**FORMATO PARA DECLARACIÓN DE ESTRUCTURAS DE DATOS**

|  |  |
| --- | --- |
| **Nombre del Proyecto:** | Hotel costa azul |
| **Nombre de los Integrantes:** | **Juan esteban Paternina torres, Julián barcas negras, leonardo Álvarez Pantoja, Bryant Díaz luna** |
| **Fecha:** | Fecha: **22/10/2024** |

**ESTRUCTURAS DE DATOS UTILIZADAS:**

1. **Nombre de la Estructura de Datos:** lista de usuarios

* **Descripción:** Servicio utilizado para gestionar la autenticación de usuarios y su registro en el sistema. Implementa la lógica para guardar usuarios, autenticar credenciales, y listar usuarios desde una base de datos.
* **Tipo de Estructura:** lista(array list)
* **Implementación en el Código:**

package com.hotelCostaAzul.HotelCostaAzul.SignUpandLogin.Servicio;

import com.hotelCostaAzul.HotelCostaAzul.SignUpandLogin.Modelo.Rol;

import com.hotelCostaAzul.HotelCostaAzul.SignUpandLogin.Modelo.Usuario;

import com.hotelCostaAzul.HotelCostaAzul.SignUpandLogin.Repositorio.UsuarioRepositorio;

import com.hotelCostaAzul.HotelCostaAzul.SignUpandLogin.dto.UsuarioRegitroDTO;

import org.springframework.stereotype.Service;

import java.util.List;

@Service

public class UsuarioServicioImp implements UsuarioServicio {

private final UsuarioRepositorio usuarioRepositorio;

public UsuarioServicioImp(UsuarioRepositorio usuarioRepositorio) {

this.usuarioRepositorio = usuarioRepositorio;

}

@Override

public Usuario guardar(UsuarioRegitroDTO registroDTO) {

// Crear una nueva instancia de Usuario usando los datos del DTO

Usuario usuario = new Usuario(

registroDTO.getNombre(),

registroDTO.getEmail(),

registroDTO.getPassword(),

List.of(new Rol("ROLE\_USER")) // Asignar rol predeterminado

);

// Guardar el usuario en la base de datos y devolverlo

return usuarioRepositorio.save(usuario);

}

@Override

public boolean autenticar(String email, String password) {

// Buscar el usuario por email

Usuario usuario = usuarioRepositorio.findByEmail(email);

// Verificar si el usuario existe y si la contraseña coincide

if (usuario != null && usuario.getPassword().equals(password)) {

return true; // Autenticación exitosa

}

return false; // Autenticación fallida

}

@Override

public List<Usuario> listarUsuarios() {

// Obtener y devolver la lista de todos los usuarios

return usuarioRepositorio.findAll();

}

}

1. **Nombre de la Estructura de Datos:** Nombre de la Estructura

* **Descripción:** Servicio encargado de gestionar la creación y almacenamiento de reservas de alojamiento, así como el cálculo de costos según la duración y tipo de habitación
* **Tipo de Estructura:** lista(array list)
* **Implementación en el Código:**

package com.hotelCostaAzul.HotelCostaAzul.SignUpandLogin.Servicio;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.jdbc.core.JdbcTemplate;

import org.springframework.stereotype.Service;

import com.hotelCostaAzul.HotelCostaAzul.SignUpandLogin.Repositorio.AlojamientoRepository;

@Service

public class AlojamientoServicio {

    private final AlojamientoRepository alojamientoRepository;

    private final JdbcTemplate jdbcTemplate;

    @Autowired

    public AlojamientoServicio(AlojamientoRepository alojamientoRepository, JdbcTemplate jdbcTemplate) {

        this.alojamientoRepository = alojamientoRepository;

        this.jdbcTemplate = jdbcTemplate;

    }

    public void actualizarCamposPorTipoHabitacion(String tipoHabitacion, int dias) {

        // Modificar la consulta para actualizar solo un registro (LIMIT 1)

        String sqlUpdate = "UPDATE alojamiento SET n\_Dias = ?, costo = ? WHERE non\_Habitacion = ? AND (n\_Dias IS NULL OR n\_Dias = 0 OR costo IS NULL OR costo = 0) LIMIT 1";

        // Calcular el costo

        double costo = calcularCosto(dias, tipoHabitacion);

        // Ejecutar la actualización

        jdbcTemplate.update(sqlUpdate, dias, costo, tipoHabitacion);

        //verificar

        String sqlCheck = "SELECT COUNT(\*) FROM alojamiento WHERE non\_Habitacion = ? AND (n\_Dias IS NULL OR n\_Dias = 0 OR costo IS NULL OR costo = 0)";

    Integer habitacionesDisponibles = jdbcTemplate.queryForObject(sqlCheck, Integer.class, tipoHabitacion);

    if (habitacionesDisponibles == null || habitacionesDisponibles == 0) {

        System.out.println("No hay más habitaciones disponibles para el tipo: " + tipoHabitacion);

        return; // Termina el método si no hay habitaciones disponibles

    }

    }

    private double calcularCosto(int nDias, String tipoHabitacion) {

        switch (tipoHabitacion) {

            case "Suite":

                return nDias \* 500000; // Costo para Suite

            case "Deluxe":

                return nDias \* 400000; // Costo para Deluxe

            default:

                return 0;

        }

    }

    public int obtenerHabitacionesDisponibles(String tipoHabitacion) {

        String sqlCheck = "SELECT COUNT(\*) FROM alojamiento WHERE non\_Habitacion = ? AND (n\_Dias IS NULL OR n\_Dias = 0 OR costo IS NULL OR costo = 0)";

        Integer habitacionesDisponibles = jdbcTemplate.queryForObject(sqlCheck, Integer.class, tipoHabitacion);

        return habitacionesDisponibles != null ? habitacionesDisponibles : 0;

    }

}

3**. Nombre de la Estructura de Datos:** Nombre de la Estructura

* **Descripción:** Interfaz que define las operaciones para gestionar usuarios en el sistema, incluyendo la creación de nuevos usuarios, autenticación y listado de usuarios
* **Tipo de Estructura:** lista(array list)
* **Implementación en el Código:** Breve descripción de cómo se implementa la estructura en el código fuente

// Importaciones necesarias para la interfaz

import com.hotelCostaAzul.HotelCostaAzul.SignUpandLogin.Modelo.Usuario;

import com.hotelCostaAzul.HotelCostaAzul.SignUpandLogin.dto.EstadiaDTO;

import com.hotelCostaAzul.HotelCostaAzul.SignUpandLogin.dto.UsuarioRegitroDTO;

import java.util.List;

// Interfaz que define los métodos para la gestión de usuarios en el sistema

public interface UsuarioServicio {

// Método para guardar un nuevo usuario utilizando el DTO

public Usuario guardar(UsuarioRegitroDTO registroDTO);

// Método para autenticar a un usuario dado su email y contraseña

boolean autenticar(String email, String password);

// Método para listar todos los usuarios registrados en el sistema

List<Usuario> listarUsuarios();

}

**OBSERVACIONES ADICIONALES:**

**Eficiencia en la Gestión de Usuarios:** La interfaz UsuarioServicio y su implementación permiten una gestión eficiente de los usuarios, facilitando la autenticación y el registro sin complicar la lógica de negocio. El uso de un DTO (Data Transfer Object) para el registro ayuda a validar y encapsular datos de entrada.