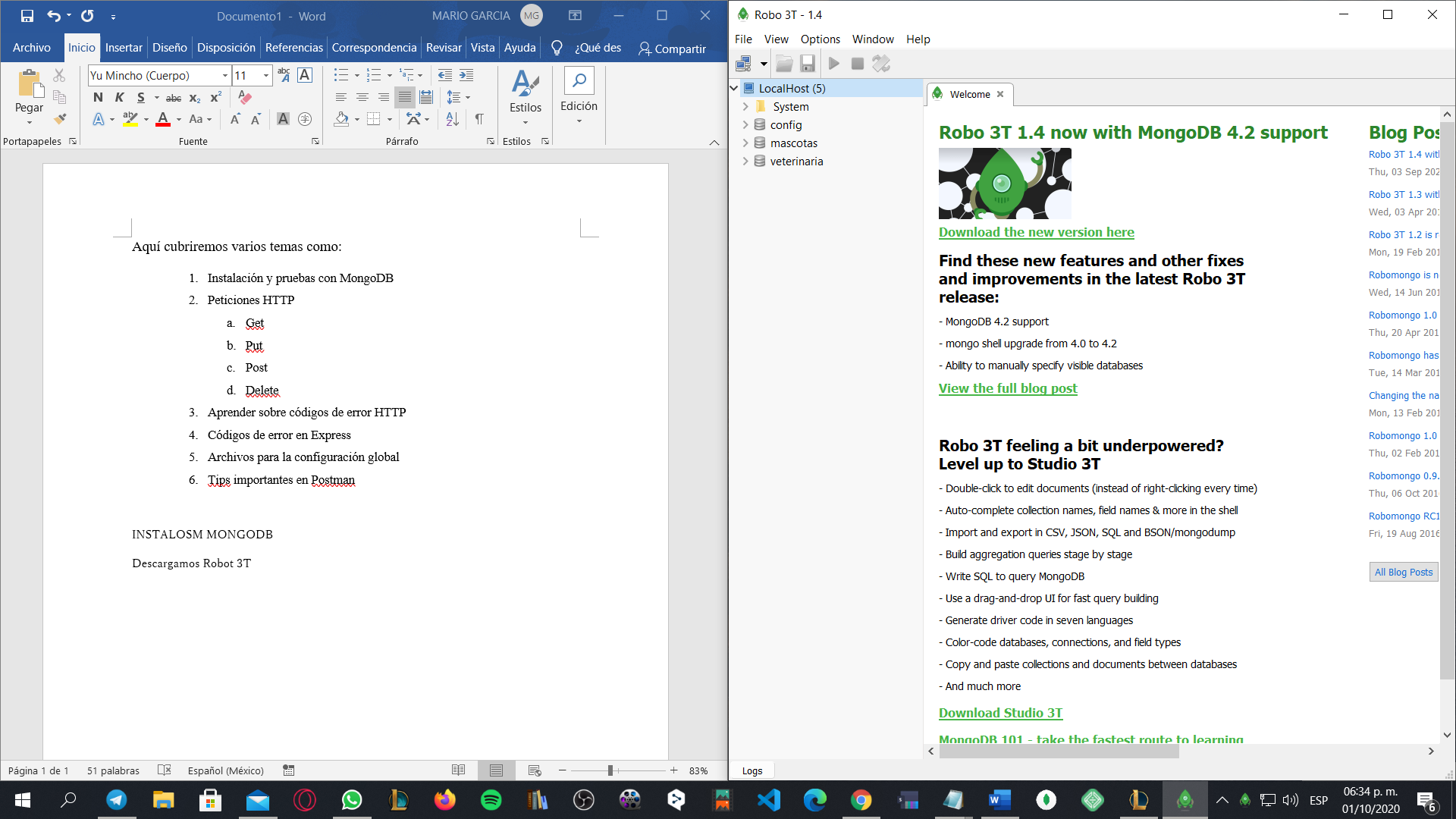
Aquí cubriremos varios temas como:

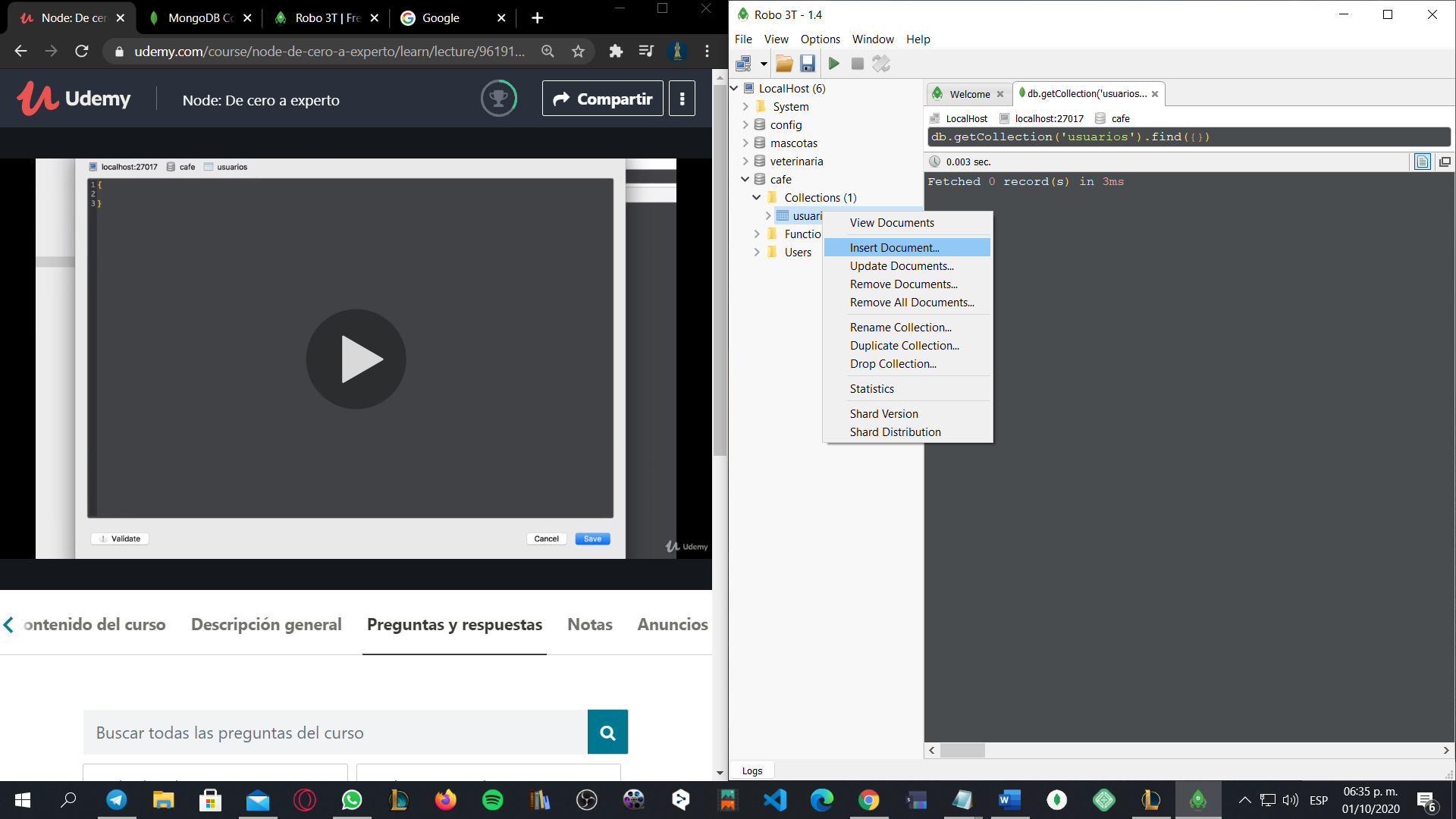
1. Instalación y pruebas con MongoDB
2. Peticiones HTTP
   1. Get
   2. Put
   3. Post
   4. Delete
3. Aprender sobre códigos de error HTTP
4. Códigos de error en Express
5. Archivos para la configuración global
6. Tips importantes en Postman

INSTALOSM MONGODB

Descargamos Robot 3T, creamos una nueva conexión en local y nos aparecen nuestra Bases de datos

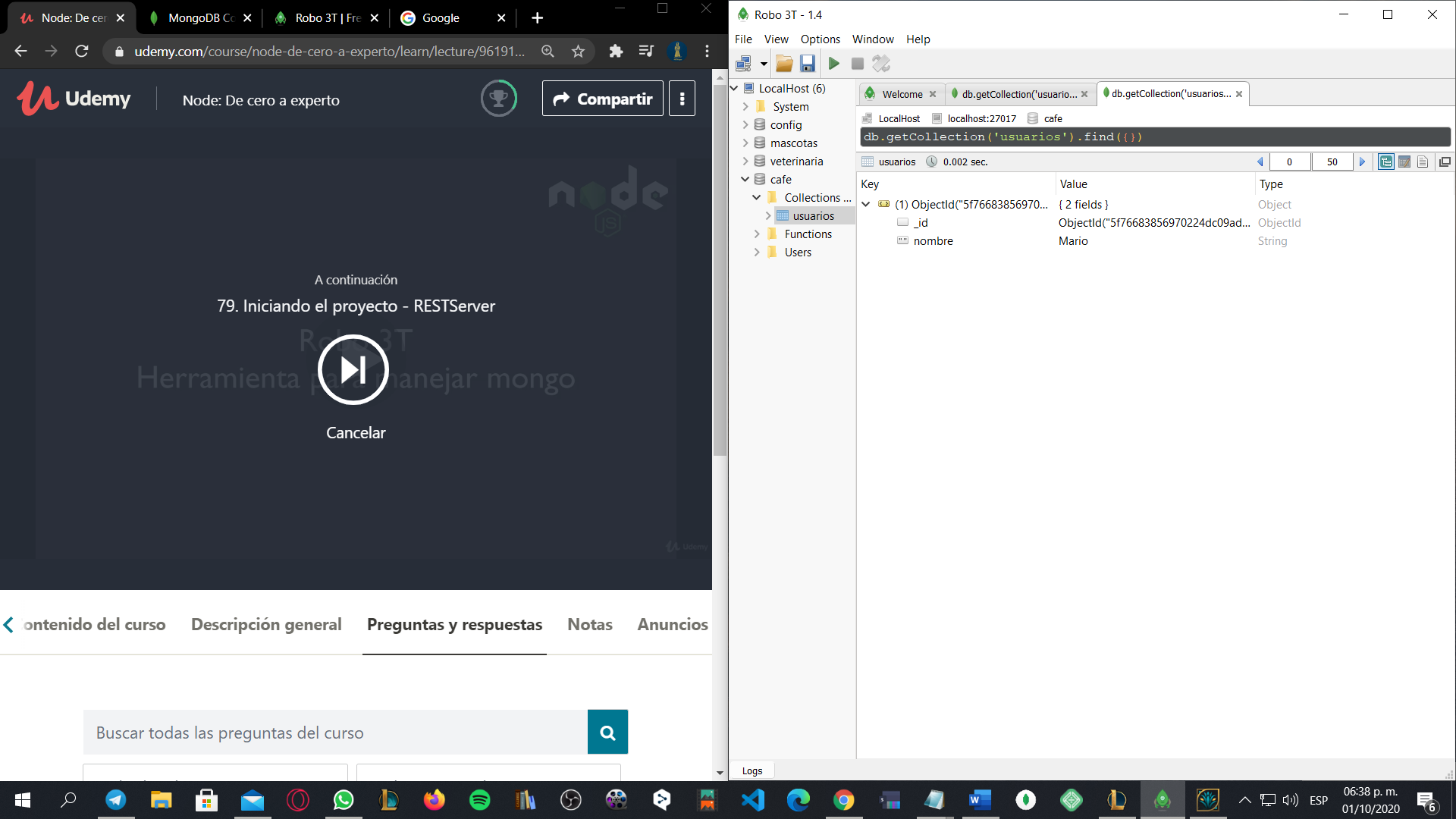


Creamos una colección y click izquiero podemos insertar los documentos



Ya adentro el jason ejemplo

“nombre”:”Mario”



**INICIANDO PROYECTO REST SERVER**

Npm init

Trabajaremos con express

npm install express –save

creamos capreta nueva server con el archivo serve.js

para ejecutarlo node server/server por que esta dentro de la carpeta

const express = require('express')

const app = express()

app.get('/', function(req, res) {

    res.send('Hello World')

})

app.listen(3000, () => {

    console.log('Escuchando el puerto 3000');

})

Como vamos a trabajar con Json el método send lo cambiamos por json

Mejor usar nodemon server/server para que este atento a los cambios

**PETICIONES HTTP - GET -PUT -POST -DELETE**

Vamos a ahacer las 4 peticiones básicas que necesitaremos

const express = require('express')

const app = express()

app.get('/usuario', function(req, res) {

    res.json('get usuario')

})

app.post('/usuario', function(req, res) {

    res.json('post usuario')

})

app.put('/usuario', function(req, res) {

    res.json('put usuario')

})

app.delete('/usuario', function(req, res) {

    res.json('delete usuario')

})

app.listen(3000, () => {

    console.log('Escuchando el puerto 3000');

})

Como nosotros vamos a modificar un usuario en especifico necesitamos poner el comid para que acepte cualquier id, y agregamos la variable id que sale del URL

app.put('/usuario/:id', function(req, res) {

    let id= req.params.id;

    res.json({

        id

    })

})

Trabajamos con el post, tenemos que obtener las variables que mande el cliente..Existe un paquete que se llama body-parser

npm install body-parser –sabe

const bodyParser = require('body-parser');

e implemantamos los middleware

const bodyParser = require('body-parser');

// parse application/x-www-form-urlencoded

app.use(bodyParser.urlencoded({ extended: false }))

// parse application/json

app.use(bodyParser.json())

con una variable que es la que,va a tiener toda la información que necesitamos, las peticviones se guardaran en esa variable

app.post('/usuario', function(req, res) {

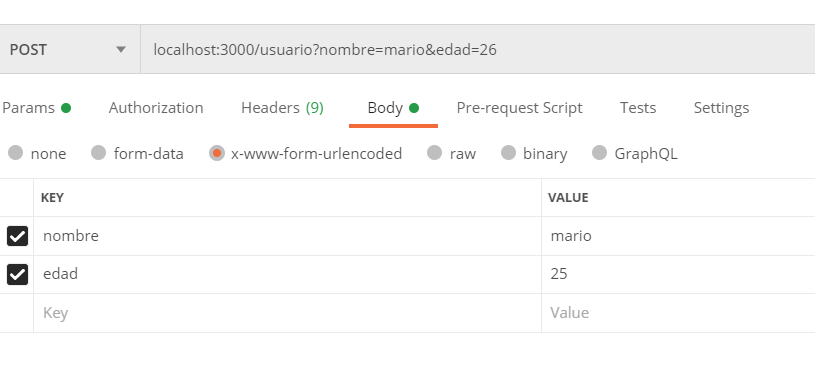
    let body = req.body;

    res.json({

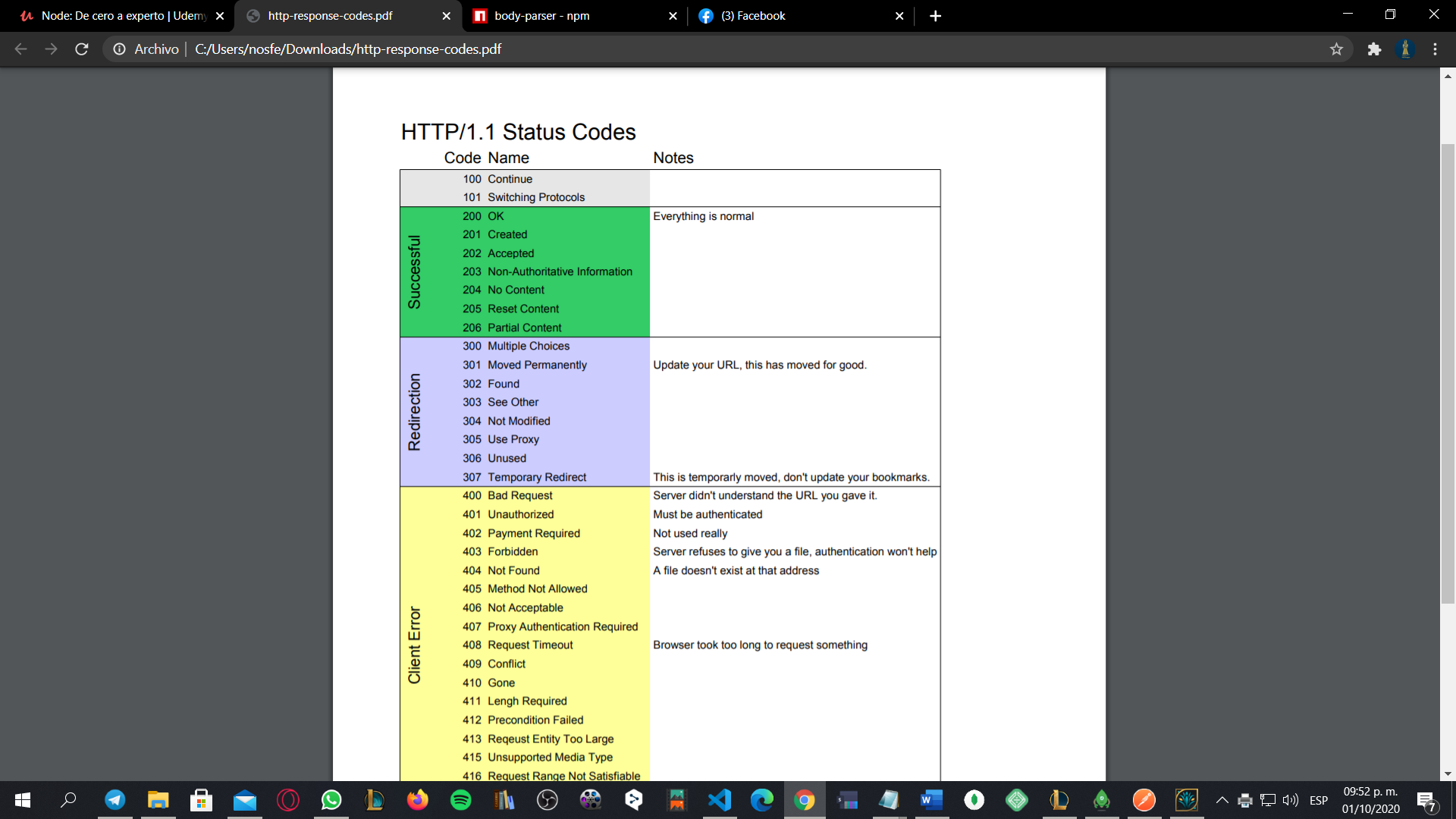
        body

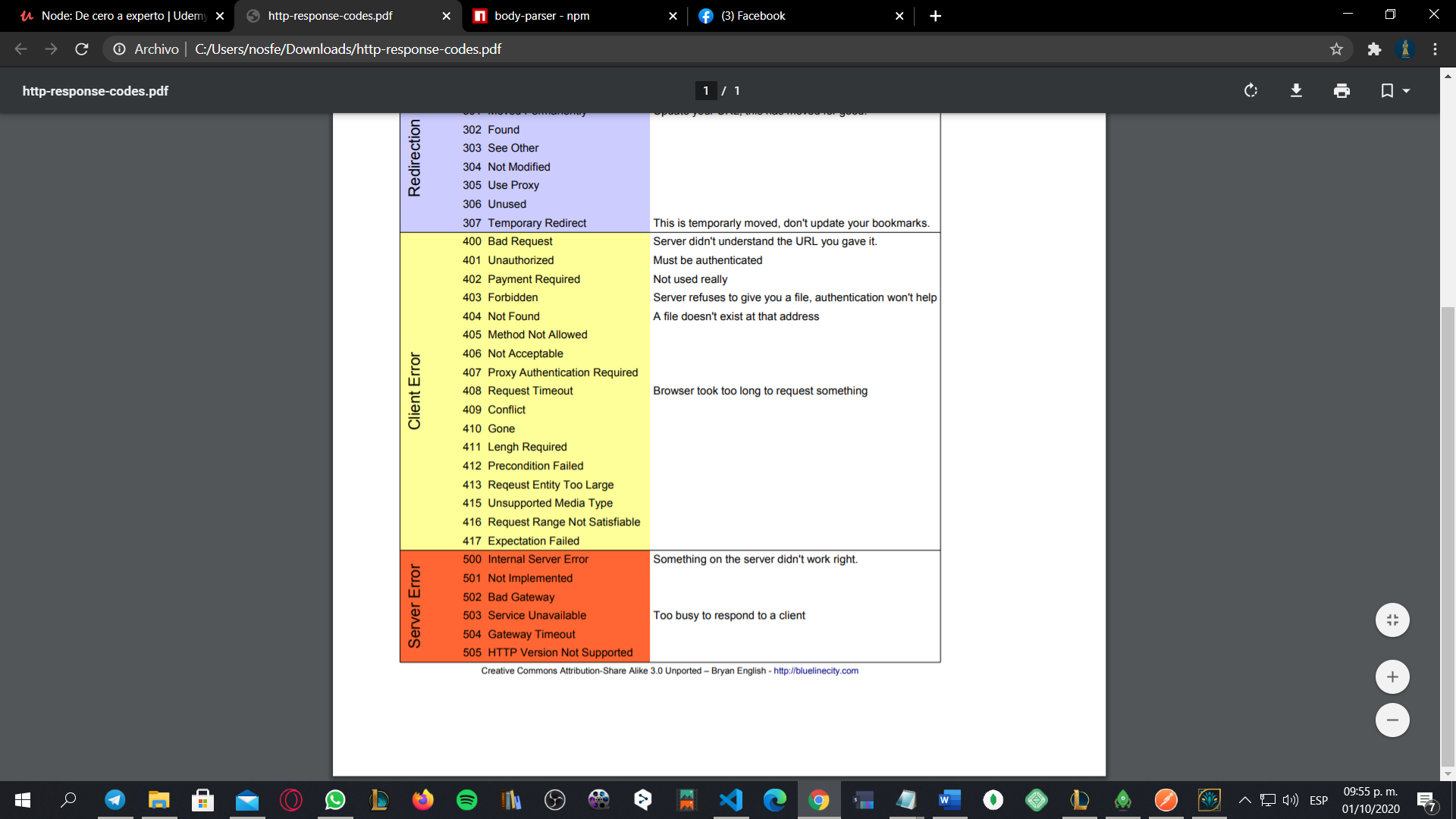
    })

Ya salió la petición pero por x-www-form-urlenconde



**CODIGOS DE RESPUESTAS HTTP**





Quedaría un post con el estatus 400

})

**USANANDO CODIGO DE RESPUESTA HTTP**

**400 para cuando el cliente no introduce un dato**

app.post('/usuario', function(req, res) {

    let body = req.body;

    if (body.nombre === undefined) {

        res.status(400).json({

            ok: false,

            mensaje: 'El nombre es necesario'

        })

    } else {

        res.json({

            persona: body

        })

    }

})

**Creando un archivo de configuración global**

Mas adelante nuestro proyecto tendrá 2 partes “desarrollo y producción”, El puerto que estamos enviando es el 300, cuando subamos a producción no sabremos que puerto tendrá, vamos a crear un archivo para estar entre producción y desarrollo sin necesidad de meter mano al código.

Hacemos una capreta en el server que se llame config y dentro el archivo config.js, esto va a reemplazar el path por que que ponga nustro servidor.

///PUERTO

process.env.PORT = process.env.PORT || 3000;

importamos en el archive principal server,js

require('./config/config');

const express = require('express')

luego donde va estar escuchando ya no será el puerto 3000, va a hacer en ese puerto que ponga el path.

app.listen(process.env.PORT, () => {

    console.log('Escuchando el puerto xxx');

})

RESPALDO DEL REST SERVER A GITHUB

Lo vamos a subir.. ya trabajamos esto

**SUBIRLO A HEROKU (RECOMIENDA LA FORMA QUE HICIMOS ANTERIOR MENTE)**

Podemos hacerlo desde la terminal esta vez.

**Heroku create**

Esto asigna un nombre en automatico del proyecto

Para subirlo

**Git push heroku master**

Y abrirlo en el explorador

**Heroku open**

Tenemos que agregar en el **script del package.json**

**“start” “node server/server.js”**

Temenos que Volver a actualizar el git antes de subirlo por los cambios ya echcos

Git add .

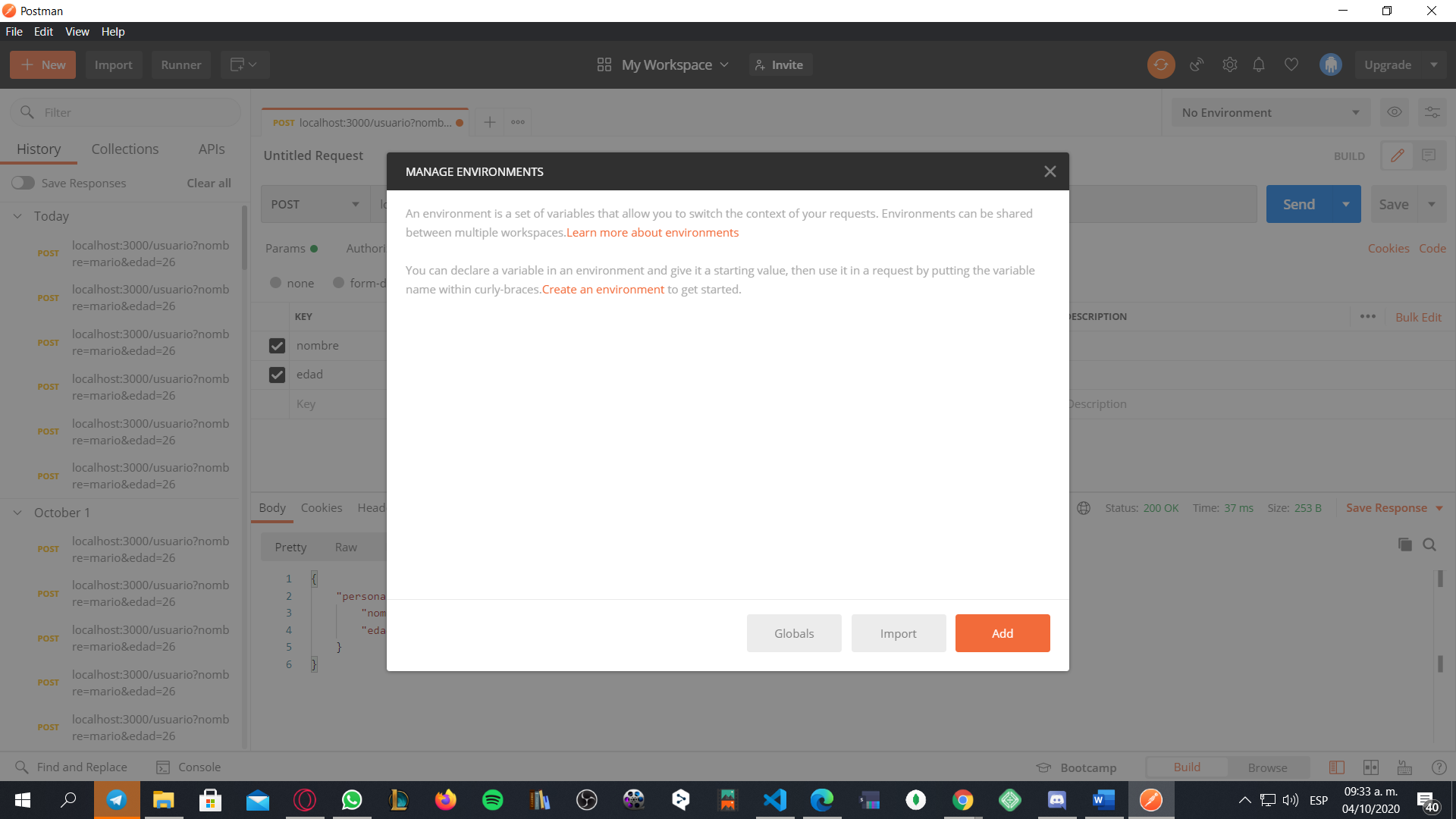
O al archivo que se modifico

**Git commit -am “Npm start listo”** //agrega commit automaticamente y agrega archivos

Y ya hacemos subimos

**Git push heroku master**

**TIP AMBIENTE PRODUCCION Y DESARROLLO EN POSTMAN**



Creamos el ambiente de producción y el otro de desarrollo, el primero con la url que nmos da el servido y el otro con el localhost:3000

Para poner en la url y seleccionarlos para que reconozca los valores que introducimos {{url}} y seleccionar el desarrollo o producción.

