REST - REpresentational State Transfer

- This is a convention, architectural style or design pattern for creating applications that are client-server based for communication generally with HTTP.
- REST was designed by Roy Fielding.
    - REST is a direct alternative to a system called SOAP (Simple Object Access Protocol) which is more secure but requires a great deal more configuration.
- There are 6 guiding principles/constraints/pillars of REST:
    - Client-Server Architecture: In this design pattern the client that is used by the end user is a separate application then the server which handles persistence and usually business logic.
    - Uniform Interface: They system architecture should be designed in a logical and consistent way allowing the client to easily intuit how to access further functionalities.
        - Generally this is done with collective and singleton resources being available at clearly defined URLS:
            - www.mycarwebsite.com/cars - this is a collective resource that gives you all cars.
            - www.mycarwebsite.com/cars/1 - this gives you a single car with id 1.
        - Also, you create a client-server relationship by providing new information with each request informing the client of additional resources. This is often called HATEOAS – Hypermedia As The Engine Of Application State.
    - Layered Server Architecture: The server should process information through layers, only exposing information about the communication layer (controllers) and abstracting and logic or persistence layers.
    - Cacheable: Because information lives at specific endpoints information should be highly cacheable by the client.
    - Stateless: Each request from the client to the server must contain all information necessary to complete the request. The server does not store any information about the client between requests.
    - (Optional) Code on Demand: This allows functionality to be extended by the client if the client downloads and executes certain code modules generally in the form of applets or scripts to be used with the resources. This is not well defined or generally agreed upon in practice.
- REST Resource:
    - Any information can be a resource: a document, an image, a collection of other resources, an object etc. However, resources are not views. This means that REST is dependent on transferring this information in a language agnostic manner, so the client has the ability to display and manipulate the information as it chooses. Examples of such methods are JSON or XML.
    - Whatever the resource is, it requires a unique identifier so it can be accessed. Generally this is the URL location it “lives” at.
- REST and HTTP:
    - REST is generally considered to only work with HTTP (or HTTPS) however Roy Fielding debates this.

- REST uses particular HTTP verbs to indicate the intention of the request. These functions generally align with the CRUD functions.
    - GET – This is used to retrieve a resource. This is the equivalent or Read in CRUD.
    - POST – This is used to add new information to the server. The information for the additional resource should be contained in the body of the request. This is equivalent to Create in CRUD.
    - PUT – This is used to update an entire already existing resource. This is analogous to Update in CRUD.
    - PATCH – This is used to update a specific part of a resource. This is still analogous to Update CRUD.
    - DELETE – This will delete a resource from the server. Analogous to Delete in CRUD.