# Introduccion

Aplicación de notas con autenticación de usuarios y rutas privadas

# Librerias / modulos

**Express** para el servidor

**Connect-flash** mensajes entre vistas, guarda la info en cookie o en sesión

**Bcryptjs** módulo de cifrado (como contraseñas)

**Express-handlebars** ViewEngine

**Express-session** guardar información en la memoria del server

**Method-override** enviar peticiones put o delete desde la vista

**Mongoose** manejar la bd mongo

**Passport** autenticación del usuario

**Passport-local** para usar bd local de autenticación (se puede usar Passport para autenticar con twitter, fb, google…)

## Modulos dev

**Dotenv** usar variables de entorno

**Nodemon** reinicia el server automáticamente

**Npm-check-updates** valida si hay actualizaciones

# Estructura

**Database.js** configuración de la base de datos

**Server.js** donde estará express

**Controllers** controladores

**Public** archivos estáticos como css

**Views** vistas

**Helpers** para ayudar a handlebars con los bucles y condicionales

## Server.js

Se crea el servidor y se exporta para usarse en index

const express = require('express');

const app = express(); //se crea el servidor

module.exports = app;

## index.js

Se importa el server

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

dotenv

require('dotenv').config();

Con esto, buscara un archivo llamado .env, lo leerá y lo asignará a la variable de entorno, para acceder a esta variable se ejecuta process.env.NOMBRE\_VARIABLE

require('./database'); //se ejecuta la BD

## Method override

Para poder usar el método delete, o put (router.delete), y poderlo enviar desde un formulario, se debe usar el middleware de method-override

//delete notes

router.delete("/notes/delete/:id", deleteNote);

Middleware

app.use(methodOverride('\_method'));

En el form que va a ir el método delete, debe ir de la siguiente manera:

 <form action="/notes/delete/{{\_id}}?\_method=DELETE" method="POST">

Y se debe agregar un input de tipo hidden en el formulario

          <form action="/notes/delete/{{\_id}}?\_method=DELETE" method="POST">

                <input type="hidden" name="\_method" value="DELETE">

## Connect-flash – Express session

Nos permite enviar mensaje entre vistas, se guardan en express sesión

Se deben importar en el server

const flash = require('connect-flash');

const session = require('express-session')

Y usarse como middleware

app.use(session({

    secret: 'Secretoseguro25448',

    resave: 'true',

    saveUninitialized: 'true'

}));

app.use(flash());

En el controlador, creamos el mensaje: req.flash

notesController.createNewNote= async (req,res)=>{

    //extraemos titulo y descripcion de la nota

    const {tittle, description} = req.body;

    //creamos una nota nueva, previamente importada, y le asignamos los valores del body, js asigna automaticamente los valores sin necesidad de escribir description: description

    const newNote = new Note({tittle, description});

    await newNote.save();//mongoose guarda la información en la bd

    req.flash('success\_mg', ' Note added suuccesfully'); //creamos el mensaje

    res.redirect('/notes')

};

En cada controlador está la ruta, pero para que se pueda acceder desde toda la aplicación a la variable success\_mg se define como variable global

app.use((req, res, next) =>{//el next se usa para que continue ejecutando lo que hay debajo

    res.locals.success\_msg = req.flash('success\_msg');

    next();

});

Cuando ejecuta el código, la variable **success\_mg** Obtiene el valor (note added… en este caso), y accedemos a ella

Se agrega un .hbs en partials, que valida si existe el mensaje, si si, lo muestra

{{#if success\_msg}}

<div class="alert alert-success alert-dismissible fade show">

    <strong> {{success\_msg}} </strong>

    <button type="button" class="close" data-bs-dismiss="alert">&times;</button>

</div>

{{!-- se crea el botón para cerrar el cuadro de dialogo, ell &times es para mostrar una x de cerrar --}}

{{/if}}