REST is short for REpresentational State Transfer. Rest is an architecture to create restful services. Rest API must be stateless, which means server does not save the client data between the requests or remembering previous requests. Requests are sent through http protocol. Rest can be used by any programming languages. RESTful API uses HTTP requests to access and use data. Clients and servers use HTTP protocol to exchange the data. The methods it uses are GET, POST, PUT, DELETE, HEAD, OPTIONS, PATCH. It returns different status or response code to represent state of object on server. HTTP stands for Hypertext transfer protocol which is an application layer protocol.

For our solution we have focused on implementation of get, post, put and delete:

Where we have methods for getting list of doctors, departments, patients, and the symptoms related to patients.

For post operation we take in the body as a set up data class so that we have control over what is sent In, a post request sent in create a department would need to send in the department id, and the department name in json format to be able to create a department. In the backend we have some measures in place to make sure that the user can put in illegal departmentId, and still be able to create the department, where we provide an acceptable id, and return that id in the response

For put operation we have made it possible to update for example a patient, where we query the id of the object to be updated in the url, and take in the updated information in the Body

HATEOAS: When a REST request is performed. We will only get the data and not any related links to the resources. But with HATEOAS implemented we can specify related actions in form av URLS. So HATEOAS is generally used to create links to access resources which are created/updated/present on the server.

For the hateoas part we provide the user with potential paths, for each element they access from the api. For example when getting out a list of patients, we provide links to all the allowed paths that can be used for that patient, with relations as well as the methods needed(post, get, put)

Rest has architectural constraints which are: Uniform interface, Statelessness, Cacheable, Layered System, Separation of Client and Server.[1] Restful web services are built on the HTTP protocol and implement operations that map to the common HTTP methods.

[1] https://docs.oracle.com/middleware/1221/wls/RESTF/intro-restful-service.htm#RESTF107

To scale the servers, we intented to use statelessness and caching. Statelessness enables greater scalability since the server does not have to maintain, update, or communicate that session state.[2]

For caching we have not implemented this on the client side yet, and thus we would in a full deploy be wasting server resources on calls for elements that have not been updated since last call, and this would have been a valuable future addition to the project.

Sources:

https://www.restapitutorial.com/lessons/whatisrest.html#

https://blog.hubspot.com/website/what-is-rest-api

Teachers REST AND HATEOAS PowerPoint slides.

[2] https://www.restapitutorial.com/lessons/whatisrest.html#