Mahisoft, Caracas 01-14-2017

Teachers:

- Melian J.
- Fagundez M.

Student: Leonardo Santella

First steps in web development: T1

Theory:

1. **Rest**: REST is an architectural style. It's a short for REpresentational State Transfer. It provides standards for computer systems on the web, making easier the communication between them. There are six constraints, one of them is optional, which ensure that a system is RESTful. These constraints are the following:
   - **Uniform interface**
   - **Stateless**
   - **Cacheable**
   - **Client-server**
   - **Layered system**
   - **Code on demand** (optional)
2. **Rest resource**: A REST resource is an element of a RESTful system. It's transferred between different parts of the web as a representation. This mean that a person in a web server can be modeled as a (Name-Id) pair, however, it can be transferred as JSON object or a XML document.
3. **Http request**: As the REST architectural pattern (previously REST was defined by Joy Fielding as Web's architectural style) need to be client-server, the main role of the client is to make requests to the servers. An http request is a request that follows the HyperText Transfer Protocol. An HTTP request has the following structure:
   - Request line. This line corresponds to an HTTP method followed by an URI of the resource requested, the HTTP version and a CRLF.
   - Zero or more headers containing metadata
   - An empty line indicating the finish of the headers
   - An optional message It's defined by an URI (Universal Resource Identifier), an HTTP verb and also it can contain headers and a body. The header can contain metadata about the request and the body can contain the representation of a resource.
4. **Http response**: Similar to an HTTP request, an HTTP response is the way as a server respond to a client. An HTTP has the following structure:
   - Status code line. At the same time, this line has the following structure: HTTP version, status code, phrase reason and CRLF, separated by white spaces.
   - Zero or more headers
   - An empty line indicating that the headers lines are finished
   - An optional message
5. **Usage of Http codes with examples**: There are different types of status code for HTTP

responses. They are characterized by the first digit of the code as follows:
    - **1XX** Informational. It means to be a code to transmit that the request has been received successfully and the client can continue the data transfer to the server. This can be useful when there is a large request body and it can be transferred as chunks.
    - **2XX** Success. It means that a request has been processed successfully. These codes are used when a user makes a request to change the state of a resource on a system and the change happens successfully and then the server forward a 200 OK status code line to the client.
    - **3XX** Redirection. These codes means that there are actions needed to be performed in order to complete the request successfully. This can be useful when there's a system that's been updated and some URIs changes, the redirection can indicate that the resource requested has been changed and it can be found at some other URI.
    - **4XX** Client-side error. These kind of codes indicate that there's a client side error. A common use case of this is the 404 code. It notifies to the user that the URI requested is not found, thus, is a client error because it tried to access to a resource that has no URI.
    - **5XX** Server-side error: These codes indicate that there's a server-side error. A common use case for this codes is when a server tries to process the request (that was successfully received) and occurs an error, this is reported as 500.
6. **Rest vs alternatives**: There are many architectural patterns besides REST as Fielding compared in his thesis, some of them are the following:
    - **Data Flow**: Pipe and filter vs. Uniform Pipe and filter Replication: Replicated repository vs. Caches
    - **Data Access**: Client-Server vs. Layered System and Layered Client-Server vs. Client-Stateless-Server vs. Client-Cache-Stateless-Server vs. Layered Client-Cache-Stateless-System vs. Remote Session vs. Remote Data Access
    - **Code Mobility**: Virtual Machine vs. Remote Evaluation vs. Code-on-Demand vs. Layered-Code-on-Demand-Client-Cache-Stateless-Server vs. Mobile Agent
    - **Peer-to-Peer**: Event-based-Integration vs. C2 vs. Distributed Objects vs. Brokered Distributed Objects
7. **What are design patterns? Why are they used? Please provide examples**:
   A design pattern is a general, reusable solution to a commonly occurring problem within a given context in software design. Design patterns can speed up the development process by providing tested, proven development paradigms. Effective software design requires considering issues that may not become visible until later in the implementation. Reusing design patterns helps to prevent subtle issues that can cause major problems and improves code readability for coders and architects familiar with the patterns. There are different types of design patterns. Some of these types are the following: creational patterns, structural patterns, behavioral patterns, architectural patterns, and concurrency patterns. Some examples: MVC (architectural), Blockchain (concurrency), Singleton (creational), Decorator (structural), Iterator (behavioral).