    {
      "nome": "Vitor Hugo",
```

```
        "array": [
        "telefone": 31999723457, ]
    },

]
]
```

Automaticamente, teremos uma tabela, contendo nome e telefone, como apresentado na Figura 5.

ID	nome	telefone
1	Carlos Eduardo	61984567834
2	Carla Menezes	65987564321
3	Rodrigo Caio	63998671243
4	Vitor Hugo	31999723457

Figura 5. Resultado do código para a tabela com nome e telefone.

Conforme vamos incrementado nosso *array*, novos elementos irão compondo nossa lista de clientes e telefones.

Agora, faremos uma consulta que retornará o nome e o telefone do `ID 3` e do `ID 1` que, no caso, são os clientes `Rodrigo Caio` e `Carlos Eduardo`, como mostra a Figura 6.

```
Select distinct cliente.nome, cliente.telefone from from ID 3;
Select distinct cliente.nome, cliente.telefone from from ID 1;
```

3	Rodrigo Caio	63998671243
1	Carlos Eduardo	61984567834

Figura 6. Resultado de nossa integração de dados.

Por fim, é fundamental destacarmos que uma mesma integração poderia ser realizada com o XML, mas optamos pelo JSON pelo fato de o XML ser mais rígido, o que dificulta o intercâmbio de dados e, por consequência, a integração com bases de dados.

Referências

O'BRIEN, J. A. *Sistemas de informação e as decisões gerenciais na era da Internet*. 2. ed. São Paulo: Saraiva, 2004. 433 p.

SOUSA, M. H. Níveis de Abstração em Banco de Dados - Arquitetura ANSI/SPARC. *M. H. Sousa*, [*S. l.*], 19 mar. 2013. Disponível em: http://mhsousa2013.blogspot.com/2013/03/niveis-de-abstracao-em-banco-de-dados.html. Acesso em: 12 out. 2019.

Leituras recomendadas

PRESSMAN, R. S. *Engenharia de software*: uma abordagem profissional. 7. ed. Porto Alegre: AMGH; Bookman, 2011. 780 p.

SOMMERVILLE, I. *Engenharia de software*. 8. ed. São Paulo: Pearson Prentice Hall, 2008. 552 p.

TONSIG, S. L. *Engenharia de software*: análise e projeto de sistemas. 2. ed. Rio de Janeiro: Ciência Moderna, 2008. 319 p.

Encerra aqui o trecho do livro disponibilizado para esta Unidade de Aprendizagem. Na Biblioteca Virtual da Instituição, você encontra a obra na íntegra.

Conteúdo:

SAGAH
SOLUÇÕES EDUCACIONAIS INTEGRADAS

Dica do Professor

POJO são classes contendo atributos e métodos GET e SET. São bastante difundidas dentro do JSON e têm inúmeras possibilidades de utilização. Pode-se utilizar POJO em JSON para simular a persistência de dados em banco de dados.

Nesta Dica do Professor, você irá ver um exemplo envolvendo uma classe Carro para demonstrar a utilização do conceito por trás da persistência de dados, usando POJO e JSON em banco de dados.

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

Exercícios

1) **Os dados são armazenados em bancos de dados, por sistemas que irão gerenciá-los e orquestrá-los em tabelas. Tais sistemas são os gerenciadores de bancos de dados, ou SGBDs. Existem três tipos de dados intercambiados por meio da formatação JSON que visam a refletir a estrutura dos dados em um banco de dados. São eles:**

A) Normalizado, seminormalizado e desnormalizado.

B) Não estruturado, semiestruturado e estruturado.

C) Pré-empacotado, empacotado e desempacotado.

D) Abstrato, concreto e híbrido.

E) Desnormalizado, estruturado e abstrato.

2) **Em uma página comum da *web*, os dados são apresentados em um formato pré-organizado para transmitir alguma informação. Eles podem vir a ser intercambiados para bancos de dados, por meio do formato JSON. Esses dados, comuns em páginas *web*, já foram parcialmente processados e são conhecidos como:**

A) abstratos.

B) empacotados.

C) normalizados.

D) semiestruturados.

E) concretos.

3) **JSON ganhou espaço nos sistemas *web* por trabalhar de forma mais simples, sem burocracia, e, integrado a JavaScript, possui *parsing* mais fácil; suporta classes e objetos; é mais rápido na execução e transporte de dados; cria arquivos menores; não é uma linguagem de marcação nem um esquema de validação de dados, e tem como principal concorrente, no intercâmbio de dados para banco de dados, a linguagem:**

A) XML.

B) HTML.

C) CSS.

D) DHTML.

E) XSLT.

4) Existem três níveis em bases de dados, cada um correspondendo às abstrações dos dados armazenados. Os três níveis colaboram com a persistência, na base de dados, dos dados recebidos no formato JSON. Respectivamente, são eles:

A) nível lógico, nível conceitual e nível de usuário.

B) nível externo, nível físico e nível interno.

C) nível conceitual, nível abstrato e nível concreto.

D) nível conceitual, nível abstrato e nível lógico.

E) nível externo, nível conceitual e nível interno.

5) A integração dos elementos JSON em um SGBD MySQL apresenta o intercâmbio de dados por meio do formato JSON, enquanto o banco de dados realizará a consulta aos dados JSON por meio da linguagem:

A) HTTP.

B) PHP.

C) SQL.

D) HTML.

E) JavaScript.

Na prática

Na prática, JSON é usado por diversas aplicações comerciais em muitas empresas, principalmente para permitir a troca de dados entre linguagens, sistemas legados e tecnologias diferentes, antes que os dados trafegados sejam armazenados em banco de dados. Empresas de arquitetura e integração de dados, por exemplo, costumam utilizar muito a sintaxe JSON.

Conheça, a seguir, algumas ferramentas para utilização do JSON.

{ FERRAMENTAS PARA UTILIZAÇÃO DO JSON }

Isso é muito importante, pois diversas tecnologias utilizam *webservices* para intercomunicação entre plataformas e, por conseguinte, fazem uso do JSON para interoperar dados, antes de armazená-los em bases de dados.

Exemplos de empresas que utilizam intercomunicação entre plataformas, por meio de JSON, são aquelas que trabalham com soluções de *Big Data* e que manipulam grandes massas de dados.

Hoje, devido ao uso intenso de webservices, algumas ferramentas foram criadas para apoiar a utilização do JSON, sendo adotadas pela maioria das empresas, tanto de *Big Data* quanto de arquitetura e integração de dados

Entre essas plataformas, destaca-se:

- JSON Parser Online
- JSONLint – The JSON Validator
- Online JSON Viewer
- Convert JSON Strings to a Friendly Readable
- Format
- JSON Generator

Caso você esteja em uma empresa de arquitetura e integração de dados, e venhaa desenvolver uma aplicação que utilize *webservice*, se optar por utilizar o formato JSON, provavelmente vai se deparar com alguma dessas ferramentas JSON.

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.