


CURS:	2020-2021	DATA:	28.05.2021	Qualificació	
MATÈRIA:	Bases de dades				
TEMA:	Bases de dades objecte-relacionals				
ALUMNE/A:					

## OBSERVACIONS

- Realització d'una prova en la que els alumnes hauran de realitzar una sèrie d'exercicis pràctics relacionats amb els continguts treballats durant el desenvolupament de les activitats AC1, AC2 i AC3
- Resultats d'aprenentatge avaluats: RA1 (60%)

## ENUNCIAT

Anem a desenvolupar, amb *Express* i *MongoDB*, una aplicació web (i també una API) que ens permeti gestionar les dades relatives a les nostres pizzes favorites. Concretament, volem emmagatzemar el nom de les pizzes, el preu, si és vegetariana i el restaurant on la podem demanar.

A continuació es detallen les tasques que haurem de seguir per desenvolupar aquesta aplicació.

1. (0,25 punts) Crea una nova aplicació web de Node.js que incorpori el *framework Express*, el motor de vistes *pug* i l'ODM *mongoose*.
2. (0,5 punts) Defineix accés als arxius estàtics que farà servir l'aplicació, centralitzant l'accés en un directori al que anomenaràs *public*.

Col·loca a l'interior del directori *public* els directoris *images* i *css* que obtindràs al descomprimir l'arxiu adjunt número 1.

3. (0,25 punts) Crea la base de dades que gestionarà l'aplicació, important els arxius JSON que s'obtenen al descomprimir l'arxiu adjunt número 2. Concretament, s'han de crear les col·leccions *restaurantes* i *pizzas*.

A l'hora d'importar les dades, comprova el tipus de dada que s'assigna a cadascuna dels camps.

4. (1 punt) Defineix dos enrutaments a l'arxiu principal de l'aplicació:
- */ (a través del mètode GET)*: s'ha de visualitzar la vista *mensaje.pug* (obtindràs aquest arxiu al descomprimir l'arxiu adjunt número 3)
  - ***Qualsevol altra ruta*** (a través de qualsevol mètode): s'ha de visualitzar la vista *mensaje.pug* (obtindràs aquest arxiu al descomprimir l'arxiu adjunt número 3)
5. (1 punt) Modifica els enrutaments creats al punt anterior (i també la vista *mensaje.pug*) per tal d'enviar com a paràmetre a la vista el missatge que s'ha de mostrar:
- */ (a través del mètode GET)*: en lloc de mostrar el text *Mensaje* s'haurà de mostrar el text *Hola, Bienvenido/a*
  - ***Qualsevol altra ruta*** (a través de qualsevol mètode): en lloc de mostrar el text *Mensaje* s'haurà de mostrar el text *ERROR 404: SITIO NO ENCONTRADO*

6. (0,5 punts) Afegeix a l'aplicació un esquema de dades per gestionar les dades dels restaurants. L'esquema estarà format per les següents propietats, atenent als requeriments que s'hi detallen:

- *nombre*: s'emmagatzemaran cadenes de text. És una propietat obligatòria

7. (1 punt) Afegeix a l'aplicació un esquema de dades per gestionar les dades de les pizzes. L'esquema estarà format per les següents propietats, atenent als requeriments que s'hi detallen:

- *nombre*: s'emmagatzemaran cadenes de text. És una propietat obligatòria. Longitud màxima de 20 caràcters
- *precio*: s'emmagatzemaran valors numèrics. És una propietat obligatòria. El valor mínim és 1
- *vegetariana*: s'emmagatzemaran cadenes de text. És una propietat obligatòria. Només s'admeten els valors *si* o *no*
- *restaurante*: s'emmagatzemaran objectes identificadors que referencien documents de la col·lecció *restaurantes*. És una propietat obligatòria

8. (1,5 punts) Crea un arxiu de rutes on es definiran els enrutaments als recursos `/pizzas`. Concretament s'han d'atendre les següents sol·licituds:

- **`/pizzas`** (a través del mètode *GET*): s'ha de visualitzar la vista `pizzas.pug` (obtindràs aquest arxiu al descomprimir l'arxiu adjunt número 3), on es mostrarà un llistat amb les dades de totes les pizzes. A l'arxiu `pizzas.pug` es mostra, a mode d'exemple, com s'han de distribuir les dades de les pizzes.

Des d'aquest llistat l'usuari ha de poder eliminar qualsevol de les pizzes existents clicant el botó *Eliminar* (substitueix els 0s, de la ruta `/pizzas/delete/00000000000000000000000000`, per l'identificador de cada pizza).

- **`/pizzas/delete/id`** (a través del mètode *GET*): s'ha d'eliminar la pizza identificada amb `id`. Tan si l'eliminació és satisfactòria, com si no, es redireccionarà a l'usuari cap al recurs `/pizzas`. En cas d'error, això sí, es mostrarà un missatge per consola

9. (1 punt) Desenvolupa l'API de l'aplicació, tenint en compte que totes les sol·licituds i respostes es realitzaran en format JSON. Concretament, l'API ha d'atendre les següents sol·licituds:

- **`/api/id`** (a través del mètode *GET*): ha de retornar les dades de la pizza identificada amb `id`. Si es produeix algun error, la resposta serà **`respuesta: 'error'`**

Al definir l'enrutament s'ha de garantir que `id` és una col·lecció de 24 dígits numèrics.

- **/api** (a través del mètode *POST*): ha d'afegir una nova pizza. Si la nova pizza es crea satisfactòriament, la resposta serà **respuesta: 'ok'**. Si es produeixen errors, es retornarà un *array* amb els missatges de tots els errors que s'han produït

10. (1 punt) Modifica l'esquema de dades que gestiona les dades de les pizzes afegint una propietat virtual que anomenaràs *precio/IVA*. Aquesta propietat permetrà accedir al preu de la pizza amb l'IVA inclòs (que és del 21%).

Posteriorment, modifica la vista *pizzas.pug* aconseguint que es mostri el preu amb IVA en una nova columna de la taula.

11. (1 punt) Modifica l'enrutament **/pizzas** (a través del mètode *GET*) per tal d'aconseguir incorporar el nom del restaurant relacionat a cadascuna de les pizzes.

Després, modifica la vista *pizzas.pug* aconseguint que es mostri el nom del restaurant en una nova columna de la taula.

12. (1 punt) Modifica l'enrutament **/pizzas** (a través del mètode *GET*) per tal d'aconseguir que les pizzes es mostrin ordenades pel nom.