NodeJS examen

🌟 Doel

Ontwikkel een **RESTful API** met **Node.js**, **Express** en **Mongoose** om codefragmenten op te slaan, te taggen en te delen.

📄 API Routes

📩 POST /api/snippets

- Voeg een nieuw codefragment toe.
- Vereist een JSON-body met minimaal de volgende velden:

```
{
  "title": "Async Function Example",
  "code": "async function fetchData() { return await fetch(url); }",
  "language": "JavaScript",
  "tags": ["async", "fetch", "JavaScript"],
  "expiresIn": 3600
}
```

- `expiresIn` is optioneel en geeft aan hoe lang (in seconden) het snippet bewaard blijft.
- Retourneert het opgeslagen codefragment met de **id**.
- Encode het codefragment voor het opslaan met onderstaande code (Om problemen met double quotes te vermijden)

📜 GET /api/snippets

- Haal alle codefragmenten op.
- Ondersteunt **filtering** via queryparameters:
  - `?language=JavaScript` → Filter op programmeertaal. (case-insensitive)
  - `?tags=async,fetch` → Filter op een of meerdere tags. (case-insensitive)
- Ondersteunt **paginering** en **sortering**:
  - `?page=1&limit=10` → Haal 10 resultaten per pagina op.
  - `?sort=createdAt&order=desc` → Sorteer snippets op datum (nieuwste eerst).
- De vervallen codefragmenten niet meer tonen.
- Decodeer alle codefragmenten met onderstaande code alvorens te tonen

---

🔍 GET /api/snippets/:id

- Haal een specifiek codefragment op via zijn unieke ID.
- Retourneert ook een **geschiedenis van wijzigingen**. (Optioneel)
- Het vervallen codefragment niet meer tonen.
- Decodeer het codefragment met onderstaande code alvorens te tonen

---

🖋️ PUT /api/snippets/:id

- Werk een bestaand snippet bij.
- De vorige versie wordt opgeslagen in een **versiegeschiedenis**. (Optioneel)

🗑️ DELETE /api/snippets/:id

- Verwijder een codefragment op basis van zijn ID.

💡 Structuur

Werk met een gestructureerde opzet door gebruik te maken van **models, controllers en routes**:

- **Models**: Definieer de database-schema's met **Mongoose**. (Elk model heeft ook een createdAt en updatedAt)
- **Controllers**: Beheer de logica van de API en verwerk aanvragen.
- **Routes**: Bepaal de API-endpoints en verbind ze met de controllers.

📊 Dashboard (EJS) /

- Bouw een route waar een tabel getoond word met alle codefragmenten => title - code - language - tags
- Zorg voor filter per "language" en "tags"
- Maak gebruik van EJS template engine
- Orden de nodige Frontend JS en CSS in de public folder
- Zie voorbeeld hieronder

📝 Deployment

1. **Code publiceren**:
   - Push de code naar **GitHub** met commits **per stap.** (minimum **10** commits)
   - Voeg een **README** toe met uitleg over de API.
2. **Database opzetten**:
   - Maak een **MongoDB-database** aan (**MongoDB Atlas**).
3. **API online zetten**:
   - Deploy de API via **Render**.

💌 Indienen

- Dien de **GitHub-repository link** en de **Render live URL** in via **Syntra Cloud**.

🧰 Problemen met double quotes vermijden

```
// Encode
const encodedCode = Buffer.from(HIER-DE-SNIPPET).toString("base64");
// Decode
const decodedCode = Buffer.from(HIER-DE-SNIPPET, "base64").toString("utf-8");
```
