**BASE DE DATOS**

**npm init -y**

- crea un fichero **package.json** indica que vamos a necesitar para ejecutar las dependencias.

**npm i express connect-flash bcryptjs express-handlebars express-session method-override mongoose passport passport-local**

- Instalación de los módulos necesarios para nuestra api

- Genera un fichero **package-lock.json** y una carpeta **node\_modules**

**npm install dotenv**

- Nos permite guardar variables en un archivo **.env**

**npm install nodemon**

**­**- Reinicia el servidor de forma automática cada vez que se produce un cambio

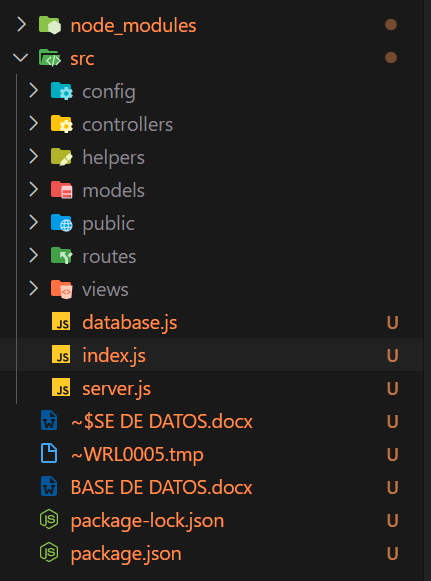
**npm install handlebars hbs**

- Corrige la lectura de datos en caso de error.

**npm install npm-check-updates**

- Permite chequear las actualizaciones de nuestros módulos

Construimos la siguiente estructura inicial:



**Database.js** (conexión a la base de datos)

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

*const*{NOTES\_APP\_MONGODB\_HOST, NOTES\_APP\_MONGODB\_DATABASE} = process.env;

*const* MONGODB\_URI = `mongodb://${NOTES\_APP\_MONGODB\_HOST}/${NOTES\_APP\_MONGODB\_DATABASE}`;

mongoose.connect (MONGODB\_URI, {

    useUnifiedTopology: true,

    useNewurlParser: true

})

    .then(*db* => console.log('Base de datos conectada'))

    .catch(*err* => console.log(err));

**Index.js** (configuración que inicia nuestra aplicación)

require('dotenv').config();

*const* app=require('./server');

require('./database');

app.listen(app.get('port'), () => {

     console.log('Server on port', app.get('port'))

})

**Server.js** (contiene toda la configuración que la relaciona el resto de ficheros de la API)

*const* express = require("express");

*const* path = require('path');

*// Inicializaciones*

*const* app = express();

*// Ajustes*

app.set('port', process.env.PORT || 4000);

app.set('views', path.join(\_\_dirname, 'views'));

*// Middlewares*

*// Variables globales*

app.use(express.static(path.join(\_\_dirname, 'public')));

*// Rutas*

app.get('/', (*req*, *res*) =>{

    res.send('hello world');

});

*// Ficheros estáticos*

*module*.*exports* = app;

En la carpeta **/models** (contiene la estructura que van a tener nuestras tablas de la base de datos)

**Nota.js**

*const* {Schema, model} = require('mongoose');

*const* NotaSchema = new Schema ({

    titulo: {

        type: String,

        required: true

    },

    descripcion: {

        type: String,

        required: true

    }

},

{

    timestamps: true

})

model('Nota', NotaSchema);

**usuario.js**

*const* { Schema, model } = require('mongoose');

*const* bcrypt = require('bcryptjs');

*const* UsuarioSchema = new Schema({

    nombre: { type: String, required: true },

    apellidos: { type: String, required: true },

    id\_usuario: { type: String, required: true },

    email: { type: String, required: true },

    password: { type: String, required: true }

}, {

    timestamps: true

});

UsuarioSchema.methods.encrypPassword = *async* *password* => {

*const* salt = await bcrypt.genSalt(10);

    return await bcrypt.hash(password, salt);

};

UsuarioSchema.methods.matchPassword = *function* (*password*) {

    return await bcrypt.compare(password, *this*.password);

}

*module*.*exports* = model('Usuario', UsuarioSchema);

En la carpeta **/views** (Se configuran las plantillas y documentos html que tendrán formato hbs, es decir irá la estética de la página).

En la carpeta **/routes** irán todas las rutas que tendrá nuestro proyecto cada una con una función diferente (se explica posteriormente)

En la carpeta **/controllers** irán todas las funciones de llamada a nuestras rutas.

**npm install morgan** (lista las peticiones que van llegando)

**npm rm handlebars –save** (Desistalar ultima versión de handlebars)

**npm i -D** [**handlebars@4.5.0**](mailto:handlebars@4.5.0)(instalar esta versión, no da errores con la base de datos)

**npm i @handlebars/allow-prototype-access** (para evitar el fallo durante la llamada de los datos a nuestro front-end)

npm **i** **@handlebars**/allow-prototype-access

En la carpeta **/config** irán todas las configuraciones más específicas como las siguientes:

En el fichero **Passport.js** irá la configuración para las sesiones de inicio de sesión del usuario y que cada usuario acceda a su perfil y sus datos correspondientes

En la carpeta **/helpers** irán todas las configuraciones comunes entre varias de nuestras rutas, en este caso ira la configuración necesaria para validar que una sesión es correcta y así conceder el permiso a una ruta o no.