UPT UNIVERSIDADE PORTUCALENSE

CONTENT

1. CRUD
2. CRUD behaviour
3. Web Services
4. Application Programming Interface (API)
5. Representational State Transfer (REST)
6. API call examples
    1. Client side
    2. Server side
7. Best practices

CRUD

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

- The usage of this paradigm is essential to build robust applications
    - Provides a well-defined structure
        - All the programmers are aware of the methods available

- Corresponds, respectively, to the methods: post, get, put e delete

- Uses an HTTP method through a specific route
    - Makes a specific database operation

- May be:
    - *Web Services*
    - *Application Programming Interfaces* (**API**)
    - *Representational State Transfer* (**REST**)

CRUD behaviour

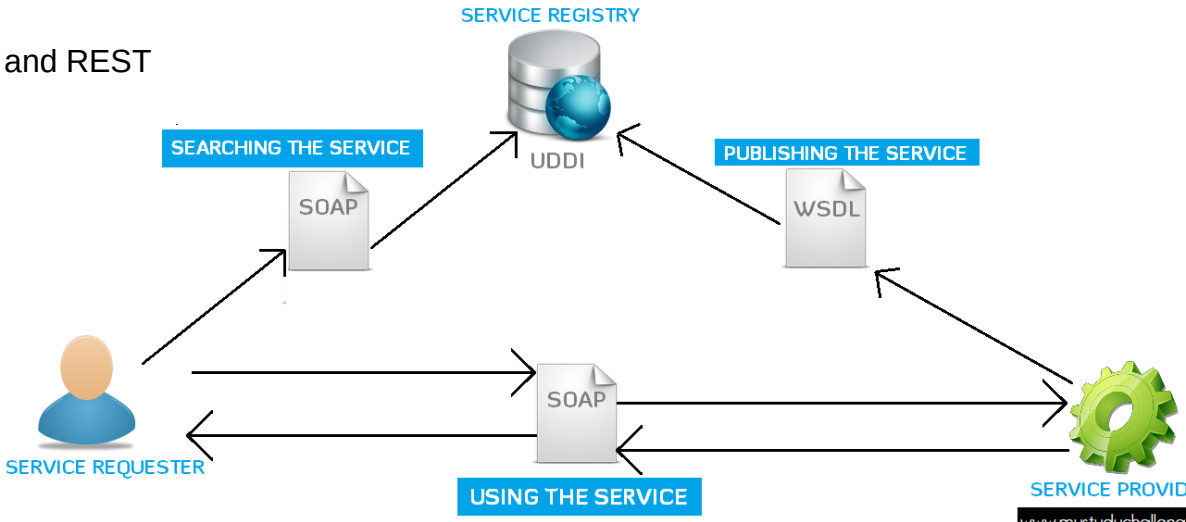
Request	Request Route	DB operation	Response	Code
Create	POST /data	Insert data sent on the request	{"success":"Registo Criado"}	201
Read (todos)	GET /data	Read	{"success";[Array com os registos]}	200
Read (específicos)	Get /data:id	Read	{"success";[Array com os registos do id]}	200
Update	PUT /classes/:id	Change data sent on the request	{"success";"Os dados foram atualizados com sucesso"}	200
Delete	DELETE /classes/:id	Remove data of an id	-	204

Web Services

- Allow converting existing solutions into applications
  - And publish their functionalities

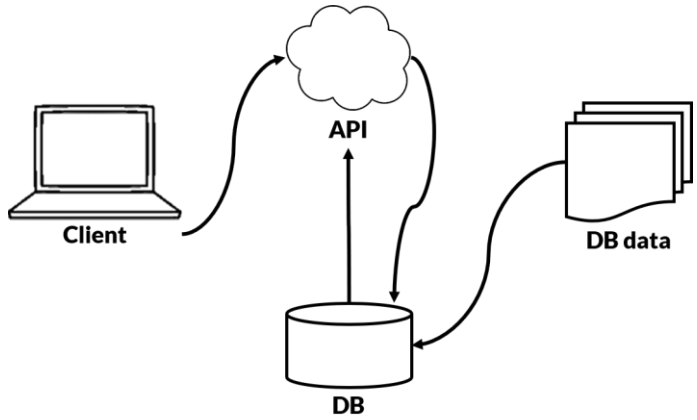
- Software components that communicate through open protocols
  - Described, published, discovered and invoked through:
    - XML messages
    - *Simple Object Access Protocol* (**SOAP**) protocol

- Replaced by API and REST



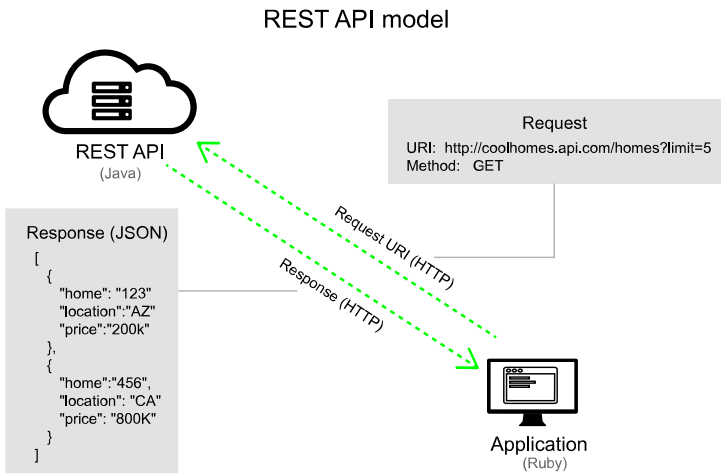
Application Programming Interface (API)

- Interface to program applications
- Set of routines and patterns defined by a software
  - Set of HTTP request/response (CML or JSON) messages
- Extend application functionalities
  - Applications use API functionalities
    - Without knowing implementation details
- Machine-machine Interface
- Generalized use in plugins



- Benefits when used with REST:
  - Efficiency
  - Broad range
  - Application diversity
  - Process management
  - Procedure automatization
  - Cooperation between applications
  - Interoperability and integration
  - Personalization

Representational State Transfer (REST)



- Communication protocol that uses patterns between web systems
  - To make the communication easyer
- REST compatible systems (RESTful)
  - No state (stateless)
  - Separate client and server layers
    - Assume the separation
      - Code can be changed any time in one side, without affecting the other
- We only need to know he message format so that the communication is efficient
- REST request needs:
  - HTTP method (get, post, put, delete)
  - Header
  - URL

API call example: client side

```
fetch(url, { // path to the API
  method: 'POST' // HTTP method
  })
  .then((resp) => resp.json()) // converts the response data (resp) into JSON
  .then(function(data) {// data contains the results in JSON formal
     let users = data.results; // reads the data
     return users.map(function(user) { // for each user
          console.log("Nome: " + user.name.first + " " + user.name.last);
  })
  .catch((err) => console.log(err))
```

API call example: server side

Using Google API with Node.js

1. Include module `npm install googleapis`

2. Specify the API and the version:

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

3. Make the call using google's credentials, available on the programmer account:

```
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){
   });
});
```

Best practices

- Request URLs should no contain verbs or method names. Examples:
  - get http://api.com/users to return a list with all users
  - get http://api.com/user to create a new user
  - post http://api.com/users/2 to update information (or create new) user with ID 2
  - delete http://api.com/users/2 to delete user with ID 2

- When the objective is to interactwith tables, send the requests with the data we wish to add:
  - get http://api.com/evento/1/users to return the list of users in event with ID 1
  - post http://api.com/evento/1/users/3 to insert user 3 in event 1
  - put http://api.com/evento/1/users/3 to update user 3's data on event 1
  - delete http://api.com/evento/1/users/users/3 to delete user 3 from event 1

- The requests must only contain names
  - The type of request and method used make different database operations

UNIVERSIDADE
PORTUCALENSE

Do conhecimento à prática.