# Creació d'una API web amb Node.js i Express

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 les UF que queden.

**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-a pi-nodejs-express/](https://docs.microsoft.com/ca-es/learn/modules/build-web-a%20pi-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

* 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 acerca de tu proyecto (nombre, versión, etc) además de listar los paquetes de los que depende.

- El archivo **package-lock.json** es un archivo generado automáticamente cuando se instalan paquetes o dependencias en el proyecto. Su finalidad es mantener un historial de los paquetes instalados y optimizar la forma en que se generan las dependencias del proyecto y los contenidos de la carpeta node\_modules/.

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

La carpeta **node\_modules** es un directorio que se crea en la carpeta raíz de nuestro proyecto cuando instalamos paquetes o dependencias mediante npm. De esta forma, desde nuestro código Javascript podemos importar paquetes externos instalados mediante npm, teniéndolos en nuestro proyecto local

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

req es un objeto que contiene información sobre la solicitud HTTP que generó el evento. En respuesta a req, usa res para devolver la respuesta HTTP deseada.

next() : ejecutará o ejecutará el código después de que finalice toda la función de middleware. return next() : Al usar return next saltará la devolución de llamada inmediatamente y el código debajo de return next() será inalcanzable.

* 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.
* Afegir el middleware isAuthorized de la pàgina 5
* 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.

## 

## 

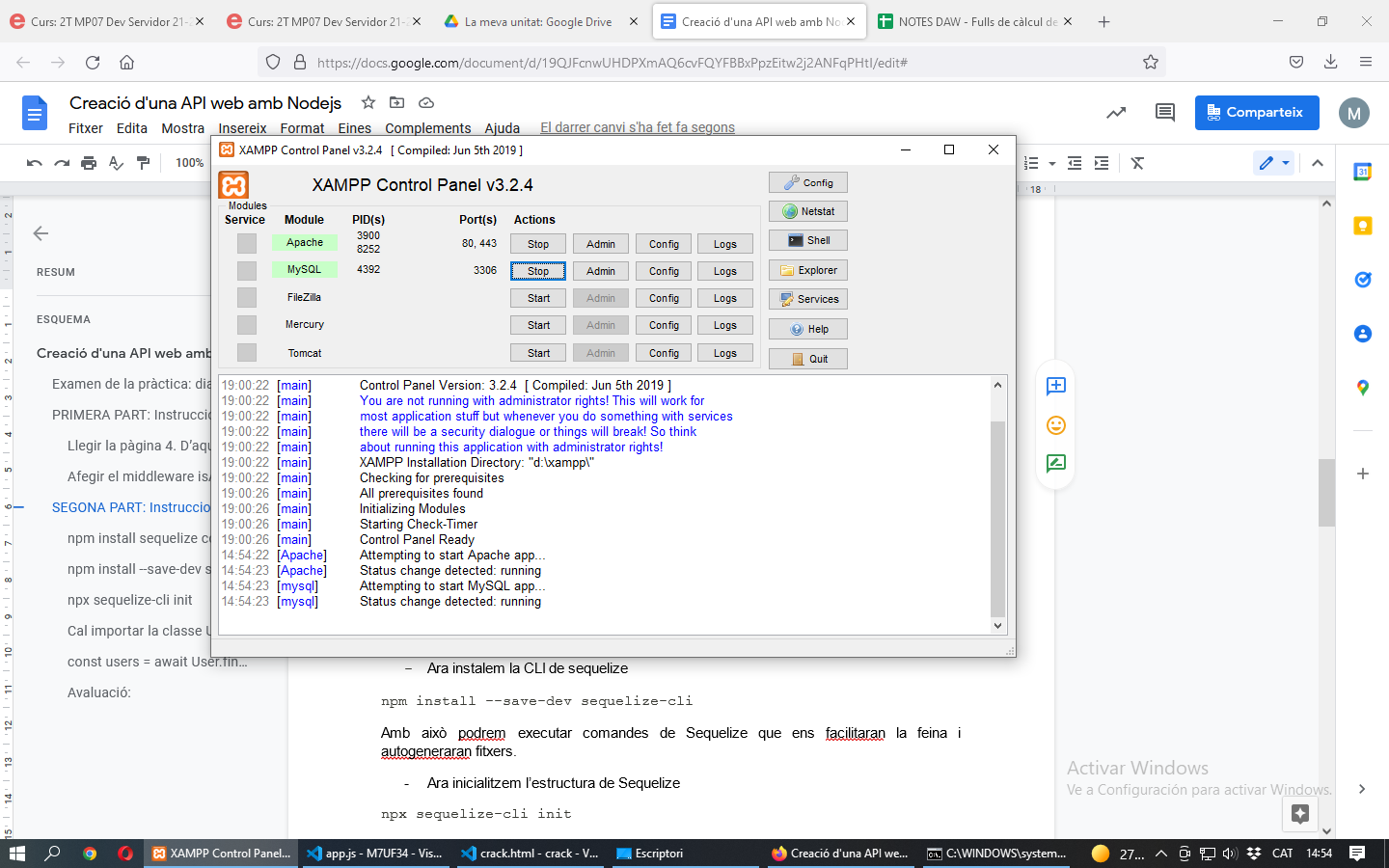
## 

## SEGONA PART: Instruccions

Volem aconseguir:

Que la informació de la resposta de /users surti d’una BD SQL que podeu instanciar amb phpmyadmin (necessitareu el port de la BD).

* Obriu el XAMPP i inicialitzeu l’Apache i el MySQL



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

app.get('/users', isAuthorized, (req,res) => {

res.json([{

id: 1,

name: 'User Userson'

}])

})

Cal importar la classe User de ./models/users.js i podem utilitzar el mètode *findAll()*

app.get('/users', isAuthorized, async (req,res) => {

const users = await User.findAll();

res.json(users);

})

Ara si executem un altre cop la crida GET /users des de POSTMAN, ens hauria de donar el JSON amb l’array d’usuaris que hi ha a la BD.

Proveu de modificar de nou la BD manualment, esborrar, afegir o modificar algun usuari. I torneu a fer la crida /users. Us hauria de donar sempre la informació de la BD.

**Avaluació:**

Pugeu a la tasca del moodle el link al vostre repo.

* La primera part de la pràctica són 5 punts, i la segona 5 punts més

De cara a l’examen:

* Les preguntes seran sobre:
* Què és una API, per què serveixen, donar algun exemple
* Què és el directori node\_modules
* De què serveix el package.json i quina diferència té amb el package-lock.json
* Com s’instal·len paquets de l’entorn npm i quina utilitat tenen
* Què són i què fan els paràmetres req, res i next d’una petició http en node.js
* Què és un middleware i donar algun exemple.
* La part de programar consistirà en fer el mateix que heu treballat a la pràctica:
* Crear una app en Express amb Node.js.
* Fer un endpoint que retorni informació inventada en format JSON
* Afegir, a aquest endpoint, un middleware d’autenticació per contrasenya
* Alguna cosa extra per a qui vulgui nota…