Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación
Ingeniería en Tecnologías de la Información
Verano 2025

Modelos de desarrollo web

Actividad 0: Levantamiento de requerimientos y análisis funcional

Docente:
Alfredo García Juárez

Alumno:
Antonio Morales Badillo

Aplicación

Mi Agenda es un una aplicación que permite a las personas crear, leer, actualizar y eliminar actividades de su agenda virtual. La aplicación muestra las actividades del día en la página principal y cuenta con un apartado donde se muestran las actividades: totales, activas, finalizadas, realizada, parcialmente realizada y no realizada. también la actividad en curso y porcentajes de las actividades. El usuario puede registrarse, iniciar sesión, cerrar sesión y asignar el estado de las actividades.

Planteamiento funcional del API

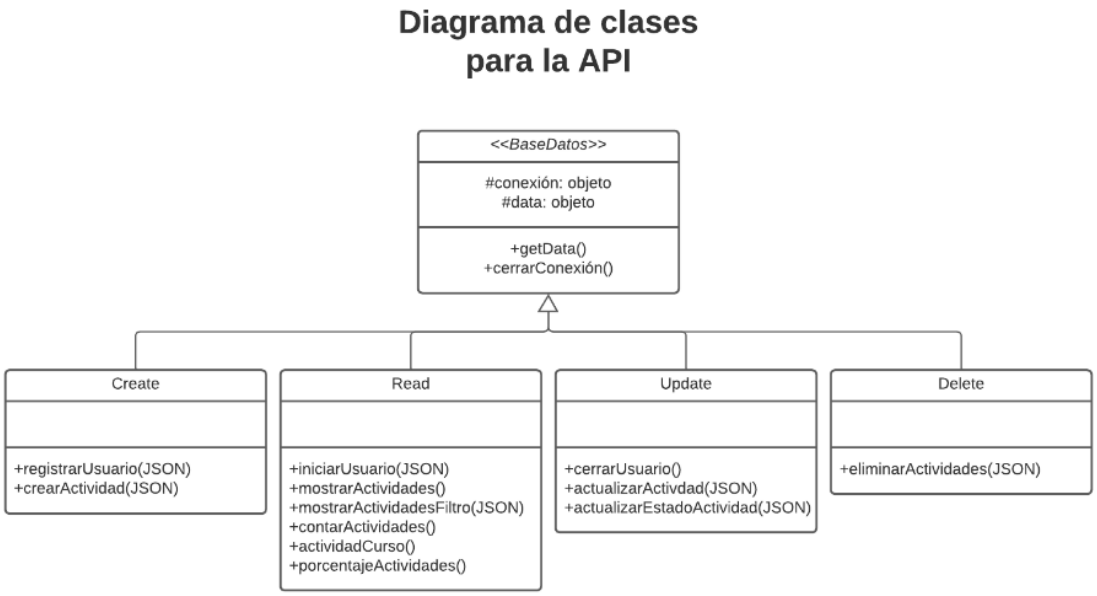
Nombre del Api: MiAgendaAPI

Propósito: proporcionar una comunicación sencilla y estructurada entre frontend y el backend, permitiendo realizar operaciones CRUD sobre las actividades y gestionar la autenticación de los usuarios.

La **API** será **interna**, ya que está diseñada para servir como intermediario entre el frontend y el backend. Su **arquitectura** que se basa en una **estructura orientada a objetos** que utiliza una clase abstracta principal encargada de gestionar la conexión con la base de datos, desde la cual heredan cuatro clases hijo que permiten hacer las operaciones fundamentales: **crear, leer, modificar y eliminar**. Esta separación permitirá tener un código modular, fácil de mantener y escalable, ya que cada clase se especializa en una operación especifica, permitiendo incorporaciones rápidas de nuevas funciones sin afectar a las existentes.

La arquitectura seguirá un estilo **REST**, lo que significa que se basa en el uso de métodos estándar **HTML**: **GET, POST, PUT y DELETE**. Para interactuar con los recursos. Cada una de sus funcionalidades está expuesta como un endpoint bien definido, lo que permite a otros componentes de la aplicación acceder a los datos de forma clara, predecible y estructurada.

Las operaciones de solicitudes y respuestas del API se serán en formato **JSON** por ser fácil de utilizar, esta estandarización para las entradas y salidas de los datos favorecerá la integración con las interfaces del frontend. Los datos son manipulados mediante clases hijas que encapsulan toda la lógica necesaria.



Requerimientos para implementar en el API

Requerimientos funcionales

1. Registrar nuevos usuarios en la aplicación
2. Iniciar sesión
3. Cerrar cesión
4. Crear actividad
5. Editar una actividad
6. Eliminar actividades
7. Actualizar estado de la actividad
8. Mostrar las actividades del usuario
9. Mostrar actividades filtradas por estado: activas, finalizadas, completadas, parcialmente completadas, no completadas
10. Mostrar el total de actividades
11. Mostrar la actividad en curso
12. Calcular porcentajes de actividades por estado

Requerimientos no funcionales

1. La API debe estar disponible en todo momento
2. Las respuestas deben estar en formato JSON
3. Debe ser escalable para permitir futuros módulos
4. La documentación debe ser clara y fácil de entender
5. Las consultas y respuestas deben ser rápidas
6. Debe ser fácil de usar
7. Debe implementar manejo de errores

Análisis funcional del API

Al ingresar al endpoint se iniciará un conexión con la base de datos y al finalizar el proceso de cada función cerrará. Cada endpoint **se conecta a una clase específica** según el tipo de operación (Crear, Leer, Actualizar, Eliminar).

Método	Endpoint	Descripción	Argumentos
Crear			
POST	/api/registrarUsuario	Registrar un nuevo usuario.	JSON: {nickname: ‘Juan1’, contraseña: ‘*********’}
POST	/api/crearActividad	Crear una nueva actividad y guardarla en la base de datos.	JSON: {Titulo: ‘actividad 1’, descripción: ‘Clase1’, fecha: ’11:00:54 08/06/2024’}
Leer			

GET	/api/iniciarUsuario	Iniciar sesión.	JSON: {nickname: ‘Juan 1’, contraseña: ‘******’}
GET	/api/mostrarActividades	Obtener todas las actividades del usuario.	INT: id
GET	/api/mostrarActividadesFiltro	Obtener actividades filtradas por estado: activa, realizada, parcialmente realizada, no realizada.	JSON: {Id: 1, filtro: ’realizado’}
GET	/api/contarActividades	Contar las actividades (totales, activas, finalizadas, etc.).	INT: id
GET	/api/actividadCurso	Obtener la actividad que está en curso.	INT: id
GET	/api/porcentajeActividades	Obtener el porcentaje de avance de las actividades.	INT: id
Actualizar			
PUT	/api/cerrarUsuario	Cerrar sesión	INT: id
PUT	/api/actualizarActivdad	Actualizar una actividad completa.	JSON: {Id: 1, Titulo: ‘actividad 1’, descripción: ‘Clase2’, fecha: ’11:00:54 08/06/2024’}

PUT	/api/actualizarEstadoActividad	Actualizar solo el estado de una actividad.	JSON: {Id:1, Titulo: ‘actividad 1’, descripción: ‘Clase1’, fecha: ’11:00:54 08/06/2024’, Estado: ‘Realizado’}
Eliminar			
DELETE	/api/eliminarActividades	Eliminar una actividad por su ID.	JSON: {Id: 1, Titulo: ‘actividad 1’, descripción: ‘Clase1’, fecha: ’11:00:54 08/06/2024’, Estado: ‘Realizado’}

Las respuestas que entregara el api serán simples:

{

Estado: 1, // (Puede ser error o exito)

Información: ‘’, // (Se detalla el error o el exito)

Datos: ‘datos’

}