MANUAL TÉCNICO

Prácticas Iniciales “C” 2S-2024

Abdiel Fernando Josè Otzoy Otzin - 202300350

Diego Alessandro Constanza Padilla – 202300601

○ Requisitos del Sistema

Para poder observar el código e instalarlo, será necesario contar con los siguientes componentes:

- **Editor de texto:** Se recomienda utilizar un editor de texto para visualizar, editar e instalar las librerías del código fuente. Visual Studio Code es una opción popular y recomendada para este propósito.
- **Node.Js:** Es necesario tener instalado Node.js en el sistema. Es un entorno de ejecución de JavaScript que permite ejecutar código JavaScript fuera de un navegador web. Puede descargar e instalar la última versión de Node.Js desde su sitio web oficial.
- **NPM (Node Package Manager):** NPM es el gestor de paquetes de Node.js que se utiliza para instalar y gestionar las dependencias de los proyectos de Node.js. Se instala automáticamente junto con Node.js.
- **Conexión a internet estable:** Para poder utilizar las herramientas de la base de datos es necesario contar con internet estable para realizar las peticiones.

○ Requerimientos mínimos del IDE

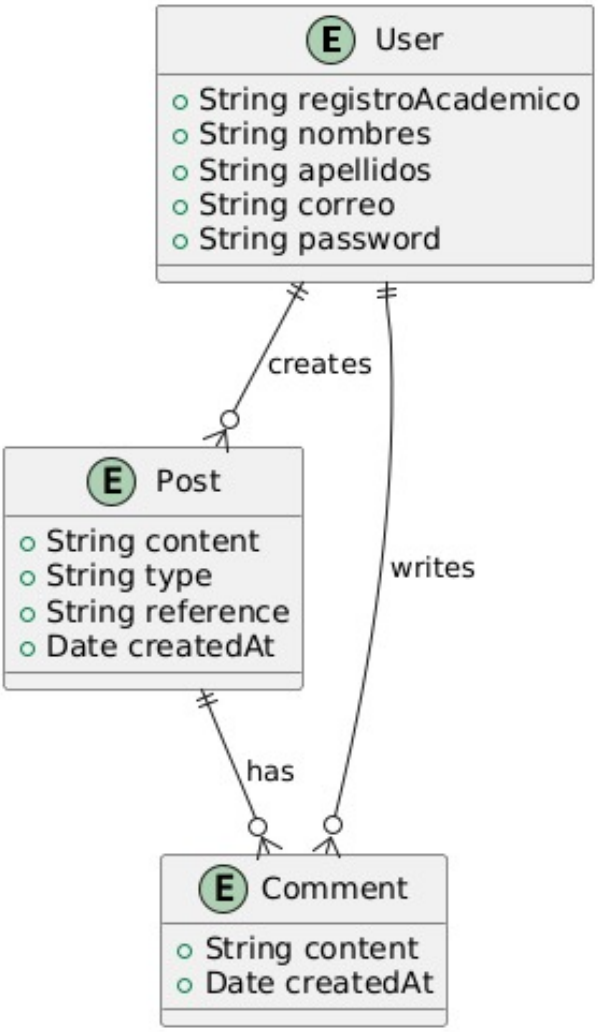
El entorno de desarrollo integrado (IDE) utilizado para trabajar con el código debe cumplir con los siguientes requisitos mínimos:

- **Node.js:** El IDE debe ser compatible con Node.js para ejecutar y depurar aplicaciones Node.js. Además, debe permitir la instalación de extensiones o complementos relacionados con Node.js.

- **Editor de Texto:** El IDE debe proporcionar funcionalidades básicas de edición de texto, como resaltado de sintaxis, autocompletado y capacidad de navegación en el código. En este caso, el programa fue desarrollado y probado en Visual Studio Code, un IDE popular que cumple con los requisitos mencionados anteriormente. Sin embargo, cualquier otro IDE que admita Node.js y ofrezca funcionalidades de edición de texto y uso de la terminal pueden ser utilizados para visualizar y editar el código.

○ Diagrama de Base de Datos

La base de datos fue implementada y usada en el entorno de MongoDB Atlas, cargado en la nube. El diagrama de relaciones utilizado en la base de datos es la siguiente:



○ Funciones dedicadas

Las funciones utilizadas en el programa hacen uso de un esquema donde el FrontEnd y el BackEnd comparten el mismo entorno, a su vez el uso de la Base de Datos utilizó una estructura estándar para conectar con la base de datos y utilizarla.

Models

Estos modelos son utilizados para manejar cada valor utilizando un esquema de objetos.

- **Modelo para posts (post.model.js):**

➔ *Esquema de comentarios:*

Contiene un esquema para comentarios donde requiere un contenido de tipo String, el esquema post, el esquema del usuario y la fecha de creación de tipo Date.

➔ *Esquema de publicaciones:*

Un esquema necesario para cada post, este contiene de manera interna el esquema por cada comentario que contiene, siendo un contenido de tipo String, el tipo que contiene si es “Curso” o “Catedràtico”, el esquema del usuario que lo envió, un esquema de comentarios que contiene todos los comentarios y la fecha de creaciòn de la publicaciòn.

- **Modelo para usuarios (user.model.js):**

➔ *Esquema de Usuarios:*

Existe un modelo para los usuarios registrados, estos modelos son utilizados al crear cada usuario, un registro académico de tipo String de manera ùnica, los nombres y apellidos de tipo String, el correo y una contraseña de tipo String, por último un listado de cursos los cuales han aprobado, cada uno de tipo String.

Actions

- **Acciones para posts (post.action.js):**

Para las acciones del post contiene diversas funciones que son utilizadas para poder realizar las labores del programa.

➔ *Crear Publicaciones “createPost”:*

Al crear una publicación se conecta a la base de datos, recibe el post enviado y lo crea utilizando la función *save* interna de MongoDB.

➔ *Obtener Todas las Publicaciones “getAllPosts”:*

Llama a las publicaciones obtenidas en la base de datos e itera mandando en un Json cada post.

➔ *Comentar una Publicaciòn Especìfica “commentPost”:*

Al recibir el comentario busca el Post de donde proviene y utilizando la función *findByIdAndUpdate* nativa de MongoDB y sus parámetros, guarda el comentario.

➔ *Filtrar Publicaciones por FIltro “getPostsByFilter”:*

Al recibir un filtro itera en las publicaciones y devuelve cada filtro por la referencia, un valor en específico.

➔ *Filtrar Publicaciones por Tipo “getPostsByType”:*

Itera en cada publicación y devuelve las publicaciones de un tipo, si es para cursos o para catedráticos.

- **Acciones para usuarios (post.user.js):**

Para las acciones del post contiene diversas funciones que son utilizadas para poder realizar las labores del programa.

➔ *Encriptar y Desencriptar para el inicio de sesiòn “encrypt” “decrypt”:*

Es utilizado para poder obtener la sesión creada al iniciar sesión y así poder verificar si está logueado.

➔ *Crear un usuario “createUser”:*

Con el usuario recibido crea una sesión, la encripta y envía el usuario para guardarla en la base de datos.

➔ *Iniciar Sesiòn “loginUser”:*

Con parte del usuario recibido verifica entre los usuarios si el Registro Académico existe, posterior si la contraseña es válida, si se verifica se redirige a la página principal, utilizando las funciones para desencriptar.

➔ *Cerrar Sesiòn “logout”:*

Formatea la sesión para poder volver a iniciar sesión y redirigè a la página para iniciar sesiòn.

➔ *Obtener la sesiòn “getSession”:*

Obtiene la sesión almacenada en ese momento, verificando que ingresó sesión correctamente.

➔ *Obtener el usuario de la sesiòn “getUserLogged”:*

Obtiene solo el usuario que ingresó sesión.

➔ *Mejorar Sesiòn “updateSession”:*

Extiende la sesión que está guardada en ese momento para poder ingresar a la página principal aùn cerrando la página.

➔ *Olvidar Contraseña “forgotPassword”:*

Recibe la petición para recuperar la contraseña, encuentra el usuario por el registro académico, verifica si el correo si es igual, si es así cambia la contraseña.

➔ *Obtener Usuario por ID “getUserById”:*

Con un id único retorna el usuario encontrado.

➔ *Mejorar Usuario por ID “updateUserById”:*

Al querer actualizar los datos de un usuario, recibe el usuario, lo encuentra bajo el id único y actualiza los datos del usuario.

➔ *Encontrar Usuario por Registro Académico “findUserByRegistroAcademico”:*

Encuentra un usuario por el registro académico, retorna el usuario si lo encuentra.

➔ *Agregar Cursos “addCoursesToUser”:*

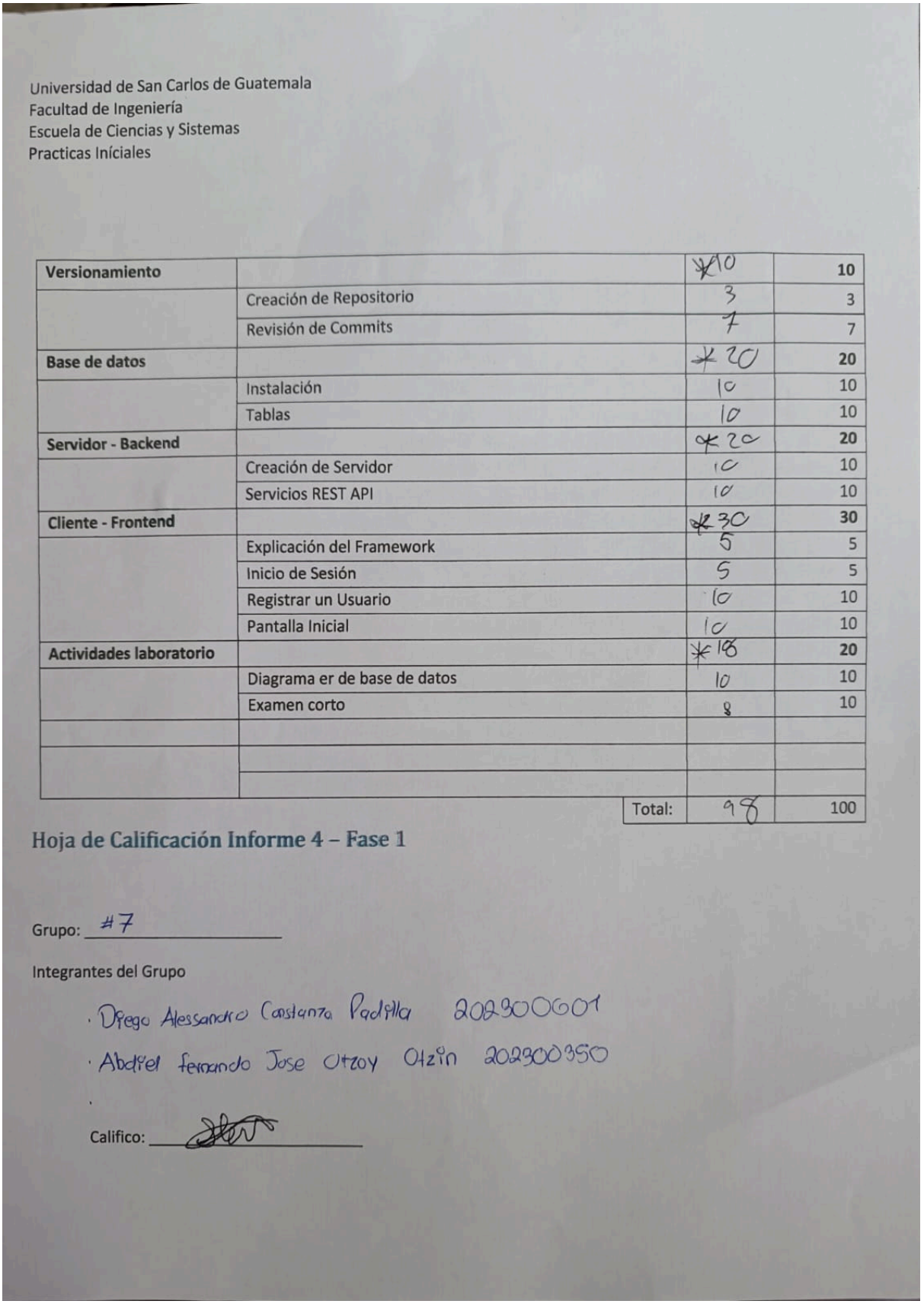
Agrega los cursos enviados al usuario que en ese momento haya iniciado sesión, si lo encuentra guarda los usuarios y lo envía a la base de datos.

➔ *Obtener los Cursos Aprobados “getCoursesApproved”:*

Obtiene todos los cursos que contengan el usuario que esté logueado en ese momento y retorna solo el listado de cursos.

○ Hoja de calificación

La hoja de calificación de la revisión de la Fase 1.



Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Practicas Inícíales

Versionamiento		*10	**10**
	Creación de Repositorio	3	3
	Revisión de Commits	7	7
Base de datos		*20	**20**
	Instalación	10	10
	Tablas	10	10
Servidor - Backend		*20	**20**
	Creación de Servidor	10	10
	Servicios REST API	10	10
Cliente - Frontend		*30	**30**
	Explicación del Framework	5	5
	Inicio de Sesión	5	5
	Registrar un Usuario	10	10
	Pantalla Inicial	10	10
Actividades laboratorio		*18	**20**
	Diagrama er de base de datos	10	10
	Examen corto	8	10
	Total:	98	100

Hoja de Calificación Informe 4 – Fase 1

Grupo: #7

Integrantes del Grupo

· Diego Alessandro Constanza Padilla 202300601
· Abdiel fernando Jose Otzoy Otzin 202300350
·

Califico: _____