Prueba Copayment: Documentación

Índice

Models

Bitacora_empleado

Este modelo representa la tabla “bitacora_empleados” de la base de datos copayment en mysql. Contiene los mismos campos y dos constructores.

Departamento

Este modelo representa la tabla “departamentos” de la base de datos copayment en mysql. Contiene los mismos campos y dos constructores.

Empleado

Este modelo representa la tabla “empleados” de la base de datos copayment en mysql. Contiene los mismos campos y dos constructores.

Usuario

Este modelo representa la tabla “usuarios” de la base de datos copayment en mysql. Contiene los mismos campos y dos constructores.

Dtos

Bitacora_empleadoDTO

Este DTO tiene como propiedades los parámetros necesarios de las peticiones para registrar la entrada y salida de los empleados en la bitácora, así como las peticiones para obtener información de las bitácoras y el cálculo del salario diario.

CalculoSalario

Este DTO tiene como propiedades los campos solicitados en los sp de mysql “calcularSalarioDiarioIndividual” y “calcularTodoSalarioDiario”. Así mismo, contiene dos constructores.

Login

Este DTO tiene como propiedades lo necesario para poder iniciar sesión. Tiene dos constructores

Utils

-Cadena de conexión (cadenaConexion)

Este método estático retorna la cadena de conexión hacía la base de datos de mysql de manera local.

Nota: al ejecutar en otra PC, favor de cambiar a la cadena de conexión por la suya.

-Convertir contraseña a Hash (passwordHash)

Este método estático recibe como parámetro un string y lo devuelve en forma hash.

-Validación de contraseñas (validarPassword)

Este método estático recibe como parámetros dos strings, uno en forma hash y otro en forma plana, y si la contraseña plana es correcta, retornará un true y en caso de no ser válida, retornará false.

DAL (Acceso a datos)

Bitacora_empleadoDAL

-Obtener Bitacora empleados: obtenerBitacoraEmpleado

Este método permite obtener los registros de las bitácoras realizadas por un empleado solamente. Recibe de parámetro un objeto tipo Bitacora_empleadoDTO que debe contener **el Id del empleado** a buscar. Retorna una lista tipo Bitacora_empleado.

Nota: En caso de que la bitácora no cuente con fecha de salida, se asigna el valor mínimo de fecha aceptable por mysql, la cual es 1001-01-01, lo que significa que dicho registro no contiene fecha de salida.

-Obtener Bitacora empleados: obtenerBitacorasEmpleados

Este método permite obtener todos los registros de las bitácoras realizadas **según la fecha que se envíe como parámetro dentro del objeto tipo Bitacora_empleadoDTO**. Retorna una lista tipo Bitacora_empleado.

Nota: En caso de que la bitácora no cuente con fecha de salida, se asigna el valor mínimo de fecha aceptable por mysql, la cual es 1001-01-01, lo que significa que dicho registro no contiene fecha de salida.

-Registrar Entrada/Salida: registrarEntrada

Este método permite crear un registro con la fecha de entrada del empleado en la tabla bitácora_empleados en mysql. Recibe como parámetro un objeto tipo **Bitacora_empleadoDTO** que **debe contener el Id del empleado y la fecha de entrada.** Retorna un string de confirmación.

-Registrar Entrada/Salida: registrarSalida

Este método permite actualizar un registro en la tabla bitácora_empleados en mysql con la fecha de salida del empleado. Recibe como parámetro un objeto tipo **Bitacora_empleadoDTO** que **debe contener el Id del empleado, la fecha de entrada y la fecha de salida.** Retorna un string de confirmación.

-Calcular salario diario: calcularSalarioDiarioIndividual

Este método permite obtener el cálculo del salario diario de un empleado según la fecha que se pase dentro del parámetro tipo **Bitacora_empleadoDTO** en la **propiedad Fecha**.

Así mismo, dicho parámetro debe contener el Id del empleado a calcular. Retorna una lista tipo CalculoSalario.

-Calcular salario diario: calcularTodoSalarioDiario

Este método permite obtener el cálculo del salario diario de todos los empleados según la fecha que se pase dentro del parámetro tipo **Bitacora_empleadoDTO** en la **propiedad Fecha**. Retorna una lista tipo CalculoSalario.

Nota: En ambos métodos (calcularSalarioDiarioIndividual y calcularTodoSalarioDiario), retornara como valor 0.00 en Salario_diario y 0.00 en Horas_trabajadas en caso de que el registro no tenga fecha de salida registrada.

DepartamentoDAL

-Obtener Departamento: obtenerDepartamento

Este método permite obtener un registro de la tabla departamentos en mysql. Recibe como parámetro el **id del departamento a buscar**. Retorna un objeto tipo Departamento.

-Obtener Departamento: obtenerDepartamentos

Este método permite obtener todos los registros disponibles de la tabla departamentos en mysql. **No recibe parámetros**. Retorna una lista tipo Departamento.

-Crear Departamento: crearDepartamento

Este método permite crear un registro en la tabla departamentos en mysql. Recibe como parámetro **un objeto tipo Departamento**. Retorna un string de confirmación.

-Actualizar Departamento: actualizarDepartamento

Este método permite actualizar completamente un registro en la tabla departamentos en mysql. Recibe como parámetro **un objeto tipo Departamento** que **debe contener las tres propiedades del modelo**. Retorna un string de confirmación.

-Actualizar Departamento: actualizarPagoHora

Este método permite actualizar la columna Pago_hora de un registro en la tabla departamentos en mysql. Recibe como parámetro un objeto tipo Departamento que **debe contener la propiedad Id_departamento y Pago_hora**. Retorna un string de confirmación.

-Eliminar Departamento: eliminarDepartamento

Este método permite eliminar un registro en la tabla departamentos en mysql. Recibe como parámetro el **id del departamento a eliminar**. Retorna un string de confirmación.

EmpleadoDAL

-Obtener empleado: obtenerEmpleado

Este método permite obtener un registro de tabla empleados en mysql. Recibe como parámetro el **Id del empleado a buscar**. Retorna un objeto tipo Empleado.

-Obtener empleado: obtenerEmpleados

Este método permite obtener todos los registros de la tabla empleados en mysql. **No recibe parámetros**. Retorna una lista tipo Empleados.

-Crear empleado: crearEmpleado

Este método permite crear un registro en la tabla empleados en mysql. Recibe como parámetro **un objeto tipo Empleado**. Retorna un string de confirmación.

-Actualizar empleado: actualizarEmpleado

Este método permite actualizar un registro en la tabla empleados en mysql. Recibe como parámetro **un objeto tipo Empleado**. Retorna un string de confirmación.

-Eliminar empleado: eliminarEmpleado

Este método permite eliminar un registro en la tabla empleados en mysql. Recibe como parámetro el **Id del empleado a eliminar**. Retorna un string de confirmación.

UsuarioDAL

-Crear usuario: crearUsuario

Este método permite crear un registro en la tabla usuarios en mysql. Recibe como parámetro **un objeto tipo Usuario**. En el proceso, la contraseña colocada en el parámetro será transformada a hash para guardarse así en la base de datos; esto lo hace mediante el método “passwordHash” de la clase “Utils”. Retorna un string de confirmación.

-Obtener usuario: obtenerUsuario

Este método permite obtener un registro de la tabla usuarios en mysql. Recibe como parámetro **el nombre del usuario.** Retorna un objeto tipo Usuario.

-Eliminar usuario: eliminarUsuario

Este método permite eliminar un registro de la tabla usuarios en mysql. Recibe como parámetro **el nombre del usuario.** Retorna un string de confirmación.

Controllers

BitacoraEmpleadoController

-Obtener Bitacora empleados: obtenerBitacoraEmpleado

Utiliza el método obtenerBitacoraEmpleado de Bitacora_empleadoDAL. **Debe** pasarse el Id del empleado a buscar en la propiedad Id_empleado del objeto Bitacora_empleadoDTO. Retorna una lista tipo Bitacora_empleado.

-Obtener Bitacora empleados: obtenerBitacorasEmpleados

Utiliza el método obtenerBitacorasEmpleados de Bitacora_empleadoDAL. **Debe** pasarse la fecha a buscar en la propiedad Fecha del objeto Bitacora_empleadoDTO. Retorna una lista tipo Bitacora_empleado.

Nota: En caso de que no se asigne el parámetro Fecha del objeto Bitacora_empleadoDTO, se asignará el día actual a la hora 00:00:00.

-Registrar Entrada/Salida: registrarEntrada

Utiliza el método registrarEntrada de Bitacora_empleadoDAL. **Debe** pasarse el Id del empleado en la propiedad Id_empleado del objeto Bitacora_empleadoDTO. Retorna un string de confirmación.

Nota: En esta petición solo es necesario asignar el Id del empleado al cual se le registrará su hora de entrada, ya que la fecha de entrada se genera automáticamente al hacer la partición; la fecha generada es la actual según el método DateTime.Now.

Registrar Entrada/Salida: registrarSalida

Utiliza el método registrarSalida de Bitacora_empleadoDAL. **Debe** pasarse el Id del empleado en la propiedad Id_empleado del objeto Bitacora_empleadoDTO. Retorna un string de confirmación.

Nota: En esta petición solo es necesario asignar el Id del empleado al cual se le registrará su hora de salida, ya que la fecha de salida y de entrada se genera automáticamente al hacer la partición; la fecha de entrada se genera con el método DateTime.Today y la fecha de salida se genera con el método DateTime.Now.

Calcular salario diario: calcularSalarioDiarioIndividual

Utiliza el método calcularSalarioDiarioIndividual de Bitacora_empleadoDAL. **Debe** pasarse el Id del empleado en la propiedad Id_empleado y opcionalmente la fecha a calcular en la propiedad Fecha del objeto Bitacora_empleadoDTO. Retorna una lista tipo CalculoSalario.

Nota: En caso de que no se asigne el parámetro Fecha del objeto Bitacora_empleadoDTO, se asignará el día actual a la hora 00:00:00.

Calcular salario diario: calcularTodoSalarioDiario

Utiliza el método calcularTodoSalarioDiario de Bitacora_empleadoDAL. **Debe** pasarse la fecha a calcular en la propiedad Fecha del objeto Bitacora_empleadoDTO. Retorna una lista tipo CalculoSalario.

Nota: En caso de que no se asigne el parámetro Fecha del objeto Bitacora_empleadoDTO, se asignará el día actual a la hora 00:00:00.

DepartamentoController

-Obtener Departamento: obtenerDepartamento

Utiliza el método obtenerDepartamento de DepartamentoDAL. **Debe** pasarse directamente el Id del registro. Retorna un objeto tipo Departamento.

-Obtener Departamento: obtenerDepartamentos

Utiliza el método obtenerDepartamentos de DepartamentoDAL. **No recibe parámetros**. Retorna una lista tipo Departamento.

-Crear Departamento: crearDepartamento

Utiliza el método crearDepartamento de DepartamentoDAL. **Debe** pasarse un objeto tipo Departamento con todos los campos necesarios. Retorna un string de confirmación.

-Actualizar Departamento: actualizarDepartamento

Utiliza el método actualizarDepartamento de DepartamentoDAL. **Debe** pasarse un objeto tipo Departamento con todos los campos necesarios. Retorna un string de confirmación.

-Actualizar Departamento: actualizarPagoHora

Utiliza el método actualizarPagoHora de DepartamentoDAL. **Debe** pasarse un objeto tipo Departamento con los campos Id_departamento y Pago_hora. Retorna un string de confirmación.

-Eliminar Departamento: eliminarDepartamento

Utiliza el método eliminarDepartamento de DepartamentoDAL. **Debe** pasarse directamente el Id del registro. Retorna un string de confirmación.

EmpleadoController

-Obtener empleado: obtenerEmpleado

Utiliza el método obtenerEmpleado de EmpleadoDAL. **Debe** pasarse directamente el Id del registro. Retorna un objeto tipo Empleado.

-Obtener empleado: obtenerEmpleados

Utiliza el método obtenerEmpleados de EmpleadoDAL. **No recibe parámetros**. Retorna una lista tipo Empleados.

-Crear empleado: crearEmpleado

Utiliza el método crearEmpleado de EmpleadoDAL. **Debe** pasarse un objeto tipo Empleado con todos los campos necesarios, opcionalmente: Correo_contacto, Telefono_contacto y Activo (en el sp se asgina automáticamente en “S” de Sí). Retorna un string de confirmación.

-Actualizar empleado: actualizarEmpleado

Utiliza el método actualizarEmpleado de EmpleadoDAL. **Debe** pasarse un objeto tipo Empleado con todos los campos necesarios, opcionalmente: Correo_contacto y Telefono_contacto. Retorna un string de confirmación.

-Eliminar empleado: eliminarEmpleado

Utiliza el método eliminarEmpleado de EmpleadoDAL. **Debe** pasarse directamente el Id del registro. Retorna un string de confirmación.

UsuarioController

-Crear usuario: crearUsuario

Utiliza el método crearUsuario de UsuarioDAL. **Debe** pasarse un objeto tipo Usuario con todos los campos necesarios. Retorna un string de confirmación.

-Iniciar sesión: iniciarSesion

Utiliza el método obtenerUsuario de UsuarioDAL y compara el valor de las contraseñas, la introducida en la petición y la que se encuentra en la tabla usuarios, mediante el uso el método “validarPassword” de la clase Utils. Si la contraseña es correcta, genera un token jwt mediante el método “generarToken” de la clase Utils, en caso contrario, retorna un “401 Unauthorized”. **Debe** pasarse un objeto tipo Login con los campos necesarios**.** Retorna un token jwt o un “401 Unauthorized”.

-Eliminar usuario: eliminarUsuario

Utiliza el método eliminarUsuario de UsuarioDAL. **Debe** pasarse directamente el Id del registro. Retorna un string de confirmación.

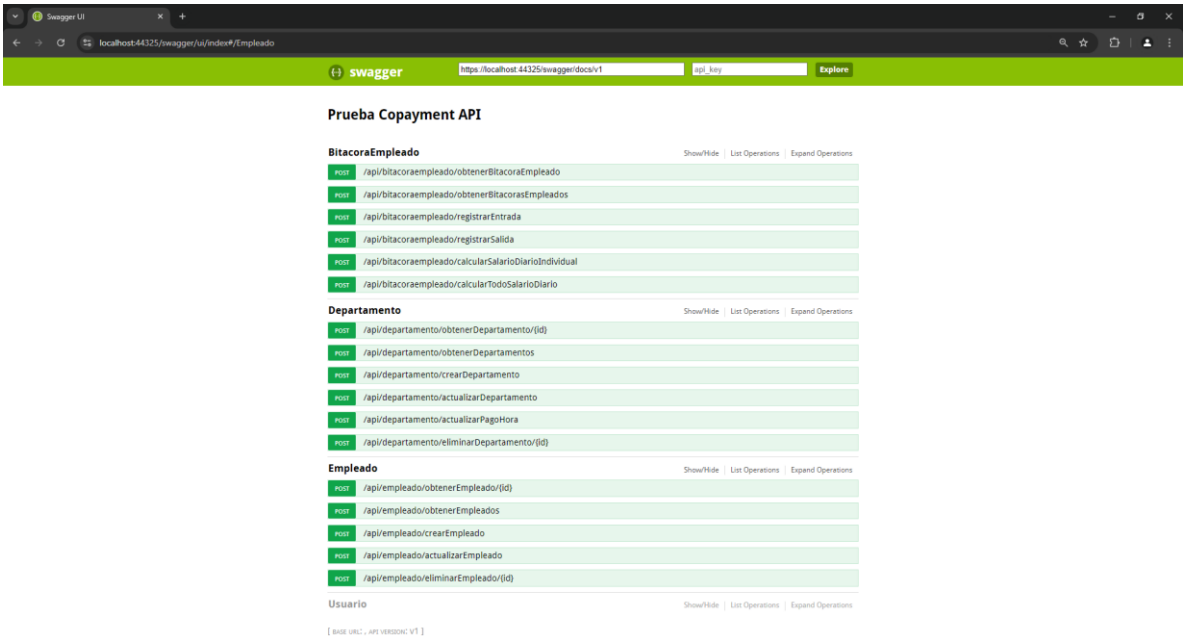
Consideraciones

Cadena de conexión

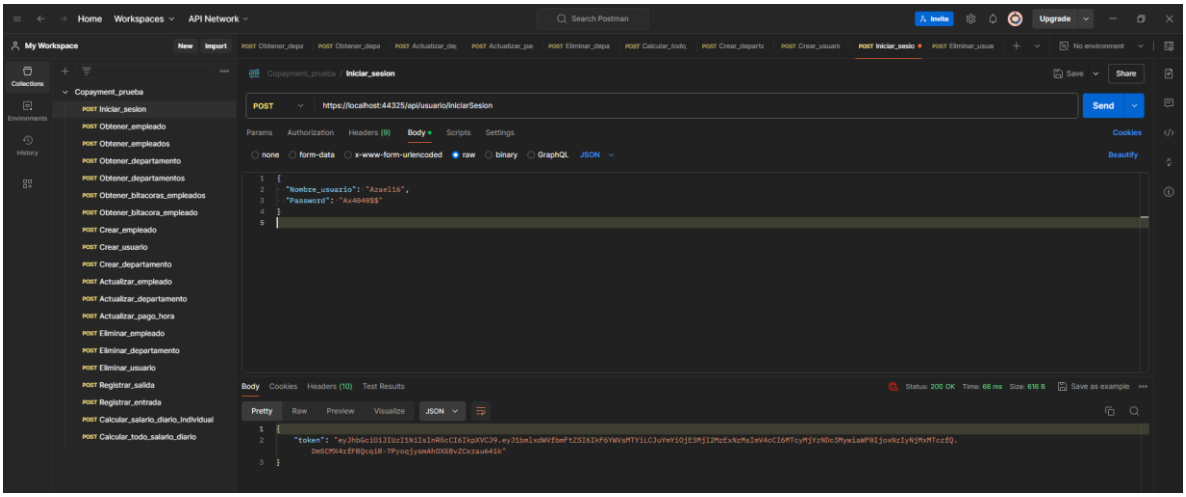
Al ejecutar la API en otra PC, deberá cambiar la cadena de conexión que se encuentra en el método **cadenaConexion** de la clase **Utils**.

¿Cómo probar la API?

La API puede ser probada directamente al ejecutar el servicio, ya que cuenta con Swagger:

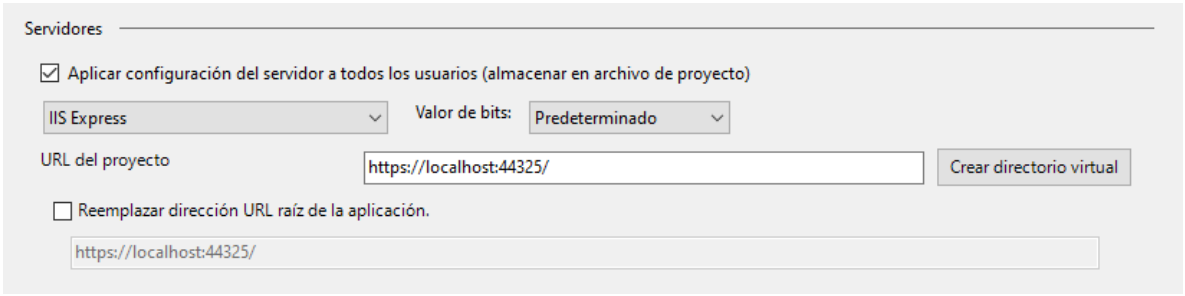


O también, puede ser probada mediante Postman:



Puerto de ejecución

Los Endpoints preparados en el JSON solicitado, están a través del **puerto 44325:**



En caso de que al momento de su ejecución se haya cambiado dicho puerto o tenga algún conflicto, favor de cambiar ese valor en los Endpoints proporcionados en el JSON para poder ejecutarlos en Postman.