|  |  |
| --- | --- |
| **Name:** |  |
| **Email:** |  |

**Question 1**

In your own words explain the 4 sub constraints in REST’s Uniform Interface. Give an everyday example to illustrate each of the constraint.

* Identify resources thru identifiers
  + E.g. the path to find a specific file must be uniquely identifiable
* Manipulation of resources through its representation
  + E.g. to retrieve JSON or HTML representation of the same data
* Self-describing message
  + E.g. telling a waiter your order, need to tell the full order details
* Hypermedia as the engine of application states
  + E.g. changes on the webpage based on what was the user’s previous action

**Question 2**

What is the difference between the following HTTP methods?

1. POST, PUT and PATCH

* Post: To submit data of a specified resource, must be for whole new record
* Put: Replace data of a specified resource, must provide id, and data for full record
* Patch: Partially modify the specified resource, must provide id

1. GET and HEAD

* Get: To request data from a specified resource
* Head: Similar to head, but only getting header data without the payload.

**Question 3**

You have a monolithic web application for managing warehouses. The application exposes the following end points

* /warehouses – list of all warehouses
* /warehouse/<warehouse\_id> – returns the warehouse’s details
* /warehouse/<warehouse\_id>/inventories – inventory list for the warehouse
* /inventories – list of all the inventories
* /inventory/<inventory\_id> – inventory detail
* /inventory/<inventory\_id>/report – generate a report

Describe how you can scale this application

* By duplication
* Duplicating the application to create redundancy, while data may or may not have been data partitioned
* E.g. 2 instances of webserver
* By functional decomposition
* For each function, there will be a separate application and it interacts with its own database
* If it needs to access another function, it will call the API of the separate application
* E.g. abc.com/warehouses -> this goes to warehouse webserver, while abc.com/inventory -> this goes to inventory webserver
* By data partitioning
* Categorising the data and splitting it up into different databases, e.g. by geography.
* E.g. API requests for warehouses are retrieved from the warehouse database

**Question 4**

Study the top headlines REST API from newsapi.org. Answer the following questions

1. List the different ways you can present the API key when performing an invocation

* As part of the request parameter or via the X-Api-Key HTTP header
* Preference to be in the http header, due to security reasons as you can bookmark if in query string

1. Construct a URL to get 30 technologies headlines from Japan

GET https://newsapi.org/v2/top-headlines?country=jp&category=technology&apiKey=API\_KEY

1. What is the status code if an incorrect API key is used?

* 401

1. How long will the result be cached?

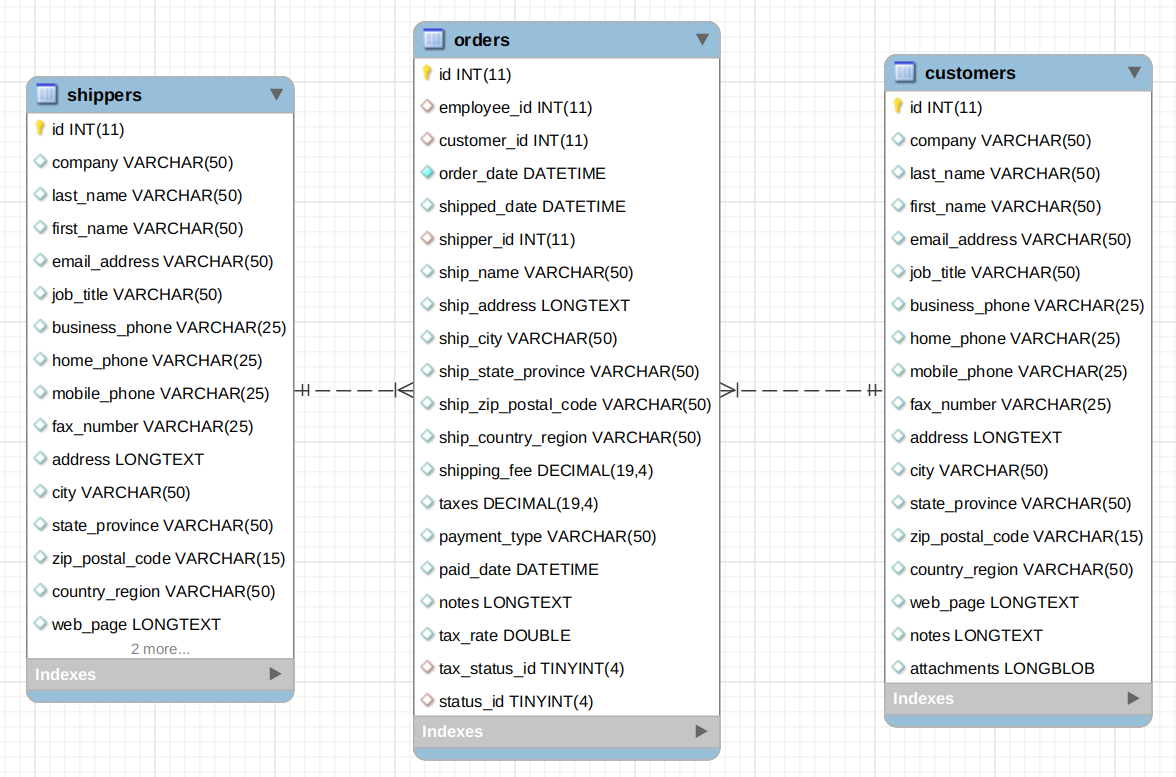
Graphical user interface, text, application

Description automatically generated

3min

**Question 5**

Study the following entity-relationship diagram



Both customer and shippers has a one to many relationships with orders.

Answer the following questions.

1. Design one or more API endpoints to return a list of customers and a single customer

/customers

/customer/<name>

1. What are some criteria and how might you might include in your endpoint (wrt Q5a)?

* To specify the data field to retrieve, else there may be too many returned fields
* This can be done via sub resources
  + e.g. /customers/first\_name

1. Show a sample output of a customer’s list as a result of performing a GET on the resource. (wrt Q5a)

{ “customers” : [ {“company”: “abc”, “last\_name”: “tan”…..}, {“company”: “xyz”, “last\_name”: “chan”…..} ] }

1. How do you provide flow control or pagination support (wrt Q5a)?

* **Flow control**: Throttling API request via token bucket
* **Pagination**: Support for offset and limit in the query string

**Question 6**

You have deployed a service to encode video viz. convert AVI to mp4, etc. Subscribers of your service uploads their video to the service; after conversion the converted video is returned to the subscriber (assume that the conversion time is short).

You charge the subscribers based on the 2 criteria.

1. Subscription rates based on the cumulative video sizes: 500GB, 1TB, 1.5TB, etc. A subscriber who subscribe to the 500GB package can upload a maximum amount of 500GB videos.
2. Charge the subscribe based on their ingress and egress traffic viz. the upload and downloads of the videos.

Design an API for this encoding service to give your subscribers control over their encoding process.

You can ignore authentication.

* Create a POST endpoint that accepts multipart/form-data
* Use content-encoding and accept-encoding to allow for compression/chunk transfer/partial response in case of loss of connection
* Implement the following endpoints
  + /convert/?package=<size>&to=<format>

**Submission**

Copy this Word document to your repository and commit it.

git add .

git commit -m ‘worksheet01’

git push origin master