Prog2005 exam

**REST**

REST is a collection of principles and methods on how to manage internet resources. When machine to machine communication was introduced and the internet information went from being available to the users only as consumers to now being able to publish their own information and communicate and became prosumers. This is where the need for REST services became apparent.

REST includes the methods POST, PATCH, GET, DELETE, HEAD, CONNECTION and some more that allow the users to interact with the resources provided by the systems and servers.

POST: Is a method to insert data into the server.

PATCH: The user can alter already existing data on a server, for example if there is a record with information about that person we could use the PATCH method to for example change the persons age.

DELETE: Method to delete data that exists in the server.

GET: Method to retrieve data that exists in the server and is the most used one.

HEAD: Only get the header information for the site, not the body. For example what the content type of the site is.

COONECTION: Status of the connection with the server.

These methods are used with URI’s (Uniform Resource Identifier) that are used to indicate where the resource that you interact with is located and looks like this for example:   
<http://www.example.com/users?id=123>

Here we have the http:// which is the scheme of the URI indicating the protocol to be used and then the host/domain and then the path which specifies where in the domain the resource to perform the method on is located.

Principles in a REST system is:   
Cache ability: Should support caching of for example frequently used resources for improving performance.

Client-server: Communication between the client and the server

Statelesness: The state should be kept on the client side, because the server should be able to handle a very large amount of requests from different clients and if the state of the communication between the server and the client was kept on the server side it would require way to much resources.

Code on demand:

Layered system: The system should be layered such that we can communicate with different servers based on what we are requesting such that not one server handles all the different requests.

Uniform interface: It should be possible to communicate with the system no matter for example what operating system you are using.

**CHATGPT ANSWER**

REST (Representational State Transfer) is an architectural style that emerged due to the evolving nature of the internet. It provides a set of principles and methods for managing and interacting with resources.

At its core, REST enables communication between machines by defining a standard set of methods for interacting with resources. These methods include:

GET: Retrieves the representation of a resource from a server. It is used to fetch content from a specific URL.

POST: Submits data to be processed by a server. It is commonly used for adding new content to a system or triggering server-side actions.

PATCH: Modifies a subset of data in an existing resource. It is used for making partial updates or modifications.

DELETE: Removes a resource from a server. It is used to delete specific content identified by a URL.

PUT: Completely replaces a resource's representation with a new one. It requires sending the entire representation of the resource.

RESTful services adhere to several principles:

Statelessness: Each client request is self-contained, and the server does not store any client-specific information between requests. This allows the server to handle a large number of requests from different clients without maintaining individual session state.

Cacheability: REST promotes the use of caching to enhance performance and reduce server load. Caching allows responses to be stored and reused when the same request is made in the future. This can significantly improve response times and reduce the load on the server.

Client-Server Communication: REST emphasizes the separation of concerns between the client and the server. The client is responsible for the user interface and user experience, while the server manages the resources and performs requested operations. This separation allows for scalability, modifiability, and independent evolution of client and server components.

Uniform Interface: A key aspect of REST is the use of uniform resource identifiers (URIs) to identify and locate resources within a RESTful system. URIs uniquely identify resources and provide a consistent way for clients to access and interact with them. Additionally, REST promotes a uniform set of methods (GET, POST, PATCH, DELETE, PUT) for interacting with resources, regardless of the underlying implementation or operating system.

**RFC**

RFC stands for (request for comments) and it contains documents about different standards for the different internet protocols that we use for example HTTP. The standards serve as guideline and reference that developers can use to make sure the system they are developing adheres to the standards. This ensures consistency and interoperability for the services on the internet.

Anyone can come with suggestions to new standards and then it will be open for the public to review and come with comments on the suggestion. Then the internet engineering taskforce will go through and review the suggest standard and it will go through many committees before it eventually might be accepted.

There are different categories for the RFC’s they include:

Informational RFC’s

Experimental RFC’s

Official RFC’s (those that are the standard)

Best practices

Historic RFC’s

**Request specifications:**

Content type for example: application/JSON, text/plain, image/h256

URL: URL for the request for example <http://example.com/rooms/specificRoom?id=2>

Where http:// = scheme, example.com = host, /rooms/specificRoom = path, ?id=2 = query

Status code: 100 = informational, 200 = ok, 300 = redirect, = 400 client error, 500 = server error.

Structure of request (body): in a POST request this would be the data you would want to add to the server.

**JSON**

Very flexible way of structuring data that allows for nested structures, several data types (lists, ints, strings…). Models objects (Java script object notation), such that in one {} contains an object. Stores the data in key value pairs such that each data field is represented by a key. It provides a human readable format for storing and reading data and most programming languages allow for collecting data from JSON. ‘

**PaaS**

PaaS stands for platform as a service and is a cloud service that virtualises the resources needed to host a service, this includes the hardware, operating system and so on. In PaaS the only thing the user manages themselves is the code and data they use for their service. Therefore, PaaS is a very simple way for developers to easily launch their service without having to buy their own servers with expensive hardware and then configuring everything from scratch like the operating system, network with routers and so on…

**Saas**

System as a service is a fully functional developed service that is ready to be used and has everything included and all necessary resources are virtualized.

**Iaas**

Infrastructure as a service is a cloud service that virtualises the hardware resources needed to run a service, but it gives you the freedom to choose the operating system you want to utilize yourself, and also the data and code for the application to be deployed on the service. You manage the operating system, middleware, runtime, data and application yourself while the IaaS provider takes care of networking, servers, storage and virtualization.

**Openstack**

**GraphQL**

**Testing**

**Webhooks**

**Peer review principles**

**Docker**

**Openstack**

**Firebase**

**Public cloud platform introduction**

**Review of networking foundations**

**Legal and economics aspect of cloud computing**

**Principles of cloud computing**