# Contenido

[1 Contenido 2](#_Toc101815661)

[Presentación 3](#_Toc101815662)

[Resumen 3](#_Toc101815663)

[Objetivo 3](#_Toc101815664)

[Finalidad del Manual 3](#_Toc101815665)

[Introducción 3](#_Toc101815666)

[1. Aspectos técnicos 4](#_Toc101815667)

[1.1 Herramientas utilizadas para el desarrollo 4](#_Toc101815668)

[1.1.1 GitHub 4](#_Toc101815669)

[1.1.2 Visual Studio Code 4](#_Toc101815670)

[1.1.3 Mongo DB 4](#_Toc101815671)

[1.1.4 Npm 4](#_Toc101815672)

[1.1.5 Node JS 4](#_Toc101815673)

[1.1.6 Express 5](#_Toc101815674)

[1.1.7 Express-session 5](#_Toc101815675)

[1.1.8 Heroku 5](#_Toc101815676)

[1.1.9 Bcrypt 5](#_Toc101815677)

[1.1.10 Hbs 5](#_Toc101815678)

[1.1.11 Joi 5](#_Toc101815679)

[1.1.12 Nodemon 5](#_Toc101815680)

[1.1.13 Passport 5](#_Toc101815681)

[1.1.14 Tailwind 5](#_Toc101815682)

[1.1.15 Chart JS 5](#_Toc101815683)

[2. Diagramas de modelamiento 5](#_Toc101815684)

[a. Registrar a un nuevo usuario por parte del Administrador 5](#_Toc101815685)

[I.2. Login de administrador y usuario 8](#_Toc101815686)

[43.1 Ver página de inicio por parte del administrador 10](#_Toc101815687)

[43.2 Ver pagina de inicio por parte del usuario 10](#_Toc101815688)

[44. Aspecto técnico del desarrollo del sistema 10](#_Toc101815689)

[2. 2. 1 Modificación local 10](#_Toc101815690)

[3. 3. Requerimientos del software 10](#_Toc101815691)

[1. 3.1 Requisitos mínimos 10](#_Toc101815692)

[4. Referencias 11](#_Toc101815693)

Contenido de tablas

[Tabla 1:Parámetros para la solicitud V](#_Toc101814403)

Contenido de ilustraciones

# Presentación

# Resumen

# Objetivo

# Finalidad del Manual

# Introducción

El manual se realiza con el fin de detallar el “NOMBRE DEL PROYECTO” en términos técnicos para que la persona que vaya a administrar, editar o configurar el software lo haga de una manera apropiada. El documento se encuentra dividido en las siguientes secciones.

Aspectos técnicos: Se darán a conocer conceptos, definiciones de los componentes del sistema desde el punto de vista teórico para mayor entendimiento por parte del lector sobre el funcionamiento del sistema de información e herramientas.

Aspecto técnico del desarrollo del sistema: Corresponde a la instrucción para el lector sobre los componentes del sistema desde una perspectiva técnica en los aspectos de almacenamiento de datos, estructura del desarrollo y recomendaciones del uso debido del sistema.

Requerimientos del software: Detalla los requerimientos básicos necesarios para el funcionamiento del software.

# Aspectos técnicos

El “NOMBRE DEL PROYECTO” tiene la finalidad de

## Herramientas utilizadas para el desarrollo

### GitHub

### Visual Studio Code

### Mongo DB

MongoDB es un **sistema de base de datos NoSQL orientado a documentos de código abierto y escrito en C++**, que en lugar de guardar los datos en tablas lo hace en estructuras de datos BSON (similar a JSON) con un esquema dinámico. Al ser un proyecto de código abierto, sus binarios están disponibles para los sistemas operativos Windows, GNU/Linux, OS X y Solaris.

Esto representa una de las diferencias más importantes con respecto a las bases de datos relacionales. Y resulta que no es que no es necesario seguir un esquema. Los documentos de una misma colección - concepto similar a una tabla de una base de datos relacional -, pueden tener esquemas diferentes.

### Npm

### Node JS

Node.js es un entorno controlado por eventos, diseñado para crear aplicaciones escalables, permitiéndote establecer y gestionar múltiples conexiones al mismo tiempo.

Las APIs de la biblioteca de Node.js son asíncronas, sin bloqueo. Un servidor basado en Node.js no espera que una API devuelva datos. El servidor pasa a la siguiente API después de llamarla, y un mecanismo de notificación de eventos ayuda al servidor a obtener una respuesta de la llamada a la API anterior.

Un subproceso escalable. Node.js utiliza un modelo de un solo subproceso con bucle de eventos. Gracias al mecanismo de eventos, el servidor responde sin bloqueos, esto hace que el servidor sea altamente escalable comparando con los servidores tradicionales como el Servidor HTTP de Apache.

### Express

### Express-session

### Heroku

### Bcrypt

### Hbs

### Joi

### Nodemon

### Passport

### Tailwind

### Chart JS

# Diagramas de modelamiento

El direccionamiento hace referencia a la determinación de cómo responde una aplicación a una solicitud de cliente en un determinado punto final, que es un URI (o una vía de acceso) y un método de solicitud HTTP específico (GET, POST, etc.).

Cada ruta puede tener una o varias funciones de manejador, que se excluyen cuando se correlaciona la ruta.

La definición de ruta tiene la siguiente estructura:

app.METHOD(PATH, HANDLER)

Donde:

app es una instancia de express.

METHOD es un método de solicitud HTTP.

PATH es una vía de acceso en el servidor.

HANDLER es la función que se ejecuta cuando se correlaciona la ruta.

## 2.1 Registrar a un nuevo usuario por parte del Administrador

Crear un nuevo usuario por parte del Administrador.

**Método**: POST

#### Parámetros para la solicitud

Tabla 1:Parámetros para la solicitud

|  |  |
| --- | --- |
| **nombre**  Localización: body  Nombre del usuario | Requerido  String |
| **apellido**  Localización: body  Apellido del usuario | Requerido  String |
| **email**  Localización: body  Email del usuario | Requerido  String |
| **password**  Localización: body  Contraseña del usuario | Requerido  String |

#### Código

var express = require('express');

var router = express.Router();

var {client, dbName} = require('../db/mongo');

var User = require('../models/models');

const bcrypt = require('bcryptjs')

/\* GET users listing. \*/

router.get('/', function(req, res, next) {

  res.render('signupAdmin');

});

router.post("/registro", function (req,res,next){

    regUser(req.body)

    .then(()=>{

      res.render('usersAdmin')

    })

    .catch((err)=>{

     console.log(err);

    })

    .finally(()=>{

      client.close;

    })

  });

async function regUser(datos){

    await client.connect();

    const db = client.db(dbName);

    const collection = db.collection('Usuarios');

    const hashUserPassword = await bcrypt.hash(datos.password, 10)

    await collection.insertOne(

      {

        nombre: datos.user,

        apellido: datos.name,

        email: datos.email,

        password: hashUserPassword

      }

    )

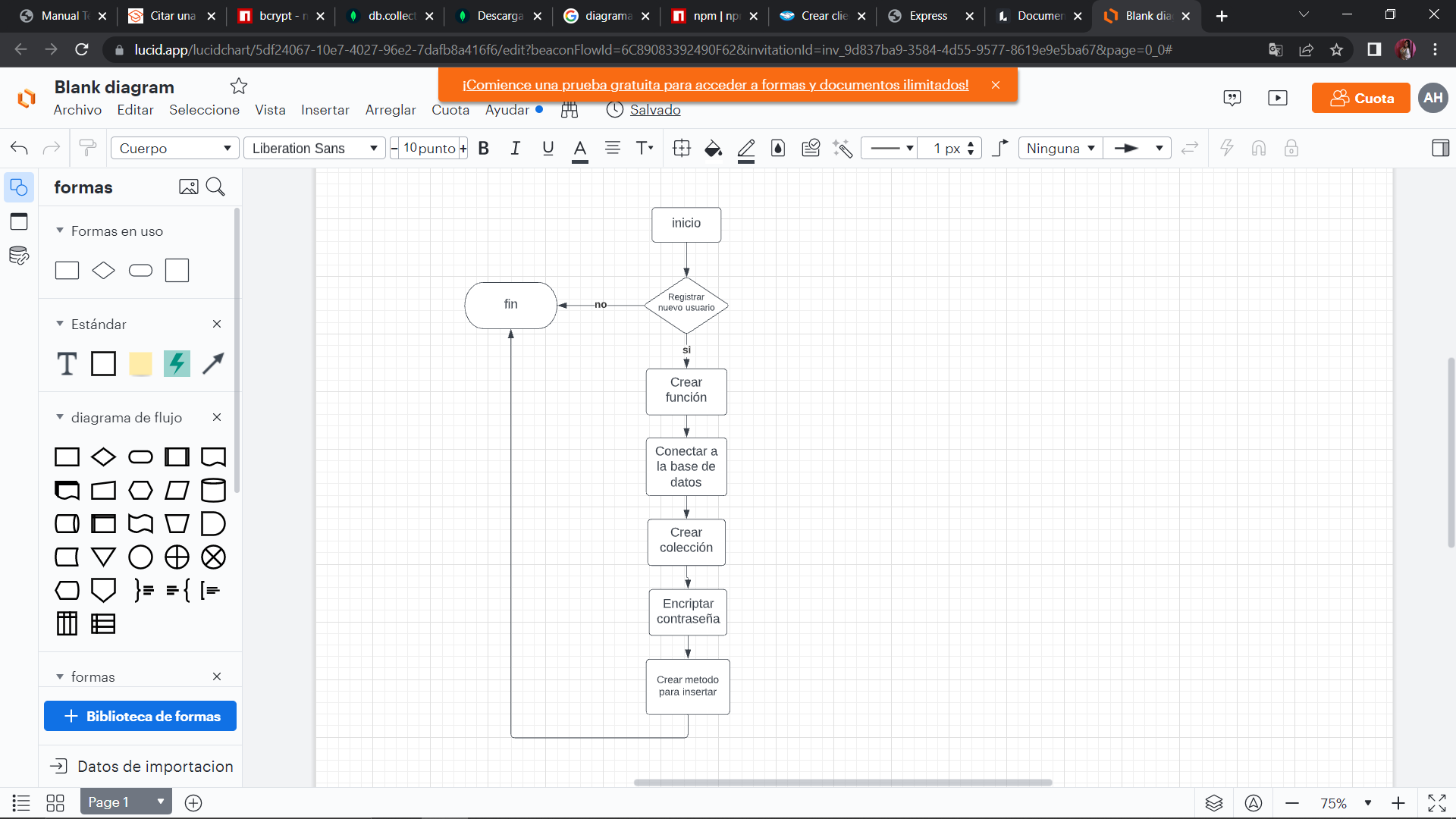
   }

module.exports = router;

En la ruta de registro se tiene dos métodos la primera ruta con el método GET para renderizar la vista donde se hará el registro de usuario.

La segunda ruta con el método POST es para obtener los datos de registro, mandados desde la función async *regUser,* donde se hace la conexión a la base de datos y a su colección *Usuarios* y con el método insertOne se insertar los datos que están dentro del objeto {nombre, apellido, email, password}, la contraseña esta encriptada mediante la librería bcrypy de npm.

#### Diagrama de flujo



## Login de administrador y usuario

Iniciar sesión mediante las credenciales previamente creadas por parte del Administrador.

Método: POST

Parámetros para la solicitud

|  |  |
| --- | --- |
| **nombre**  Localización: body  Nombre del usuario | Requerido  String |
| **apellido**  Localización: body  Apellido del usuario | Requerido  String |
| **email**  Localización: body  Email del usuario | Requerido  String |
| **password**  Localización: body  Contraseña del usuario | Requerido  String |

1. var express = require('express');
2. var router = express.Router();
3. var {client, dbName} = require('../db/mongo');
4. var User = require('../models/models');
5. const bcrypt = require('bcryptjs')
6. /\* GET users listing. \*/
7. router.get('/', function(req, res, next) {
8. res.render('signupAdmin');
9. });
10. router.post("/registro", function (req,res,next){
11. regUser(req.body)
12. .then(()=>{
13. res.render('usersAdmin')
14. })
15. .catch((err)=>{
16. console.log(err);
17. })
18. .finally(()=>{
19. client.close;
20. })
21. });
22. async function regUser(datos){
23. await client.connect();
24. const db = client.db(dbName);
25. const collection = db.collection('Usuarios');
26. const hashUserPassword = await bcrypt.hash(datos.password, 10)
27. await collection.insertOne(
28. {
29. nombre: datos.user,
30. apellido: datos.name,
31. email: datos.email,
32. password: hashUserPassword
33. }
34. )
35. }
37. module.exports = router;

## Ver página de inicio por parte del administrador

## Ver pagina de inicio por parte del usuario

# Aspecto técnico del desarrollo del sistema

En esta sección se procede a realizar una descripción detallada sobre los aspectos técnicos del aplicativo, relacionado con la instalación de las herramientas necesarias para realizar cualquier modificación requerida en el sistema.

## 2. 1 Modificación local

# 3. Requerimientos del software

En esta sección se detallan los requisitos mínimos del sistema para poder ejecutar los aplicativos usados para modificar “NOMBRE DEL PROYECTO”

## 3.1 Requisitos mínimos

Sistema Operativo: Windows 7

Procesador:

Memoria RAM:

Disco Duro:

Resolución de Pantalla:

Periféricos:

# Referencias

*What is GitHub*. (s. f.). W3 Schools. Recuperado 25 de abril de 2022, de https://www.w3schools.com/whatis/whatis\_github.asp

MongoDB. (s. f.). *¿Qué Es MongoDB?* Recuperado 25 de abril de 2022, de https://www.mongodb.com/es/what-is-mongodb