Engineering Tech Lead Challenge

Hello There!

My name is Hector Rios Carranza, by I hereby allow myself to put to your kind consideration the test you take to participate for the ENGINEERING TECH LEAD vacancy, for which I send the evidence of the points requested for the system, I hope you like it and I thank you very much for the opportunity to participate!

Grettings , Hector Rios!

You have to create a very simple system that will consist of 3 parts: a client to send and display information, a queue to process the information and an API that will do some logic and use a database to persist information.

The client will have two parts, one to input some information and other to display some information.

The queue will receive the information and process it and the API stores and retrieves info from the DB.

The process is simple:

When a message (any text max 20 characters) is submitted it has to pass through the queue then the queue will process the message. This process includes waiting a random time (5 to 10 seconds) to use the API to write it to the database, if the message includes some letters it has to be discarded (b,c,d,t) at the end send a notification to the client that a message has been processed with an ID.

The client should use the API to retrieve the messages stored in the database.

For the Back-End we expect the solution to include:

- RESTful API
- Use Express.js over Node.js
- Connection to the database can be MYSQL or PostgreSQL (use an ORM like Sequelize)

EVINDENCE:

```
const express = require('express')
const cors = require('cors')
const app = express()
const port = 3000
const sequelize = require('./database/db.js');

//Middleware
app.use(express.json());
app.use(express.urlencoded({extended:false}));
app.use(cors());
app.get('/', (req, res) => {
  res.json("Welcome to my Express-Sequelize Interface, by Hector Rios"
});

//app.use('/api/posts',require('./routes/posts'));
app.use('/api/posts',require('./routes/messages'));

app.listen(port, () => {
  console.log(`Example app listening at http://localhost:${port}`)
  //conectarse a la base de datos.
  sequelize.sync({force:false}).then(()=>{
    console.log("Se conecto a la base")
  }).catch(error=>{
    console.log('Error de conexion',error)
  })
})
```

messages.js

```
const express = require('express');

const router = express.Router();
const Post = require('../database/models/Mesagges');
//select all
router.get('/select',(req,res)=>{
    Post.findAll().then(posts=>{
        res.json(posts);
    })
})

//INSERT MESAGGES;
router.post('/',(req,res)=>{
    Post.create({
        message:req.body.message

    }).then(post => {
        res.json(post)
    })
});

//SELECT BY ID
router.get('/:id',(req,res)=>{
    Post.findByPk(req.params.id).then(post=>{
        res.json(post);
    })
})

module.exports = router;
```

Queue

EVIDENCE:

```php
class Queue {

    function __construct() {

    }

    function analizeMessage($params){
        $error = false;
        $msj = "Everything ok";
        if($params["message"]!=""){
            $pos = strpos($params["message"], 'b');
            if ($pos === false) {
                 $pos = strpos($params["message"], 'c');
                if ($pos === false) {
                    $pos = strpos($params["message"], 'd');
                    if ($pos === false) {
                         $pos = strpos($params["message"], 't');
                        if ($pos === false) {
                            $time= rand(5,10);
                            sleep($time);

                            $ch = curl_init();
                            curl_setopt($ch, CURLOPT_URL, "http://localhost:3000/api/
                                posts/");
                            curl_setopt($ch, CURLOPT_VERBOSE, 1);
                            curl_setopt($ch, CURLOPT_HTTPHEADER,array('Content-type:
                                application/x-www-form-urlencoded; charset=UTF-8'));
                            curl_setopt($ch, CURLOPT_HEADER, false);
                            curl_setopt($ch, CURLOPT_TIMEOUT, 30);
                            curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
                            curl_setopt($ch, CURLOPT_POST, 1);
                            $message ='message='.urlencode($params["message"]);
                            curl_setopt($ch, CURLOPT_POSTFIELDS, $message);
                                $httpResponse = curl_exec($ch);
```

For the Front-End we expect the solution to:

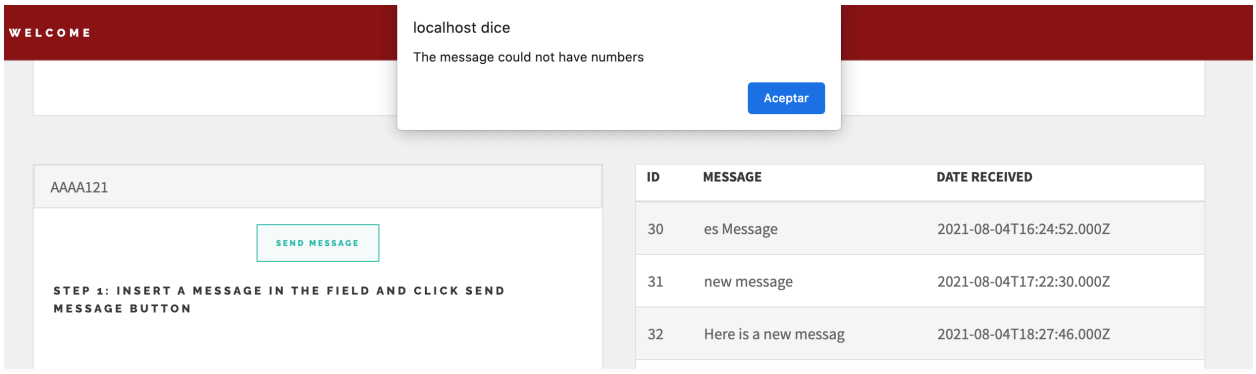
A screen with 3 sections:

- Use Angular

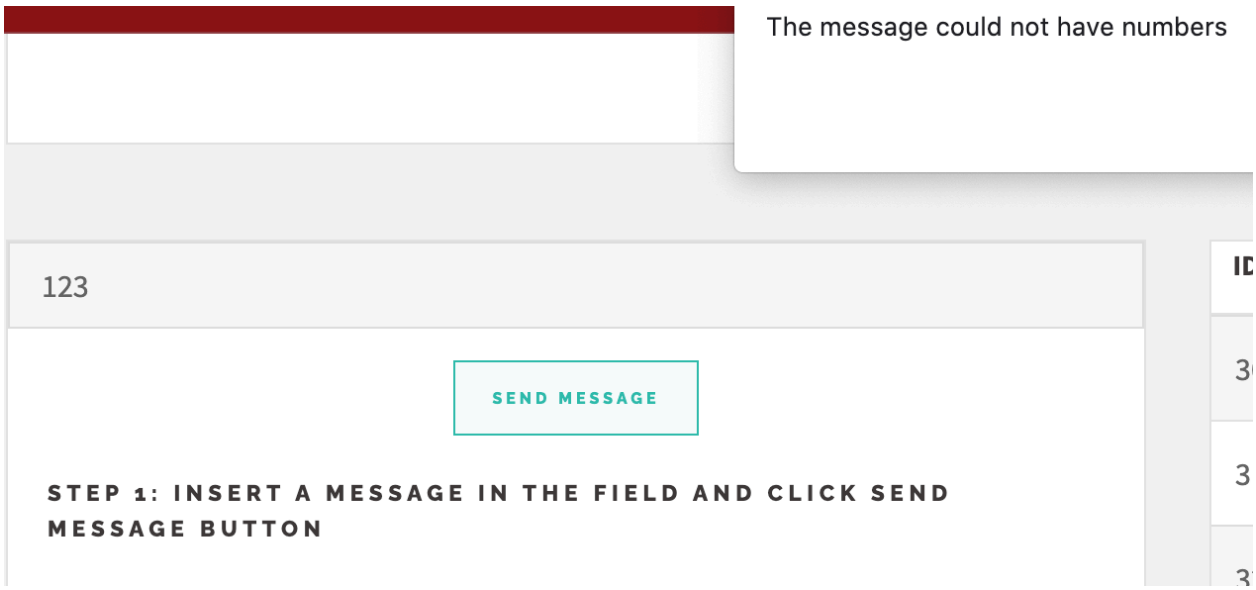
```
var app = angular.module('testApp',[]);

app.controller('funciones',function($scope,$http){
    $scope.mensaje="";
    $scope.numeros = [0,1,2,3,4,5,6,7,8,9,0];
    $scope.status = "error";
    $scope.id = "";
    $scope.validarMensaje = function(){
        var success = true;
        if($scope.mensaje!=""){
            if($scope.mensaje.length<=20){
              for(i=0;i<$scope.numeros.length;i++){
               if($scope.mensaje.indexOf($scope.numeros[i]) != -1){
                   success = false;
               }
              }


              if(success){
```

```
<!-- Mini Post -->
<article class="mini-post" ng-controller="funciones as a2">
            <header>
                    <h3>STEP 1: Insert a message in the field and click send message button</h3>
            </header>
          <div class="row gtr-uniform">
            <div class="col-12 col-12-xsmall">
                <input type="text" ng-model="mensaje" name="mensaje" maxlength="20"  placeholder="Write your message here"  />
            </div>
            <div class="col-12 col-12-xsmall" style="text-align:center">
                <button class="button small" ng-click="validarMensaje()" >Send Message</button>
            </div>
          </div>
        </article>
```

- notifications area



- an input for the messages (validate on size and not allowing numbers)



- a table that will show messages with a button to refresh or pull new messages. A plus if new messages are shown without refreshing.
-

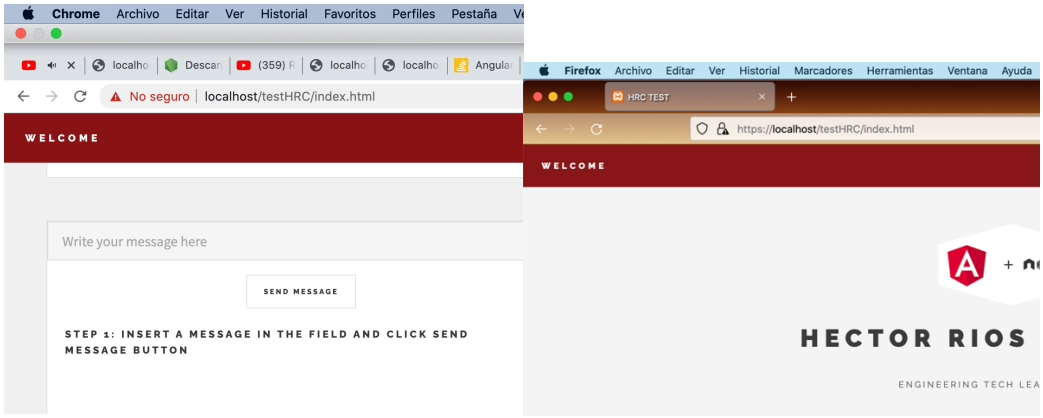
ID	MESSAGE	DATE RECEIVED
30	es Message	2021-08-04T16:24:52.000Z
31	new message	2021-08-04T17:22:30.000Z
32	Here is a new messag	2021-08-04T18:27:46.000Z
33	ok!	2021-08-04T18:32:41.000Z
34	new	2021-08-04T18:33:12.000Z
35	AAAA	2021-08-04T18:55:27.000Z

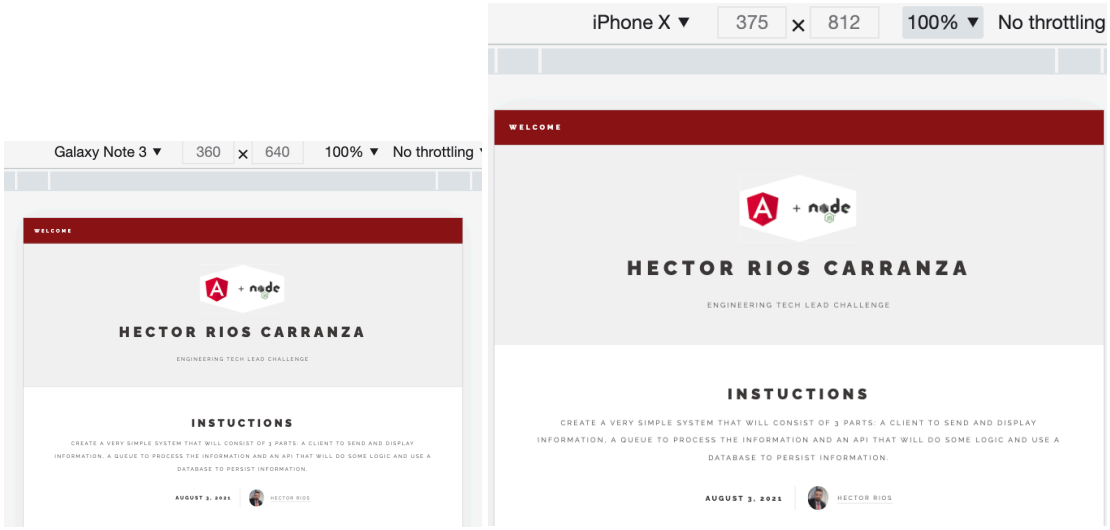
REFRESH TABLE

STEP 2: SAVED MESSAGES

HERE THE MESSAGES SAVED BY THE API ARE SAVED.

- The page must work in Chrome, Firefox, Safari, IE11, Edge, IOS and Android.



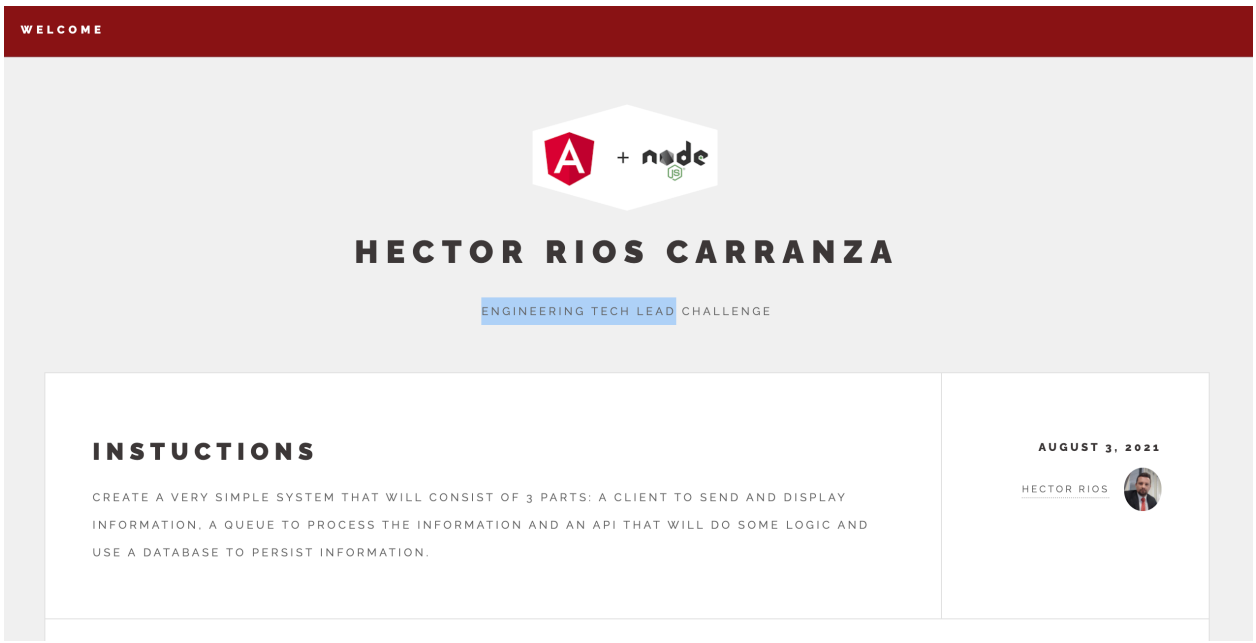


- Implement basic error handling in the task creation form.

```
                    $httpResponse = curl_exec($ch);

                    if(curl_getinfo($ch, CURLINFO_HTTP_CODE) === 200){
                    if (strstr($httpResponse,"ERROR errNum")){
                         $msj = ("Error saving message : ".$httpResponse);
                         $error  = true;
                    }else{
                       $array = json_decode($httpResponse,false);

                       $id = $array->id;
                       $msj=$id;
                    }
                    }else{
                         $msj = ("Error saving message: ".curl_error($ch).'('.curl_errno($ch).')'.$httpResponse);
                         $error  = true;
                    }
                    curl_close($ch);
               }else{
                    $error = true;
                    $msj = "The message could not content letter t";
               }
          }else{
               $error = true;
               $msj = "The message could not content letter d";
          }
     }else{
          $error = true;
          $msj = "The message could not content letter c";
     }
}else{
     $error = true;
     $msj = "The message could not content letter b";
}
}{
```

- For styling you can use any CSS preprocessor like SCSS or Less.**

  Evidence: In the CLIENT


WELCOME
+ node
HECTOR RIOS CARRANZA
ENGINEERING TECH LEAD CHALLENGE
INSTUCTIONS
CREATE A VERY SIMPLE SYSTEM THAT WILL CONSIST OF 3 PARTS: A CLIENT TO SEND AND DISPLAY INFORMATION, A QUEUE TO PROCESS THE INFORMATION AND AN API THAT WILL DO SOME LOGIC AND USE A DATABASE TO PERSIST INFORMATION.
AUGUST 3, 2021
HECTOR RIOS