UPT UNIVERSIDADE PORTUCALENSE

CONTEÚDO

1. CRUD
2. Comportamento CRUD
3. Web Services
4. Application Programming Interface (API)
5. Representational State Transfer (REST)
6. Exemplos de chamada a API
    1. Do lado do cliente
    2. Do lado do servidor
7. Boas práticas

CRUD

- **C**reate, **R**ead, **U**pdate, **D**elete

- Utilização deste paradigma fundamental para a construção de aplicações robustas
    - Fornece estrutura bem definida e transversal
        - Todos os programadores conhecem os métodos disponíveis

- Corresponde, respetivamente, aos métodos: post, get, put e delete

- Utiliza um método HTTP através de uma rota específica
    - Faz determinada operação na base de dados

- Pode ser formalizado através de:
    - *Web Services*
    - *Application Programming Interfaces* (**API**)
    - *Representational State Transfer* (**REST**)

Comportamento CRUD

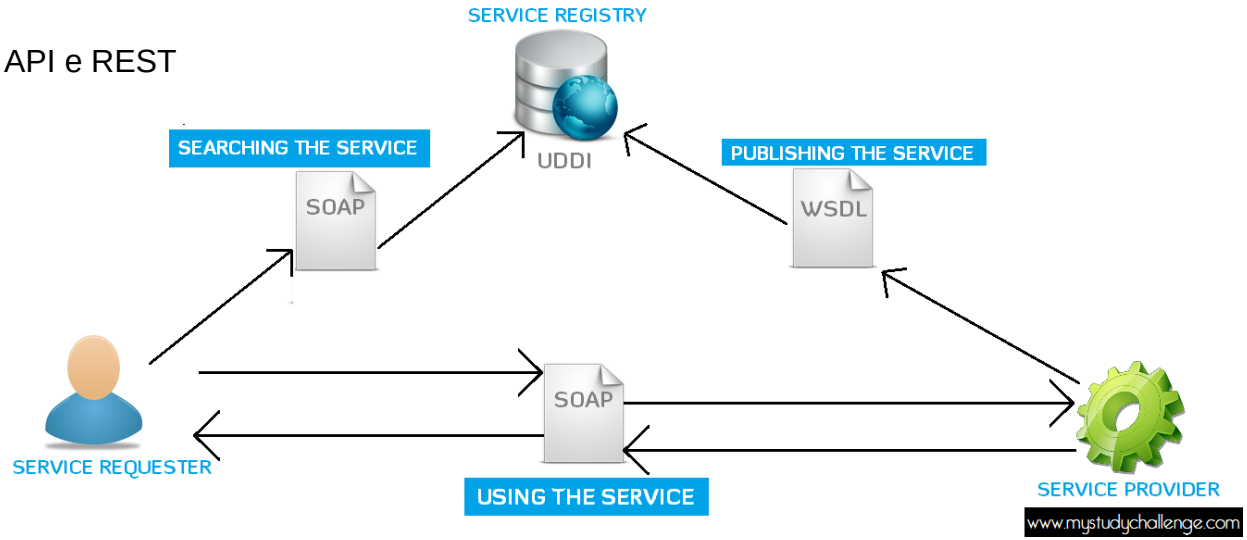
Pedido	Rota de Pedido	Operação na BD	Resposta	Cód.
Create	POST /data	Inserir dados enviados no pedido	{"success":"Registo Criado"}	201
Read (todos)	GET /data	Ler	{"success";[Array com os registos]}	200
Read (específicos)	Get /data:id	Ler	{"success";[Array com os registos do id]}	200
Update	PUT /classes/:id	Alterar dados enviados no pedido	{"success";"Os dados foram atualizados com sucesso"}	200
Delete	DELETE /classes/:id	Remover dados de um id	-	204

Web Services

- Permitem converter as soluções existentes em aplicações
    - E publicar as suas funcionalidades

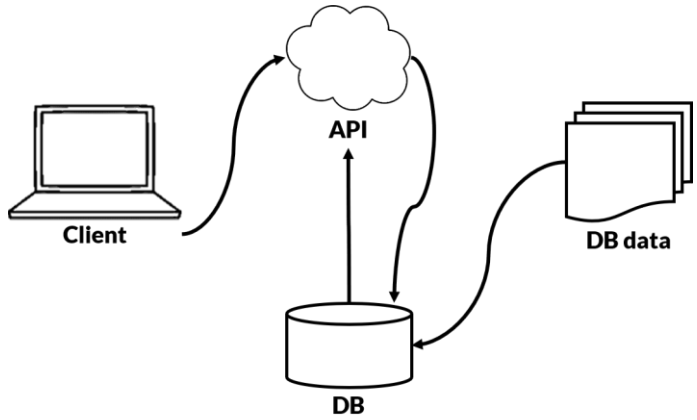
- Componentes de software que comunicam através de protocolos abertos, autocontidos, autodescritíveis
    - Descritos, publicados, descobertos e invocados via Web através de:
        - Mensagens XML
        - Protocolo *Simple Object Access Protocol* (**SOAP**)

- Substituídos por API e REST



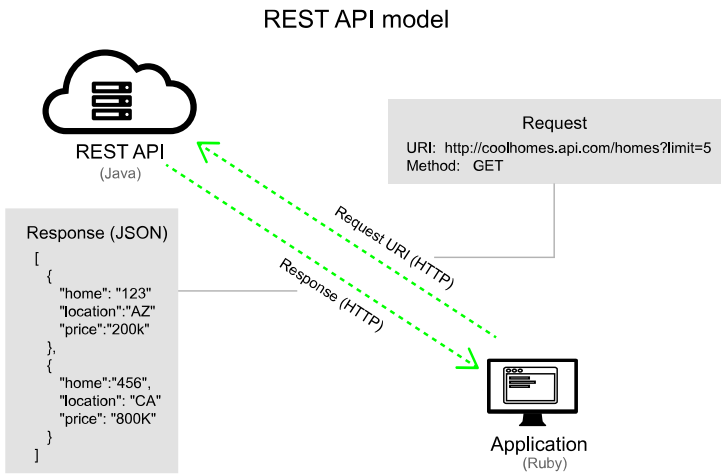
Application Programming Interface (API)

- Interface de programação de aplicações
- Conjunto de rotinas e padrões estabelecidos por um software
  - Conjunto de mensagens de pedido/resposta HTTP (XML ou JSON)
- Estendem funcionalidades de aplicações
  - Aplicações usam funcionalidades da API
    - Sem saber detalhes da implementação
- Interface máquina-máquina
- Utilização generalizada nos plugins



- Utilizada com REST traz benefícios:
  - Eficiência
  - Amplo alcance
  - Diversidade de aplicações
  - Gestão de processos
  - Automatização de procedimentos
  - Parcerias entre aplicações
  - Interoperabilidade e integração
  - personalização

Representational State Transfer (REST)



- Protocolo de comunicação que usa padrões entre sistemas Web
  - Para facilitar a comunicação
- Sistemas compatíveis com REST (RESTful)
  - Não têm estados (stateless)
  - Separam camadas do cliente e do servidor
    - Pressupõem que é separado
      - Código pode ser alterado a qualquer momento de um dos lados, sem afetar o outro
- Para que a comunicação seja eficiente é apenas preciso saber qual o formato das mensagens
- Pedido REST necessita:
  - Método HTTP (get, post, put, delete)
  - Cabeçalho
  - URL

Exemplo de chamada a API do lado do cliente

```
fetch(url, { //caminho para a API
  method: 'POST' //método HTTP
  })
  .then((resp) => resp.json()) // transforma os dados da resposta (resp) em JSON
  .then(function(data) {//data contém os resultados em formato JSON
     let users = data.results; // lê os dados presentes em data
     return users.map(function(user) { //para cada user
          console.log("Nome: " + user.name.first + " " + user.name.last);
  })
  .catch((err) => console.log(err))
```

Exemplo de chamada de API do lado do servidor

Utilização de API da Google com Node.js

1. Incluir o módulo `npm install googleapis`

2. Especificar a API e a versão:

```
var googleapis = require('googleapis');
var client;
googleapis.discover('plus','v1').execute(function(err,data){
   client = data;
});
```

3. Efetuar a chamada utilizando as credenciais da Google disponibilizadas na conta de programador:

```
var oauth2 = new googleapis.OAuthClient(CLIENT_ID, CLIENT_SECRET, 'postmessage');
oauth2.getToken(code, function(err,tokens){
   oauth2.credentials = tokens;
   client.plus.people.get({
      userId: 'me'
   }).withAuthClient(oauth2).execute(function(err,result){
   });
});
```

Boas práticas

- URL de pedidos não devem conter verbos nem o nome dos métodos. Exemplos:
  - get http://api.com/users para retornar uma lista com todos os users
  - get http://api.com/user para criar um novo user
  - post http://api.com/users/2 para atualizar informação (ou criar novo) de user com ID 2
  - delete http://api.com/users/2 para eliminar o user com ID 2

- Quando o objetivo é interagir com tabelas, enviar os pedidos preenchidos com dados que pretendemos adicionar
  - get http://api.com/evento/1/users para retornar a lista de users a participar no evento com ID 1
  - post http://api.com/evento/1/users/3 para associar o user 3 ao evento 1
  - put http://api.com/evento/1/users/3 para atualizar os dados do user 3 que está associado ao evento 1
  - delete http://api.com/evento/1/users/users/3 para eliminar o user 3 do evento 1

- Os pedidos devem conter apenas nomes
  - Tipo de pedido e método utilizado fazem a diferente operação na BD

UNIVERSIDADE
PORTUCALENSE

Do conhecimento à prática.