Curso de Back-End con Node.js (Avanzado)

Proyecto Final

Proyecto Final

Proyecto Final (1/9)

El proyecto final de curso consistirá en construir una API para un “clon” de la red social Twitter.

En la misma, se deberá dar soporte para lectura, escritura, edición y borrado (CRUD) de:

- Usuarios.
- Tweets.

La API, además, brindará un sistema de autenticación.

Proyecto Final (2/9)

La forma en la que se hará el ejercicio será incremental. Primero nos enfocaremos en crear Usuarios y Tweets.

Tendremos *endpoints* para lectura, escritura, edición y borrado de ambas colecciones.

En el caso de los *tweets*, además, se deberá brindar soporte para buscarlos por texto, filtrarlos por autor, ordenarlos por fecha y paginarlos.

En caso de los autores, se deberá poder buscarlos por texto.

Proyecto Final (3/9)

Tanto los esquemas de *tweets* como de usuarios deberán ofrecer validaciones para todos sus campos.

A su vez, los *endpoints* deberán requerir autenticación según corresponda.

Tener especial atención a las validaciones de autoría: Un usuario sólo puede escribir *tweets* asociados a *sí mismo*, y así para todos los casos que corresponda.

Proyecto Final (4/9)

Para probar la API, se hará uso de una Front-End app creada por los docentes para este ejercicio: https://ha-node-proyecto-final-front-end.vercel.app. La idea es poder consumir la API desde una Interfaz de Usuario, que es lo que suele suceder en la “vida real”.

Además se podrá seguir usando Insomnia para hacer tests.

Dado que ahora la API será consumida desde una Front-End app en un navegador será necesario agregar soporte CORS: https://github.com/expressjs/cors#installation

Proyecto Final (5/9)

Para que la Front-End app funcione, la API debe respetar el “contrato” definido en las siguientes diapositivas.

Dicho contrato es un mínimo de lo que debe cumplir la API.

Nota: Se asume que todos los documentos obtenidos a través de la API incluyen el `id` agregado por MongoDB. Además del “contrato”, cada alumno podrá agregar *endpoints* y funcionalidades adicionales (que consideren pertinentes), por más de que no vayan a ser consumidos desde la Front-End app de prueba.

Proyecto Final (6/9)

POST /users

Body:

```
{
  email: String,
  username: String,
  password: String
}
```

Response:

```
{
  token: String,
  user: {
    email: String,
    username: String,
    id: String
  }
}
```

Proyecto Final (7/9)

POST /sessions

Body:

```
{
  email: String,
  username: String,
  password: String
}
```

Response:

```
{
  token: String,
  user: {
    email: String,
    username: String,
    id: String
  }
}
```

Proyecto Final (8/9)

`GET /tweets`

Body:

(vacío)

Response:

```
[
  {
    text: String,
    author: { username: String }
  }
]
```

Proyecto Final (9/9)

🔒 **`POST /tweets`**

Body:

```
{ text: String }
```

Response: