


Mongoose Schemas

 gmoralesc.gitbooks.io/creando-apis-con-node-js-express-y-mongodb/content/persistencia-de-datos-con-mongodb/mongoose-schemas.html

Los *Schemas* de *Mongoose* brindan una funcionalidad y consistencia adicional a los modelos, hasta el momento no he validado con rigurosidad los datos enviados por el usuario y es aquí donde los *Schemas* nos pueden ayudar mucho, además agregan una gran cantidad de funcionalidades que veremos con detalle en las siguientes secciones.

Comenzamos por establecer las diferentes propiedades de cada uno de los atributos de nuestro modelo (`/server/api/v1/posts/model.js`):

```
const mongoose = require('mongoose');

const { Schema } = mongoose;

const fields = {
  location: {
    type: String,
    default: '',
    trim: true
  },
  photo_url: {
    type: String,
    required: true,
  },
  description: {
    type: String,
    default: '',
    trim: true
  },
};

const post = new Schema(fields, {
  timestamps: true,
});

module.exports = mongoose.model('post', post);
```

Revisemos cada uno de los cambios realizados en nuestro modelo:

- Extraemos el constructor del esquema (`Schema`) del objeto `mongoose` .
- Renombramos el antiguo objeto modelo como `fields` , ahora además de especificar el tipo de atributo, podemos asignarle un valor por defecto (`default`), efectuarle operaciones con `trim` para quitar los espacios en blanco y establecer si el atributo es requerido o no.
- Utilizamos los `fields` y las opciones para construir el `Schema` y almacenarlo en la variable `post`
- Agregamos la opción `timestamps` al `Schema` para que automáticamente nos añada los atributos de `createdAt` y `updatedAt` , el primero se establece una vez se guarda satisfactoriamente el documento y el segundo una vez se actualice

satisfactoriamente el documento.

Como estamos añadiendo validaciones básicas al modelo si el documento no puede guardarse porque le falta un atributo requerido después de la función `save` ejecutara la función `catch` en vez la función `then`, lo cual llegará a nuestro *middleware* de errores, pero no podemos decir que es un error *500 Internal Server Error*, en este caso es un *422 Unprocessable Entity*, la cual significa que no se pudo procesar la entidad debido a un error en este caso que le faltan campos requeridos, por lo tanto procedemos a modificar nuestro *middleware* de errores (`/server/index.js`):

...

```
app.use((err, req, res, next) => {
  let {
    statusCode = 500,
  } = err;
  const {
    message,
  } = err;

  if (err.message.startsWith('ValidationError')) {
    statusCode = 422;
  }

  logger.error(`Error: ${message}`);
  res.status(statusCode);
  res.json({
    error: true,
    statusCode,
    message,
  });
});
module.exports = app;
```

Más información:

- [Mongoose Schema](#)
- [Mongoose Schema Types](#)