# Creació d'una API web amb Node.js

Aquesta pràctica consistirà en crear una petita API REST amb Node.js. S’avaluarà mitjançant un lliurament de la pràctica molt senzill. Servirà per puntuar la UF3 i la UF4.

**Examen de la pràctica: dia 18/5 (5 punts preguntes + 5 punts programar)**

Software que heu de tenir instal·lat:

* Visual Studio Code (o l’IDE que preferiu, però jo us recomano aquest)
* Git
* Node.js (des d’[aquest link](https://nodejs.org/en/))
* Postman o Insomnia per provar la API

**IDE a utilitzar:** Visual Studio Code amb els plugins de Node.js i Express

Tutorial i documentació que heu de seguir:

<https://docs.microsoft.com/ca-es/learn/modules/build-web-api-nodejs-express/>

Es tracta d’anar seguint aquest tutorial mitjançant les instruccions que hi ha a continuació, però procurant comprendre el codi i què fa cada fitxer, així com les comandes de node.js que emprarem via terminal. Ja que això serà important de cara a l’examen: comprendre què fa cada cosa i no simplement copiar el codi del tutorial.

**PRIMERA PART: Instruccions (s’aniran actualitzant!):**

* Llegir i entendre les pàgines 1 i 2 del tutorial, que contenen teoria
* Instalar node.js
* Crear un nou repo a github per aquest projecte
* Seguir les instruccions de la pàgina 3 del tutorial: "Creación de una aplicación web básica con Express"
* Després del punt 4 de les instruccions de la pàgina 3, podeu fer el primer commit i push.
* En aquest moment, haurieu de tenir 3 fitxers: ***app.js***, ***package.json*** i ***package-lock.json***
* Investigueu què fan els fitxers ***package.json*** i ***package-lock.json***.

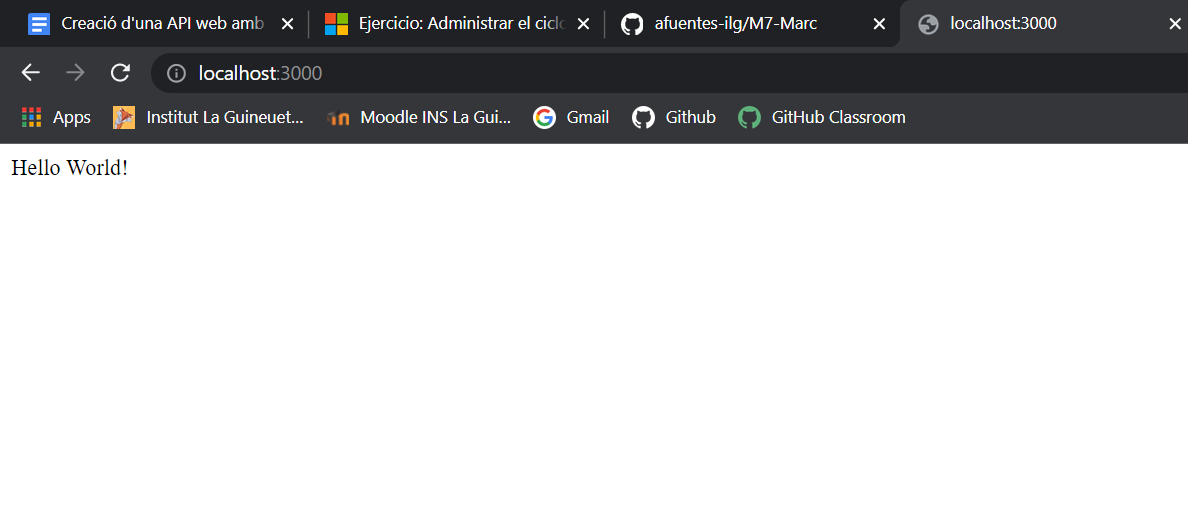
package.json es un fichero que contiene información sobre nuestro proyecto (nombre, versión, etc) además de listar los paquetes de los que depende.

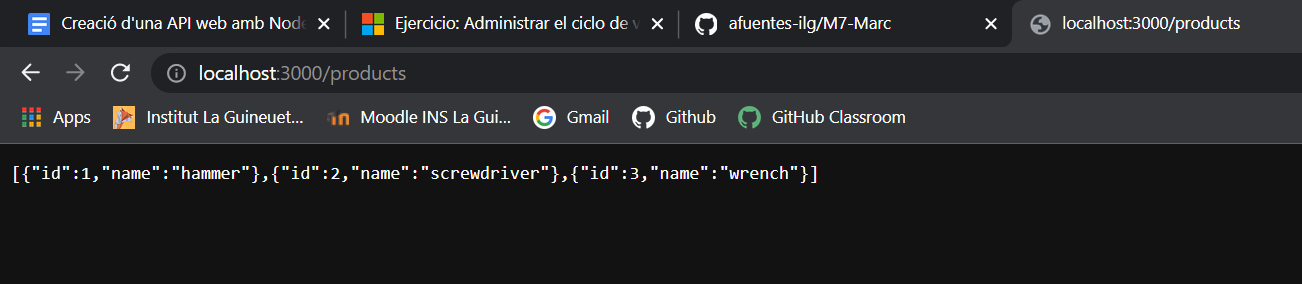
Mientras que el package-lock.json evita este comportamiento general de actualizar versiones minor o fix de modo que cuando alguien clona nuestro repositorio y ejecuta npm install en su equipo

* Investigueu què és la carpeta ***node\_modules***.

Los paquetes que hemos instalado de node.js y express se guardan en esta carpeta de node\_modules

* Executar i veure que funciona. Acte seguit podem fer commit i push
* Seguir les instruccions de la pàgina 3 del tutorial: "Creación de una aplicación web que devuelve datos JSON"
* Executar i veure que funciona tal com es descriu al tutorial. Provar l'endpoint productes amb el navegador i també amb postman.
* Fer commit i push.





Estas capturas de las hago porque ahora me dice que modifique el app.js e imagino que al hacer commit y push se modificara este contenido por otro

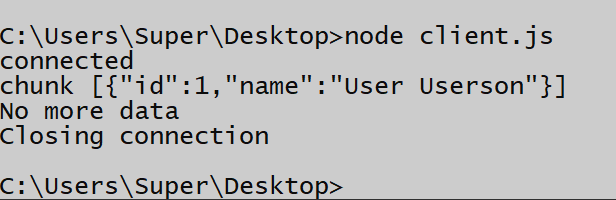
* Llegir la pàgina 4. D’aquí és important comprendre què són els paràmetres ***req***, ***res*** i ***next***.

**req:** este parámetro es la solicitud entrante. Contiene los encabezados de solicitud y la dirección URL de llamada

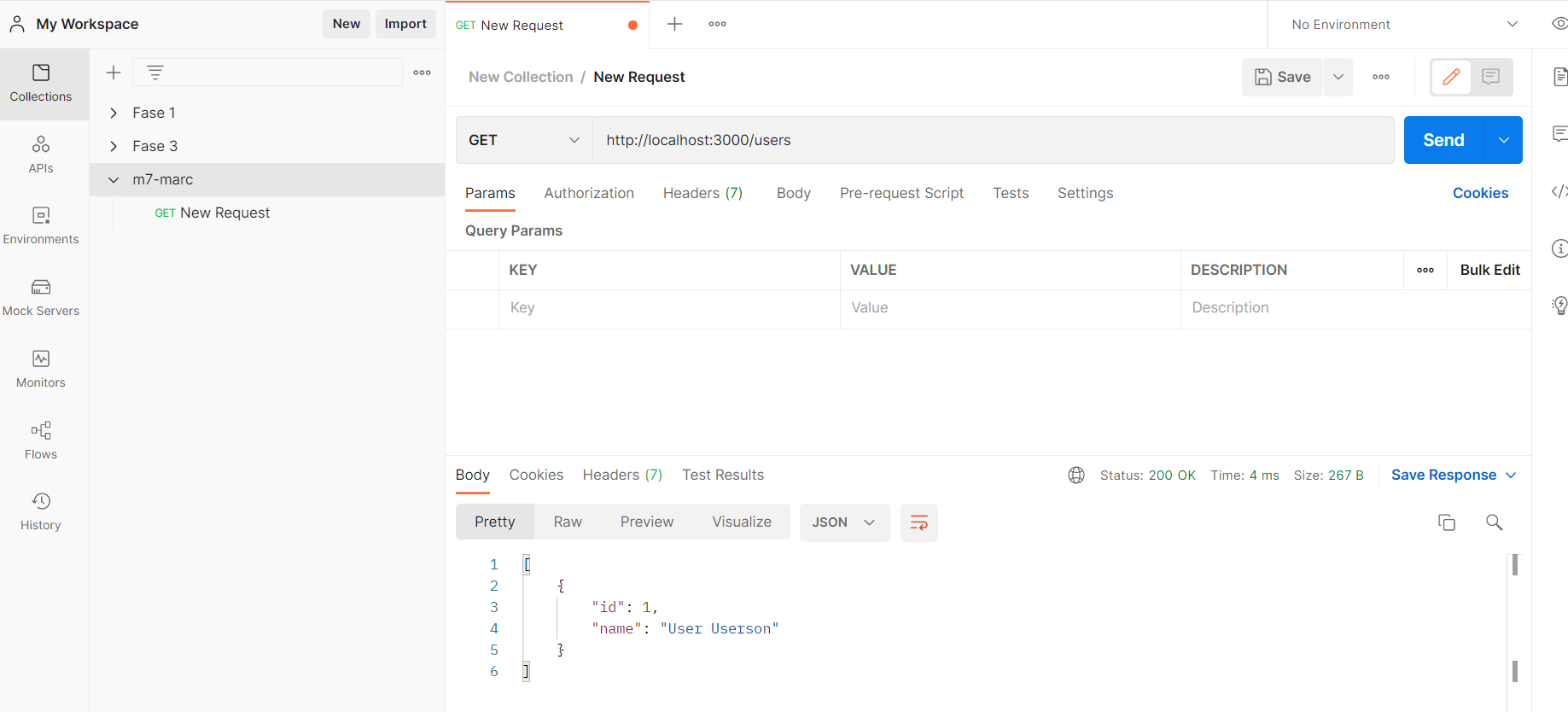
**re**s: este parámetro es una secuencia de respuesta. datos que se devuelen al cliente que realiza la llamada.

**next:** este parámetro indica que la solicitud es correcta y que está lista para procesarse

* Aneu a la pàgina 5 però no us baixeu el repo que diu. Simplement modifiqueu el vostre ***app.js*** perquè sigui com el d’aquesta pàgina. També haureu de crear l’arxiu ***client.js*** i provar-lo tal com diu.



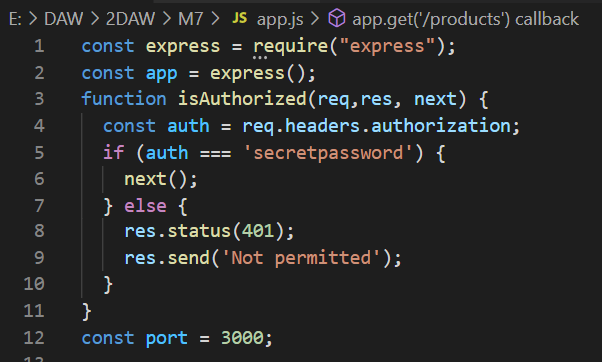
* Un cop hàgiu provat tot el que diu en aquesta pàgina, obriu el postman (o insomnia) i feu una crida al get de ***/users***. Investigueu com fer-ho perquè funcioni i retorni el llistat d’usuaris.

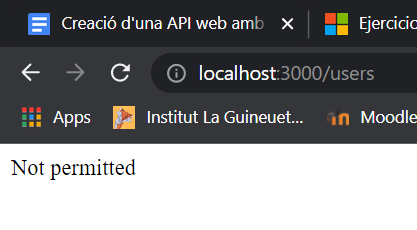


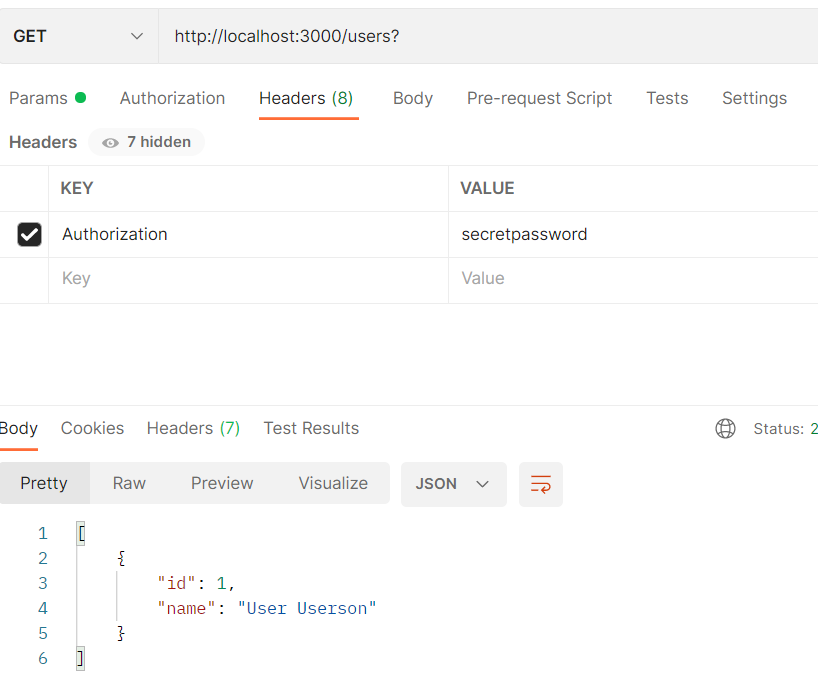
* Finalment, contesteu el qüestionari de la pàgina 6



* Un cop hàgiu fet tot això, podeu fer un últim push. Recordeu penjar al moodle el link al vostre repo.
* Afegir el middleware isAuthorized de la pàgina 5







## SEGONA PART: Instruccions (in progress)

Volem aconseguir dues coses:

* Que la informació de la resposta de /users surti d’una BD SQL que podeu instanciar amb phpmyadmin (necessitareu el port de la BD). Utilitzarem la llibreria **Sequelize** per express. La podeu instalar com a paquet del projecte i definir els models pertinents, a més de la connexió amb la BD.

Un cop tenim la BD encesa i funcionant, tornem al VS Code i seguim la part de programació. Utilitzarem la llibreria **Sequelize** per *express*. La podeu instal·lar com a paquet del projecte i definir els models pertinents, a més de la connexió amb la BD.

* Sequelize és un ORM (object relational mapper)
* Per començar a treballar amb la BD, executem dins del projecte:

npm install sequelize cors mysql2

*npm install* serveix per instal·lar paquets (llibreries) de node.js dins d’un projecte.

Mireu com han canviat els fitxers *package.json*, *package-lock.json*, i el directori *node\_modules*. Haurien d’haver aparegut les 3 llibreries instal·lades.

* Ara instalem la CLI de sequelize

npm install --save-dev sequelize-cli

Amb això podrem executar comandes de Sequelize que ens facilitaran la feina i autogeneraran fitxers.

* Ara inicialitzem l’estructura de Sequelize

npx sequelize-cli init

Haurien d’haver aparegut, dins del vostre projecte, els directoris models, seeders, migrations i config

* Crear el Model Usuari:

npx sequelize-cli model:generate --name User --attributes username:string,password:string,email:string

Entreu al fitxer que s’ha generat a /models i fixeu-vos en l’estructura del fitxer.

* Executar migracions:

npx sequelize-cli db:migrate

Ara entreu al phpmyadmin i mireu l’estat de la vostra BD, hauria de tenir una nova taula Users amb els atributs username, password i email

* Generar un seeder d’usuaris de mentida:

npx sequelize-cli seed:generate --name demo-user

Això genera un fitxer a /seeders amb el qual

* Executar el seeder

npx sequelize-cli db:seed:all

* Ara substituïm el codi de /users perquè retorni les dades reals de la BD

