PROYECTO SAAWIT

SAAWIT: API que permite gestionar noticias colaborativas, donde los usuarios puedan registrarse y publicar una noticia en una serie de categorías temáticas fijas. Segundo proyecto realizado para Hackaboss usando tecnología javascript, Node.js, MySQL y Postman.

Proyecto realizado por: Ana Arévano, Muchika Chettakul y Juan Garrido Troche.

Contenido

SAAWIT

En este segundo proyecto de Hackaboss tendremos que desarrollar nuestro conocimiento acerca de las siguientes tecnologías: JavaScript, Node.js, MySQL y Postman. Ya que es un proyecto de noticias colaborativas tipo Reddit hemos investigado acerca del origen del nombre. Éste viene del acortamiento de la frase I already read it (ya lo he leído) por lo que hemos sugerido varios nombres parecidos buscando sinónimos de leído como checkeddit, seeit, sawit hasta llevarlo a saawit: as I saw it (como yo lo vi).

Título

Web de noticias colaborativas.

Descripción

En SAAWIT, implementaremos una API que permita gestionar noticias colaborativas, estilo a la aplicación Reddit o menéame, donde los usuarios puedan registrarse y publicar una noticia en una serie de categorías temáticas fijas.

Usuarios anónimos

- Visualizar la lista de últimas noticias del día ordenadas por valoración.
- Visualizar noticias de días anteriores.
- Filtrado por:
    - Tema
- Login.
- Registro:
    - Nombre
    - Email
    - Biografía
    - Foto

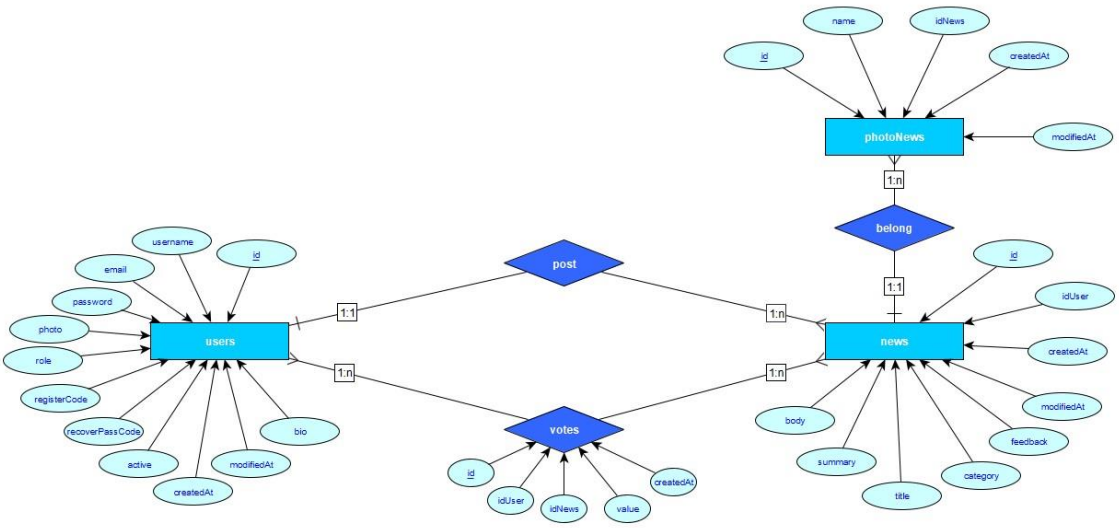
Usuarios registrados

- Lo mismo que los anónimos
- Publicar una nueva noticia:
    - Título.
    - Foto (opcional).
    - Entradilla.
    - Texto de la noticia.
    - Tema.

- Editar una noticia publicada por el mismo usuario.
- Borrar una noticia publicada por el usuario.
- Votar positivamente o negativamente otras noticias.
- Opcional:
    - Gestión del perfil de usuario (Nombre, Email, Biografía, Foto, …)

BASE DE DATOS

Para realizar la base de datos hemos utilizado la herramienta yEd Graph Editor, donde hemos creado las entidades con los siguientes atributos:



ENTIDADES

Users:

- Id: Clave primaria, Identificador único, autoincremental y entero de cada registro.
- username: Tipo texto, VARCHAR con un máximo de 50 caracteres, requerido.
- email: Tipo texto, VARCHAR de un máximo de 100 caracteres, requerido y obligatorio.
- password: Tipo texto, VARCHAR de un máximo de 100 caracteres, requerido y obligatorio.
- photo: Tipo texto, VARCHAR (100), opcional.
- role: Solo las opciones de: admin, mod(puede bloquear usuarios o eliminar post si no corresponden a la categoría pueden cambiar el perfil de user a mod) y user (por defecto).
- recoverPassCode: Código de recuperación de contraseña: VARCHAR (20)
- active: El usuario puede darse de alta pero hasta que no introduce el código de registro no se activa su cuenta. Por defecto vendrá como false. BOOLEAN.
- createdAt: Fecha de creación del usuario. Timestamp y requerido.

- modifiedAt: Fecha de modificación de alguno de los atributos. Timestamp.
- bio: Biografía del usuario. VARCHAR(500)

news:

- id: Clave primaria, Identificador único, autoincremental, solo puede ser un número entero y positivo.
- idUser: Clave foránea de la tabla user y número entero.
- createdAt: Fecha de creación del usuario. Timestamp y requerido.
- modifiedAt: Fecha de modificación de alguno de los atributos. Timestamp.
- category: categorías de temática fija: [deportes, videojuegos, noticias, programación, viajes, tecnología, música, memes]
- feedback: total votos (positivos – negativos). Entero y puede ser positivo o negativo. Por defecto es cero.
- title: Título de la noticia, obligatorio, VARCHAR(100)
- photo: Imagen de la noticia, opcional, VARCHAR(100)
- summary: Entradilla de la noticia, opcional, VARCHAR(250)
- body: Texto de la noticia, obligatorio, MEDIUMTEXT.

photoNews:

- id: Clave primaria, Identificador único, autoincremental y entero de cada registro.
- idNews: Clave foránea de la tabla user y número entero.
- name: Nombre de la imagen creado de manera automática, obligatorio, VARCHAR(100)
- createdAt: Fecha de creación del usuario. Timestamp y requerido.
- modifiedAt: Fecha de modificación de alguno de los atributos. Timestamp.

RELACIONES

Votes: Un usuario puede votar de 1 a n noticias (1:n) y 1 noticia puede ser votada entre 1 y n usuarios (1:n). Como existen n noticias y usuarios (n:m) esta relación se convierte en una tabla incluyendo los atributos de id de ambas entidades (users y news):

- Id: Clave primaria, Identificador único, autoincremental y entero de cada registro.
- IdUser: Clave foránea de la tabla user, obligatorio y número entero.
- idNews: Clave foránea de la tabla news, obligatorio y número entero.
- value: positivo o negativo. Obligatorio y BOOLEAN.
- createdAt: Fecha de creación del usuario, obligatorio y Timestamp.

Post: Un usuario puede publicar de 1 a n noticias (1:n) y 1 noticia solo puede ser publicada por un usuario (1:1). Por lo tanto, el id de usuario debe añadirse a la tabla de noticias. No es necesario crear una tabla para esta relación.

Belong: Una noticia puede tener entre 0 y 3 fotografías (1:n) y 1 foto puede pertenecer solo a una noticia (1:1). Por lo tanto, el id de noticia debe añadirse a la tabla de photoNews. No es necesario crear una tabla para esta relación.

CREACIÓN DE TABLAS EN MYSQL WORKBENCH

Las tablas se crearán dentro de la base de datos llamada saawit que, a su vez, se incluirá dentro del proyecto dentro del archivo bbdd/initDB.js:

- CREATE DATABASE IF NOT EXISTS saawit;
- USE saawit;

  Se eliminan las tablas de forma inversa a como se han creado
- DROP TABLE IF EXISTS photoNews;
- DROP TABLE IF EXISTS votes;
- DROP TABLE IF EXISTS news;
- DROP TABLE IF EXISTS users;

- CREATE TABLE IF NOT EXISTS users (

  id INT UNSIGNED PRIMARY KEY AUTO_INCREMENT,

  username VARCHAR(100) UNIQUE NOT NULL,

  email VARCHAR(100) UNIQUE NOT NULL,

  password VARCHAR(100) NOT NULL,

  bio VARCHAR(500) NOT NULL,

  photo VARCHAR(100),

  role ENUM('admin', 'mod', 'user') DEFAULT 'user',

  registrationCode VARCHAR(100),

  recoverPassCode VARCHAR(20),

  active BOOLEAN DEFAULT false,

  createdAt TIMESTAMP NOT NULL,

  modifiedAt TIMESTAMP);

- CREATE TABLE IF NOT EXISTS news (

  id INT UNSIGNED PRIMARY KEY AUTO_INCREMENT,

  category ENUM('deportes', 'videojuegos', 'noticias', 'programación', 'viajes', 'tecnología', 'música', 'memes', 'general') DEFAULT 'general',

  score INT DEFAULT 0,

  idUser INT UNSIGNED NOT NULL,

  FOREIGN KEY (idUser) REFERENCES users(id),

  title VARCHAR(100) NOT NULL,

  photo VARCHAR(100),

  summary VARCHAR(250),

  body MEDIUMTEXT NOT NULL,

  createdAt TIMESTAMP NOT NULL,

  modifiedAt TIMESTAMP);

- CREATE TABLE IF NOT EXISTS votes (

  id INT UNSIGNED PRIMARY KEY AUTO_INCREMENT,

  value BOOLEAN NOT NULL DEFAULT false,

  idUser INT UNSIGNED NOT NULL,

  FOREIGN KEY (idUSer) REFERENCES users(id),

  idNews INT UNSIGNED NOT NULL,

  FOREIGN KEY (idNews) REFERENCES news(id),

  createdAt TIMESTAMP NOT NULL);

- CREATE TABLE IF NOT EXISTS photoNews (

  id INT UNSIGNED PRIMARY KEY AUTO_INCREMENT,

  name VARCHAR(100) NOT NULL,

  idNews INT UNSIGNED NOT NULL,

  FOREIGN KEY (idNews) REFERENCES news(id),

  createdAt TIMESTAMP NOT NULL,

  modifiedAt TIMESTAMP);

INTRODUCCIÓN

Antes de instalar alguna dependencia crearemos nuestro fichero package.json con la siguiente línea de comandos, situada en la carpeta de nuestro proyecto:

- npm init
- npm init -y (si queremos crear package.json con valores por defecto)

En nuestro proyecto queda del siguiente modo:

```
{
  "name": "saawit",
  "version": "1.0.0",
  "description": "API que permita gestionar noticias
colaborativas",
  "main": "server.js",
  "scripts": {
    "dev": "nodemon server.js
  },
  "repository": {
    "type": "git",
    "url": "git+https://github.com/JuanGarridoTroche/saawit.git"
  },
  "author": "Juan Garrido Troche, Muchika Chettakul, Ana Arévano",
  "license": "ISC",
  "bugs": {
    "url": "https://github.com/JuanGarridoTroche/saawit/issues"
  },
  "homepage": "https://github.com/JuanGarridoTroche/saawit#readme",

}
```

DEPENDENCIAS NECESARIAS PARA SAAWIT

Las dependencias son aquellas aplicaciones o bibliotecas requeridas por otro programa para poder funcionar correctamente. Por ello, se dice que determinado programa "depende" de tal aplicación o biblioteca (library). Para instalarlos, emplearemos un sistema de gestión de paquetes de Node llamado "npm" (Node Package Management).

Existen 3 tipos de dependencias:

- Dependencias CORE: Aquellas que ya vienen instaladas y se pueden utilizar solo llamando al módulo, como, por ejemplo, path o fs.
- Dependencias: Aquellas necesarias que instalaremos para que nuestra API funcione de manera correcta. Ejemplo: mysql2, dotenv, bcrypt, etc.
- Dependencias para desarrolladores(-D): Aquellas dependencias que nos harán la programación más fácil como control de sintaxis (eslint), formateo del código para que se vea ordenado (prettier) o reinicio de nuestro servidor cada vez que guardamos un cambio (nodemon). Este tipo de dependencias se instalan añadiendo un -D en la línea de comandos. En cuanto finalice nuestro proyecto y lo pongamos en producción se podrían desinstalar (npm uninstall mi_DevDependence).

A continuación, vamos a indicar que dependencias son las que vamos a utilizar en nuestro proyecto:

NOMBRE	INSTALACIÓN	TIPO	USO
express	npm i express	Dependencia	Nos permite levantar un servidor. Además nos permite leer los datos enviados desde el body en formato raw -JSON.
dotenv	npm i dotenv	Dependencia	Nos permite leer los valores de las variables del archivo oculto .env
mysql2	npm i mysql2	Dependencia	Crea una conexión con una base de datos y una vez conectados, podemos hacer "crud" de tablas y/o registros pero nunca crear la base de datos.
bcrypt	npm i bcrypt	Dependencia	Permite encriptar y desencriptar datos.
nodemon	npm i nodemon -D	Dev	Monitoriza los cambios en el código fuente que se está desarrollando y automáticamente reinicia el servidor que está corriendo sin tener que reiniciarlo.
jsonwebtoken	npm i jsonwebtoken	Dependencia	Nos crea un token de seguridad estándar para transmitir de forma segura en internet, por medio del formato JSON.

fileUpload	npm i express-fileupload	Dependencia	Decodifica (o deserializa) los datos enviados desde el body con el formato “form-data”.
fs	No hay que instalar	Core	Gestiona los ficheros.
path	No hay que instalar	Core	Gestiona las rutas de nuestros ficheros.
sharp	npm i sharp	Dependencia	Sirve para manipular nuestras imágenes como redimensionar o cambiar a otro tipo de extensión.
uuid	Npm i uuid	Dependencia	Renombra los ficheros que subamos a nuestra bbdd.
joi	Npm i @hapi/joi	Dependencia	Sirve para validar todo tipo de datos introducidos por el usuario.
randomstring	npm i randomstring	Dependencia	Genera cadenas de caracteres alfanuméricos. En nuestro caso lo utilizaremos para generar el código de recuperación.
nodemailer	npm i nodemailer	Dependencia	Nos permite enviar correos a través de una configuración sencilla.

RUTAS / ENDPOINTS

Un endpoint es una pasarela que conecta los procesos del servidor de la aplicación con una interfaz (API, Application Programming Interface), es decir, la ruta que hay que añadir a nuestra URL, a la que se envían peticiones.

Antes de empezar a programar nuestra API debemos establecer los endpoints necesarios para que nuestra aplicación sea funcional:

MÉTODO	RUTA	FUNCIÓN	USO	PERMISOS
ENDPOINTS USUARIOS				
POST	/users/login	loginUser.js	Login de usuario.	Todos los usuarios
POST	/users	newUser.js	Registrar un nuevo usuario.	Todos los usuarios
PUT	/users/password	editPassword.js	Editar password del usuario.	isAuth
PUT	/users/password/solicitude	sendRecoverPassword.js	Solicitud de nueva contraseña por email.	Todos los usuarios
PUT	/users/ password/recover	recoverPassword.js	Recuperación de contraseña.	isAuth, isImg
PUT	/users/photo	editPhoto.js	Editar nuestra foto de perfil.	isAuth, isImg
ENDPOINTS NOTICIAS				
GET	/news/top	topRankedNews.js	Lista de las últimas noticias del día ordenadas por valoración.	Todos los usuarios
GET	/news/date=?	newsByDate.js	Noticias de días anteriores.	Todos los usuarios
GET	/news?category=”general”	newsByCategory.js	Noticias filtradas por categoría.	Todos los usuarios
POST	/news	createNews.js	Crear una noticia.	isAuth
PUT	/news/:idNews	editNews.js	Editar una noticia ya creada.	isAuth
DELETE	/news/:idNews	deleteNews.js	Borrar una noticia publicada.	isAuth
POST	/news/:idNews	voteNews.js	Vota una noticia publicada (de otro usuario).	isAuth