Al contrari que altres pràctiques, aquesta conté un parell de preguntes amb una part que s'ha de respondre en aquest document

ACTIVITAT
Objectius: - Practicar conceptes relacionats amb construcció, implementació, i integració de components de programari específics per a la persistència de dades, incloent la seva especificació, empaquetatge, i desplegament.
Instruccions: - Responeu a l'espai de cada pregunta, si ho feu amb diapositives enganxeu la diapositiva en aquest mateix espai. - Es valorarà la cura en la presentació del document i que segueixi l'estructura indicada.
Criteris d'avaluació: - Cada pregunta té el mateix pes - Es valorarà la presentació i els comentaris al codi
Entrega: - Aquest document anomenat **memoria.pdf** i el codi corresponent

Repositoris de referència:
https://github.com/jpala4-ieti/DAM2-MP06-UF04-Java-Base.git
https://github.com/jpala4-ieti/DAM2-MP06-UF04-NodeJS-Base.git

Noms i Cognoms:
Repositori GitHub amb exercicis resolts:

Preparació de l'activitat

Clona els dos repositoris que t'hem facilitat com a base per realitzar els exercicis.

Exercicis

Exercici 1. GitHub Actions (2 punts)

GitHub Action: Projecte de Java amb Maven

Amb ajuda d'aquesta guia
https://docs.google.com/presentation/d/1-SXV1_WuHSHnNmQJpALZRrQIjGu3r4-fdU0k9R5QIkg/edit#slide=id.g2cb25062a1e_0_0

Modifica el README.md per tal que el badge mostri la informació del teu repositori i no de l'original.

Assegura't que l'acció s'executa correctament.

URL del repositori: https://github.com/calvarez123/actionsUF4_Pr4

Explica quina part del fitxer pom.xml és la que permet empaquetar l'aplicació en format .jar

Dentro del pom.xml, la parte que importa es el bloque <build>. Ahí es donde se configura cómo empacar la aplicación en un .jar. En este caso, se usa el plugin maven-assembly-plugin que se encarga de hacer el trabajo pesado. Con esta configuración, el plugin crea un .jar que incluye todas las dependencias necesarias.

Exercici 2. Modificació aplicació NodeJS

Repositori base https://github.com/jpala4-ieti/DAM2-MP06-UF04-NodeJS-Base.git

Apartat 2.1. Pre-requisits (2 punts)

Què has de fer:

- Llegir el fitxer ***README.md***
- Posar en marxa mongodb i la interfície gràfica per gestionar-lo amb docker-compose. És exactament el mateix que a la UF3.
- Crear les BD a través de la interfície web **`dam2-mp06-uf04`** **`i`** **`dam2-mp06-uf04-test`**
- Modifica ***./src/server.js*** per canviar el missatge d'arrencada i que aparegui el teu cognom i les teves inicials.
- Arrencada de l'API executant `npm start`

Apartat 2.2. Importació de dades i correcció d'error (2 punts)

Executa aquests dos scripts

```
node src/utils/import_users.js
node src/utils/import_posts.js
```

El d'importació d'usuaris funciona correctament i utilitza el model definit a

```
const User = require('../api/models/user');
```

El d'importació de post provoca duplicats si s'executa múltiples vegades.
I la definició del model està en el mateix fitxer ***import_posts.js***

Què has de fer:

- Corregeix l'error d'inserció de duplicats fixant-te en ***import_users.js***
- Acaba movent la definició del model a dins de ../api/models/post

Apartat 2.4. Tests unitari i d'integració (2 punts)

Executa `npm test` des de l'arrel del projecte

Corregeix l'error

```
PASS test/integration/events.test.js
  Event Integration Tests
    √ should create a new event and retrieve it (113 ms)

Test Suites: 1 failed, 1 passed, 2 total
Tests:       1 failed, 2 passed, 3 total
Snapshots:   0 total
Time:        1.596 s, estimated 2 s
```

Explica:
Quina eina es fa servir en aquest projecte per realitzar els tests? Justifica la resposta
A on està configurada?
Què fa exactament el test d'integració del que parla el resultat?

(Pista: comença mirant les referències a test en el fitxer package.json)

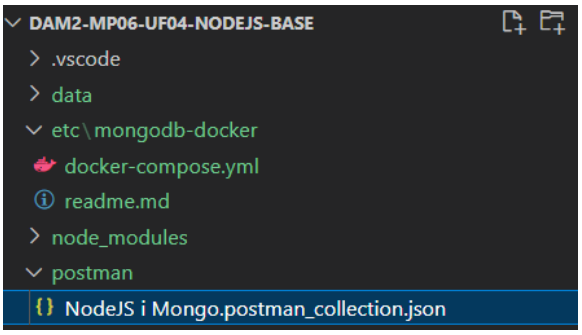
En esta aplicación se utiliza la herramienta jest para hacer los test.
esta configurada en jest.config.js

el test esta evaluando una funcion llamada calculateAverageAge que calcula la media de edad de unos usuarios. pero no esta dando el numero que debería de dar.

Apartat 2.3. Refactorització de l'API (4 punts)

Què has de fer:

- Importa la col·lecció de postman



- Amb la API functionant, fes la prova d'inserció d'un event i comprova que ha funcionat correctament.
    - Com a alternativa pots usar curl

```
curl -X POST http://localhost:3000/api/events \
-H "Content-Type: application/json" \
-d '{"name":"Event Name","date":"2023-04-05T09:00:00Z","description":"Event Description"}'
```

- Modifica el codi d'app.js per aconseguir que aquesta endpoints (continua després de la imatge…)

```
// POST endpoint per inserir un esdeveniment
app.post('/api/events', async (req, res) => {
  try {
    const event = new Event(req.body);
    await event.save();
    res.status(201).send(event);
  } catch (err) {
    res.status(400).send(err.message);
  }
});

// Endpoint per recuperar un esdeveniment per ID
app.get('/api/events/:id', async (req, res) => {
  try {
    const event = await Event.findById(req.params.id);
    if (!event) {
      return res.status(404).send("L'esdeveniment no s'ha trobat.");
    }
    res.send(event);
  } catch (err) {
    res.status(500).send(err.message);
  }
});
```

- Funcionin de la mateixa forma que el de users

```
const userRoutes = require('./api/routes/userRoutes');
…
app.use('/api', userRoutes);
```

En acabar la refactorització, comprova que els tests continuen funcionant correctament.