



FACULTAD DE CIENCIAS

Sistema de reservas para una cadena hotelera

Proyecto Final | Estructuras de Datos

Valeria Cano Soto

Dr. Ulises Rodríguez Domínguez
Semestre 2025-2

Contenido

- 1 Descripción
- 2 Área de aplicación
- 3 Objetivo
- 4 Alcances
- 5 ¿Cómo se cumplieron los alcances?
- 6 Presentación funcional

Descripción

- Desarrollo de un **sistema de reservas** para una cadena hotelera con múltiples sucursales, orientado a la **gestión eficiente de usuarios, habitaciones y reservas**, haciendo uso de **árboles AVL** como estructura de datos principal.



Contenido

- 1 Descripción
- 2 Área de aplicación
- 3 Objetivo
- 4 Alcances
- 5 ¿Cómo se cumplieron los alcances?
- 6 Presentación funcional



Área de aplicación



- Bases de Datos
- Aplicaciones de software orientadas a servicios
- Sector hotelero y de hospitalidad en general

Contenido

- 1 Descripción
- 2 Área de aplicación
- 3 **Objetivo**
- 4 Alcances
- 5 ¿Cómo se cumplieron los alcances?
- 6 Presentación funcional

Objetivo

Implementar **árboles AVL** (y **listas enlazadas simples**, para el manejo de claves repetidas) como estructuras de datos principales en un sistema funcional de reservas hoteleras.

Se busca aprovechar la utilidad y eficiencia de estas estructuras en un contexto realista.

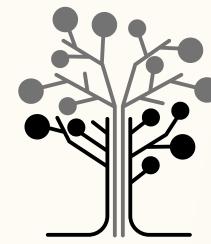
Secundarios

- Permitir búsquedas eficientes de hoteles y habitaciones por múltiples criterios.
- Evitar conflictos de *overbooking* al registrar nuevas reservas.
- Diseñar un sistema modular que permita una gestión ordenada de cada entidad.

Contenido

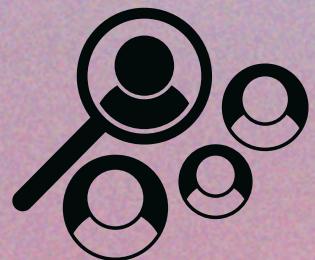
- 1 Descripción
- 2 Área de aplicación
- 3 Objetivo
- 4 **Alcances**
- 5 ¿Cómo se cumplieron los alcances?
- 6 Presentación funcional

Alcances



Implementación de árboles AVL

Para indexación eficiente y consultas de hoteles y habitaciones por nombre, ubicación, categoría, precio y fecha.



Implementación de funciones

Para:

- Registrar usuarios.
- Realizar, consultar o cancelar reservas.
- Incluir total facturado en un periodo.



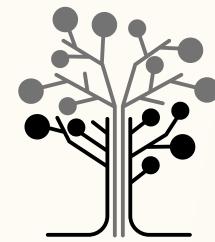
Interfaz gráfica

Amigable con el usuario.

Implementación de Tablas Hash

Para almacenar una gran cantidad de usuarios.

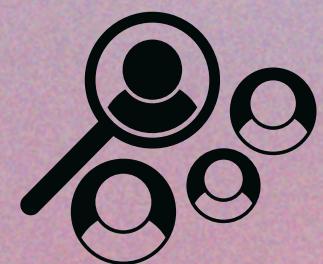
Alcances



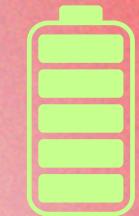
Implementación de árboles AVL



Para indexación eficiente y consultas de hoteles y habitaciones por nombre, ubicación, categoría, costo/precio y fecha.



Implementación de funciones



Para:

- Registrar usuarios.
- Realizar, consultar o cancelar reservas.
- Incluir total facturado en un periodo.

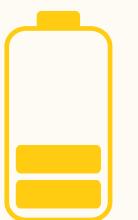


Interfaz gráfica

Amigable con el usuario.



Implementación de Tablas Hash



Para almacenar una gran cantidad de usuarios.

Contenido

- 1 Descripción
- 2 Área de aplicación
- 3 Objetivo
- 4 Alcances
- 5 **¿Cómo se cumplieron los alcances?**
- 6 Presentación funcional

¿Cómo lo
hice?

Entidades

Hotel



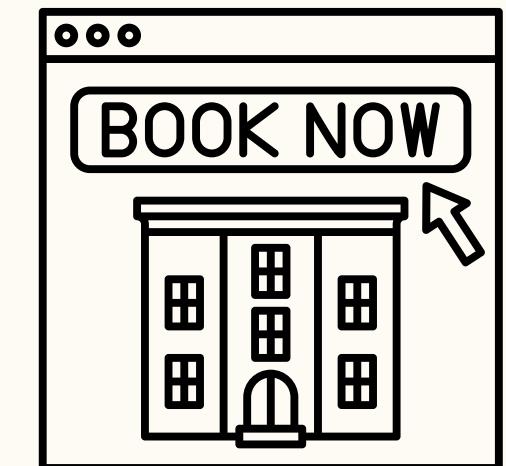
Usuario



Habitacion

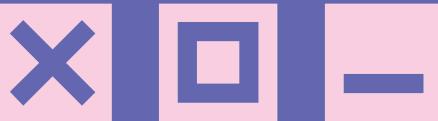


Reserva



¿Cómo lo hice?

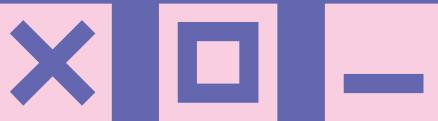
Hotel



```
public class Hotel {  
    private int id;                                // ID único del hotel  
    private String nombre;                          // Nombre del hotel  
    private String ciudad;                         // Ciudad donde se ubica el hotel  
    private String direccion;                      // Dirección completa: calle, número, colonia, etc.  
    private String telefono;                        // Número telefónico  
    private String email;                           // Correo electrónico  
    private int categoria;                         // Categoría de 1 a 5 estrellas  
  
    public Hotel(int id, String nombre, String ciudad, String direccion,  
                String telefono, String email, int categoria) {  
        this.id = id;  
        this.nombre = nombre;  
        this.ciudad = ciudad;  
        this.direccion = direccion;  
        this.telefono = telefono;  
        this.email = email;  
        this.categoria = categoria;  
    }  
}
```

¿Cómo lo hice?

Habitación



```
public class Habitacion {  
    private int id;                                // ID único de La habitación  
    private int idHotel;                            // ID del hotel al que pertenece  
    private int numero;                            // Número de La habitación  
    private String tipo;                           // Tipo de habitación: Individual, Doble, Suite, etc.  
    private int capacidad;                         // Capacidad máxima  
    private String amenidades;                     // Lista de amenidades separadas por coma  
    private double precioNoche;                   // Precio por noche  
    private String disponibilidad;                // Estado actual de disponibilidad  
    private LocalDate fechaDisponible;           // Fecha a partir de la cual está disponible  
  
    public Habitacion(int id, int idHotel, int numero, String tipo, int capacidad,  
                      String amenidades, double precioNoche, String disponibilidad,  
                      LocalDate fechaDisponibleDesde) {  
        this.id = id;  
        this.idHotel = idHotel;  
        this.numero = numero;  
        this.tipo = tipo;  
        this.capacidad = capacidad;  
        this.amenidades = amenidades;  
        this.precioNoche = precioNoche;
```

¿Cómo lo hice?

Usuario



```
X  □  -  
  
public class Usuario {  
    private int id;                                // ID único del cliente  
    private String nombre;  
    private String apellidoPaterno;  
    private String apellidoMaterno;  
    private String email;  
    private String telefono;  
    private String ciudad;  
    private String direccionCompleta; // Calle, número, colonia, etc.  
    private LocalDate fechaRegistro;  
  
    public Usuario(int id, String nombre, String apellidoPaterno, String apellidoMaterno,  
                  String email, String telefono, String ciudad, String direccionCompleta,  
                  LocalDate fechaRegistro) {  
        this.id = id;  
        this.nombre = nombre;  
        this.apellidoPaterno = apellidoPaterno;  
        this.apellidoMaterno = apellidoMaterno;  
        this.email = email;  
        this.telefono = telefono;  
        this.ciudad = ciudad;
```

¿Cómo lo hice?

Reserva

```
X  □  -  
public class Reserva {  
    private int idReserva;  
    private int idCliente;  
    private int idHabitacion;  
    private LocalDate fechaInicio;  
    private LocalDate fechaTermino;  
    private String estadoReserva;      // Cancelada, Confirmada, Finalizada.  
    private double importeTotal;  
  
    public Reserva(int idReserva, int idCliente, int idHabitacion,  
                  LocalDate fechaInicio, LocalDate fechaTermino,  
                  String estadoReserva, double importeTotal) {  
        this.idReserva = idReserva;  
        this.idCliente = idCliente;  
        this.idHabitacion = idHabitacion;  
        this.fechaInicio = fechaInicio;  
        this.fechaTermino = fechaTermino;  
        this.estadoReserva = estadoReserva;  
        this.importeTotal = importeTotal;  
    }  
}
```



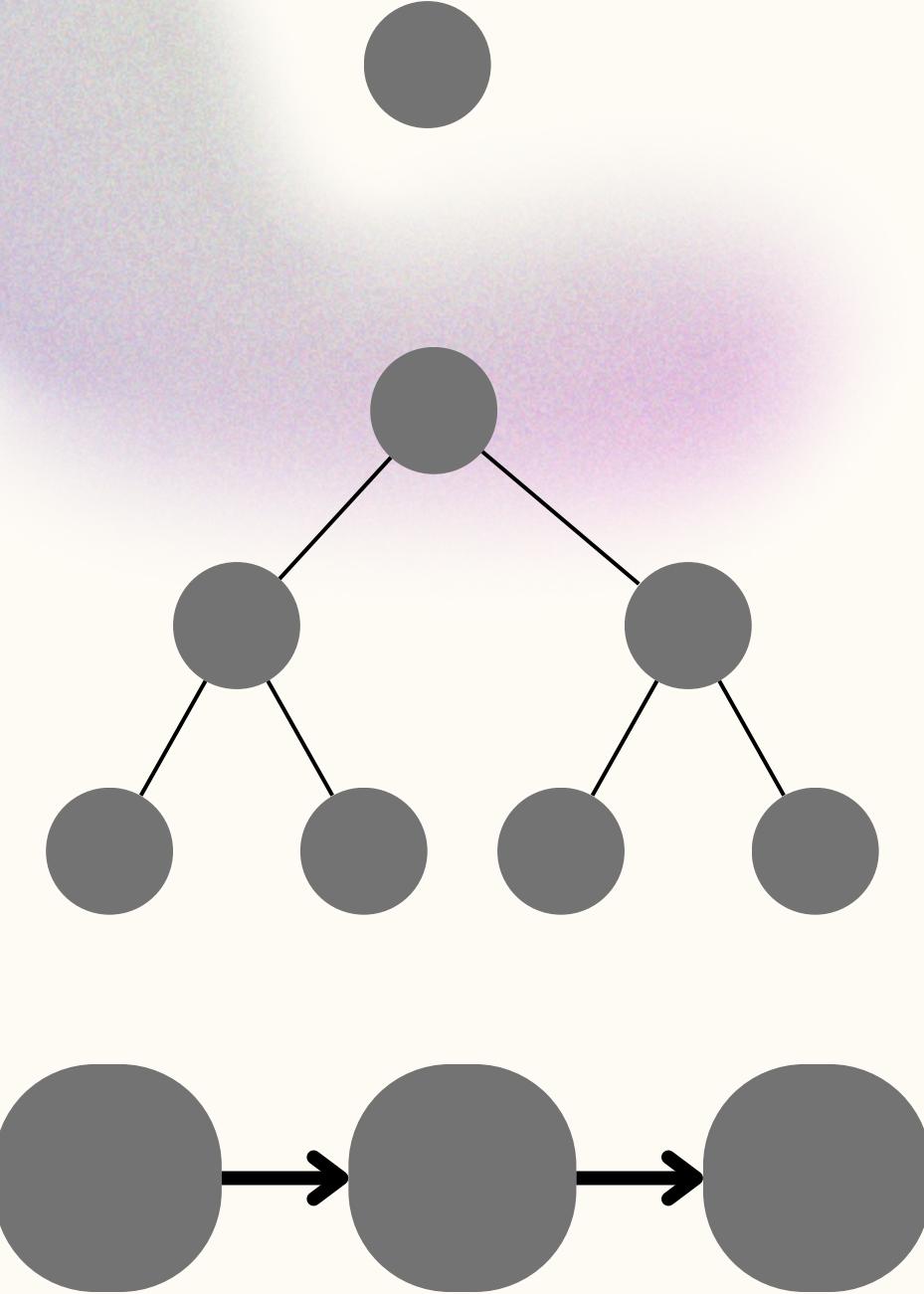
**¿Cómo lo
hice?**

EDD

Nodo

Árbol AVL

Lista Enlazada Simple
(para manejo de claves repetidas)



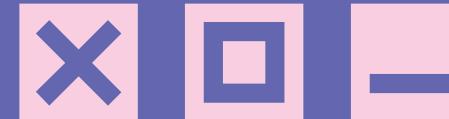
¿Cómo lo hice?

Nodo

```
X  □  -  
public class Nodo<T> {  
    T clave;          // Valor almacenado en el nodo  
    Nodo<T> padre;    // Padre  
    Nodo<T> hijoIzq;  // Hijo izquierdo  
    Nodo<T> hijoDer;  // Hijo derecho  
    ListaEnlazadaSimple<T> repetidos;  
  
    // Constructor para crear un nodo por clave  
    public Nodo(T clave) {  
        this.clave = clave;  
        this.hijoIzq = null;  
        this.hijoDer = null;  
        this.padre = null;  
        this.repetidos = new ListaEnlazadaSimple<>();  
    }  
}
```

¿Cómo lo hice?

Árbol AVL



```
public class ArbolAVL<T> implements TDABinaryTree<T> {
```

```
    private Nodo<T> raiz;
```

```
    private Comparator<T> comparator;
```

// Constructor sin comparador: usa Comparable

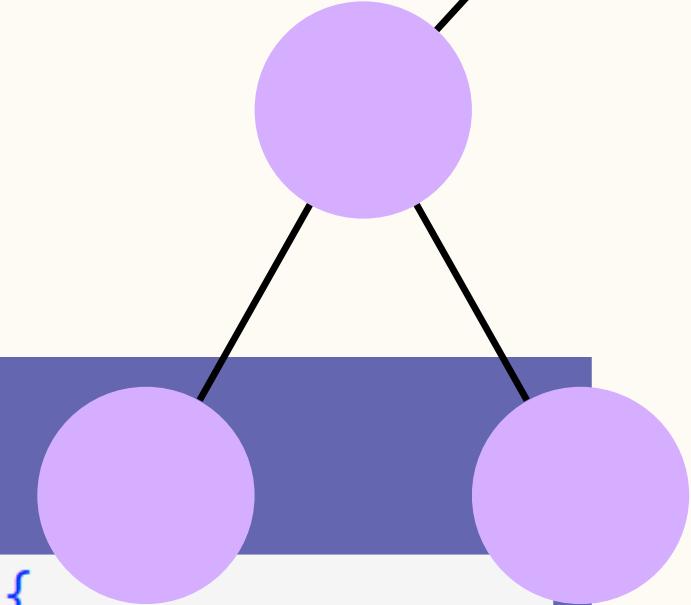
```
    public ArbolAVL() {
```

```
        this.comparator = null;
        this.raiz = null;
    }
```

// Constructor con comparador personalizado

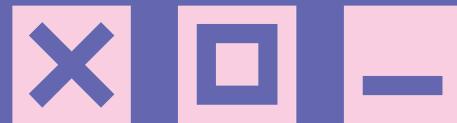
```
    public ArbolAVL(Comparator<T> comparator) {
```

```
        this.comparator = comparator;
        this.raiz = null;
    }
```

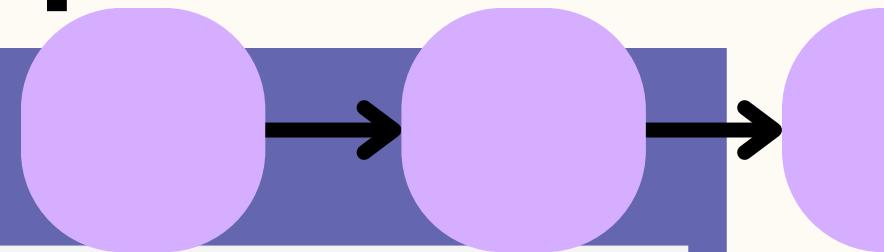


¿Cómo lo hice?

Lista Enlazada Simple



```
public class ListaEnlazadaSimple<T> implements TDALista<T> {  
    /**  
     * Clase interna que representa un nodo de La Lista.  
     */  
    private class Nodo { ...  
  
    /**  
     * Clase interna para la iteración sobre La Lista.  
     */  
    private class IteradorLista implements Iterator<T> { ...  
  
    private Nodo cabeza;  
    private Nodo cola;  
    private int tamaño;  
  
    public ListaEnlazadaSimple() {  
        this.cabeza = null;  
        this.cola = null;  
        this.tamaño = 0;  
    }  
}
```



¿Cómo lo
hice?

Comparadores

Hotel



Comparar dos hoteles por:

- ID
- Nombre
- Ubicación
- Categoría

Comparar dos habitaciones por:

- ID
- Precio por noche
- Fecha de disponibilidad

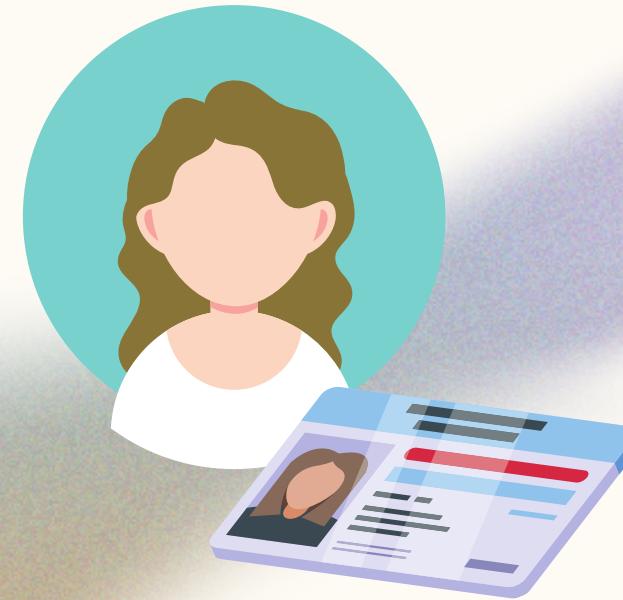
Habitacion



¿Cómo lo
hice?

Comparadores

Usuario



Comparar dos usuarios por:
• ID

Comparar dos reservas por:

- ID Reserva
- ID Cliente
- ID Habitación



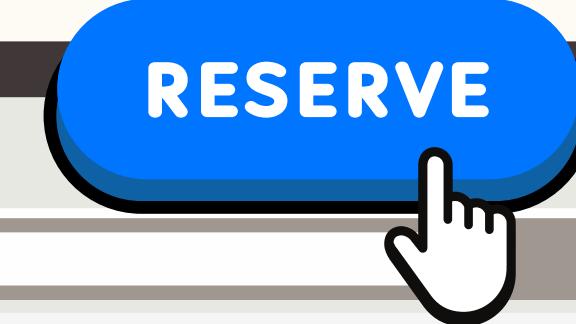
Reservas

¿Cómo lo hice?



```
/*
 * Comparador para ordenar hoteles por nombre.
 */
public class ComparadorNombre implements Comparator<Hotel> {
    @Override
    public int compare(Hotel h1, Hotel h2) {
        String n1 = normalizar(h1.getNombre());
        String n2 = normalizar(h2.getNombre());
        return n1.compareTo(n2);
    }

    private String normalizar(String s) {
        if (s == null) return "";
        s = Normalizer.normalize(s.toLowerCase(), Normalizer.Form.NFD);
        return s.replaceAll(regex:"\\p{InCombiningDiacriticalMarks}+", replacement:
    }
}
```



```
* Comparador para ordenar reservas por ID Usuario.
*/
public class ComparadorIDUsuarioR implements Comparator<Reserva> {
    @Override
    public int compare(Reserva r1, Reserva r2) {
        return Integer.compare(r1.getIdCliente(), r2.getIdCliente());
    }
}
```

¿Cómo lo
hice?

Gestores



Habitaciones



Hoteles

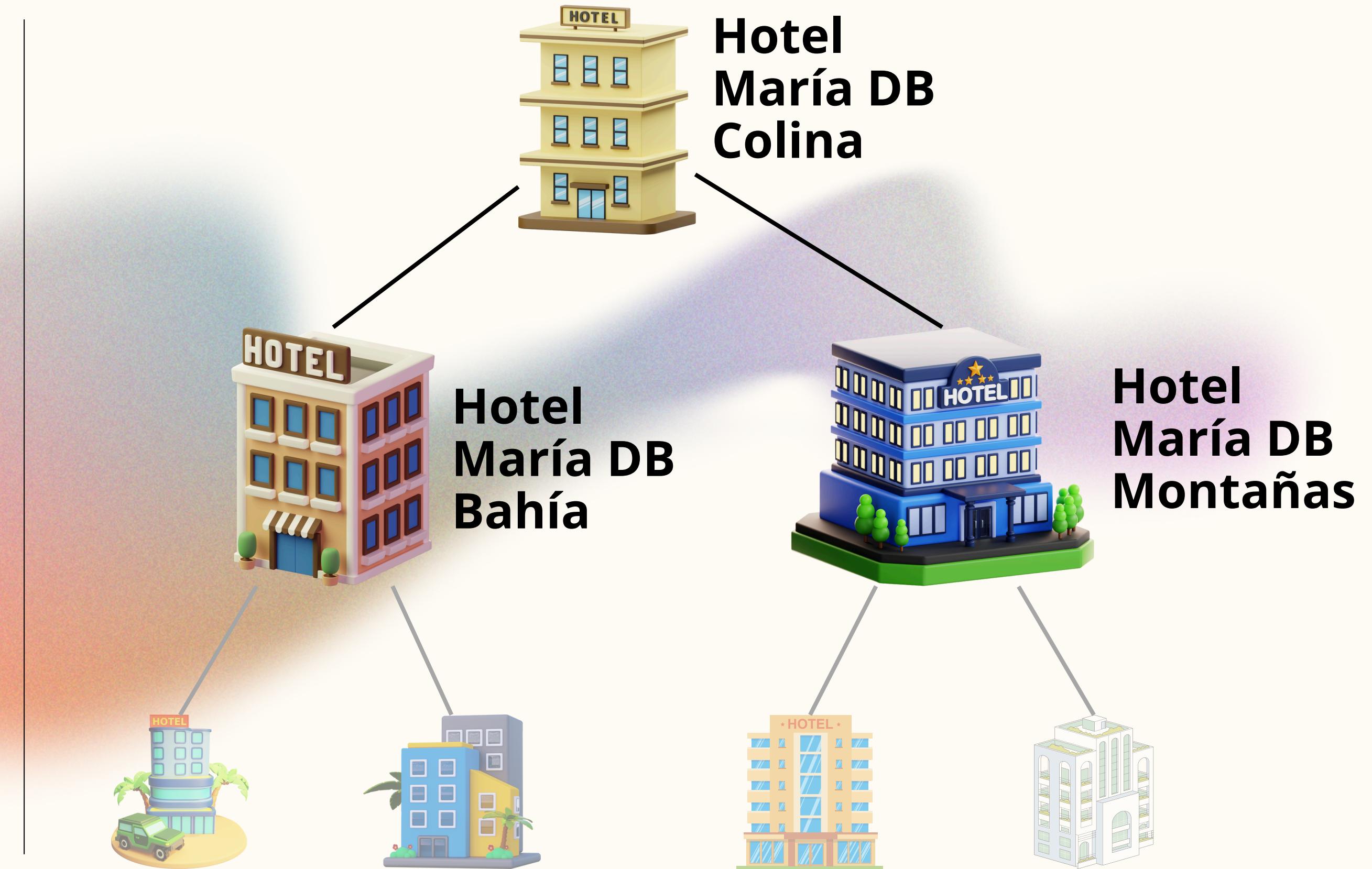


Reservas

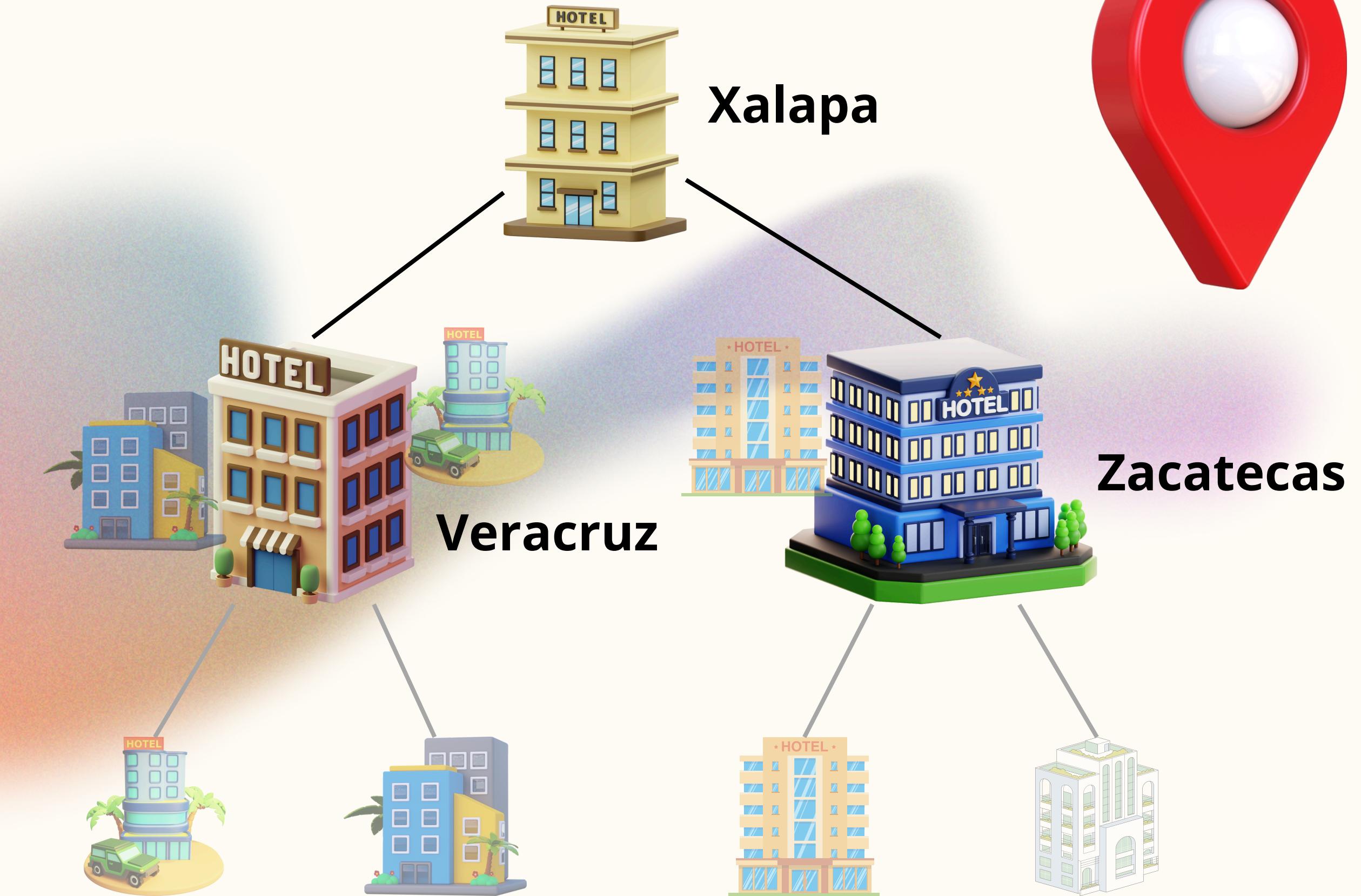
Usuarios



¿Cómo lo hice?



¿Cómo lo hice?



¿Cómo lo hice?



¿Cómo lo hice?



\$ 1, 700.°°



\$ 1, 750.°°



\$ 1, 850.°°



¿Cómo lo hice?

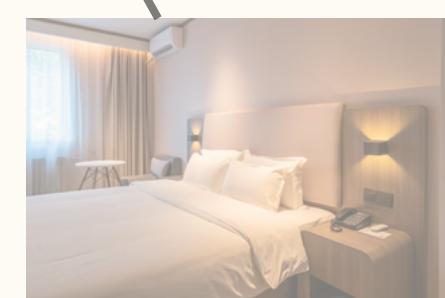


2025-05-20

2025-05-18

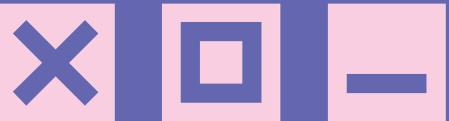


2025-05-24



¿Cómo lo hice?

GestorHoteles

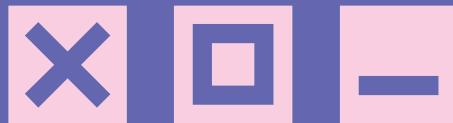


```
public class GestorHoteles {  
    // Árbol AVL para cada criterio de búsqueda (nombre, ciudad, categoría)  
    private ArbolAVL<Hotel> arbolPorID;  
    private ArbolAVL<Hotel> arbolPorNombre;  
    private ArbolAVL<Hotel> arbolPorCiudad;  
    private ArbolAVL<Hotel> arbolPorCategoria;  
  
    // Contador de IDs únicos para nuevos hoteles  
    private static int contadorHoteles = 1;  
  
    /**  
     * Constructor: inicializa Los árboles con Los comparadores correspondientes.  
     */  
    public GestorHoteles() {  
        this.arbolPorID = new ArbolAVL<>(new ComparadorIDHotel());  
        this.arbolPorNombre = new ArbolAVL<>(new ComparadorNombre());  
        this.arbolPorCiudad = new ArbolAVL<>(new ComparadorCiudad());  
        this.arbolPorCategoria = new ArbolAVL<>(new ComparadorCategoria());  
    }  
}
```



¿Cómo lo hice?

GestorHabitaciones



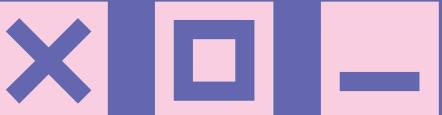
```
/*
 * Clase encargada de gestionar el registro y las búsquedas de habitaciones.
 */
public class GestorHabitaciones {
    private ArbolAVL<Habitacion> arbolPorPrecio;
    private ArbolAVL<Habitacion> arbolPorFecha;
    private ArbolAVL<Habitacion> arbolPorID;
    private int contadorHabitaciones;

    /**
     * Constructor que inicializa los árboles AVL con sus respectivos comparadores.
     */
    public GestorHabitaciones() {
        this.arbolPorPrecio = new ArbolAVL<>(new ComparadorPrecio());
        this.arbolPorFecha = new ArbolAVL<>(new ComparadorFechaDisponible());
        this.arbolPorID = new ArbolAVL<>(new ComparadorIDHabitacionH());
        this.contadorHabitaciones = 1;
    }
}
```



¿Cómo lo hice?

GestorUsuarios



```
import comparadores.ComparadorIDUsuario;
import estructuras.ArbolAVL;
import estructuras.Nodo;

public class GestorUsuarios {
    private ArbolAVL<Usuario> arbolUsuarios;
    private int contador = 1;

    public GestorUsuarios() {
        this.arbolUsuarios = new ArbolAVL<>(new ComparadorIDUsuario());
    }

    /**
     * Registra un nuevo usuario en el sistema con ID generado de manera automática.
     * @param nombre Nombre del usuario.
     * @param apellidoPaterno Apellido paterno.
     * @param apellidoMaterno Apellido materno.
     * @param email Correo electrónico.
```



¿Cómo lo hice?

GestorReservas

```
public class GestorReservas {  
  
    private GestorHabitaciones gestorHabitaciones;  
    private ArbolAVL<Habitacion> arbolHabitacionesPorID; // Árbol AVL de habitaciones ordenadas por ID  
    private ArbolAVL<Reserva> arbolReservasPorCliente; // Árbol AVL de reservas ordenadas por cliente  
    private ArbolAVL<Reserva> arbolReservasPorHabitacion; // Árbol AVL de reservas ordenadas por habitación  
    private ArbolAVL<Reserva> arbolReservasPorID; // Árbol AVL de reservas ordenadas por ID  
    private static int contadorReservas = 1; // Contador único para generar ID de reserva  
  
    /**  
     * Constructor que inicializa Los árboles AVL requeridos con sus respectivos comparadores  
     */  
    public GestorReservas(GestorHabitaciones gestorHabitaciones) {  
        this.gestorHabitaciones = gestorHabitaciones;  
        this.arbolHabitacionesPorID = gestorHabitaciones.getArbolPorID();  
  
        this.arbolReservasPorCliente = new ArbolAVL<>(new ComparadorIDUsuarioR());  
        this.arbolReservasPorHabitacion = new ArbolAVL<>(new ComparadorIDHabitacionR());  
        this.arbolReservasPorID = new ArbolAVL<>(new ComparadorIDReserva());  
    }  
}
```



Diagrama UML

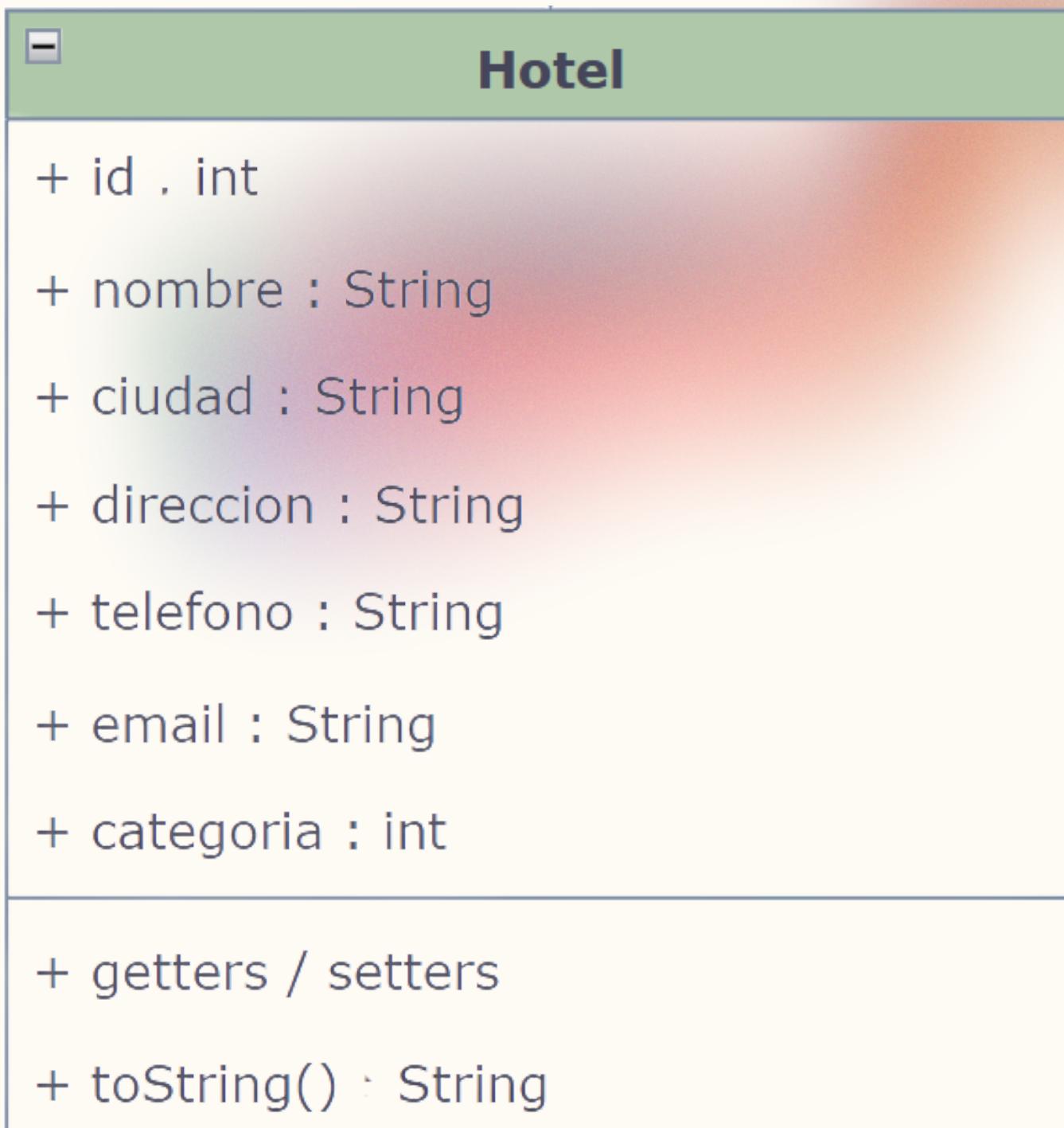


Diagrama UML

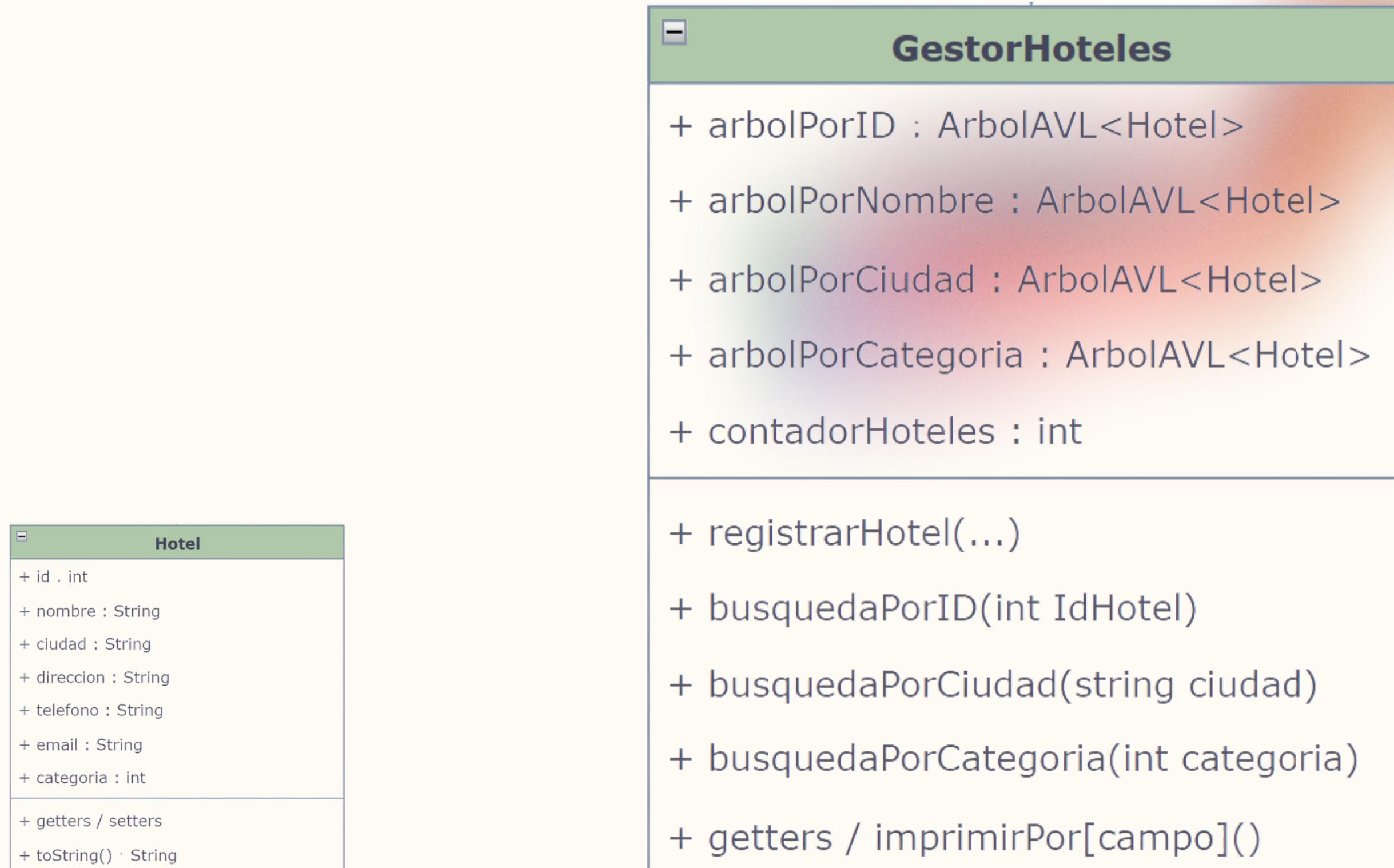


Diagrama UML

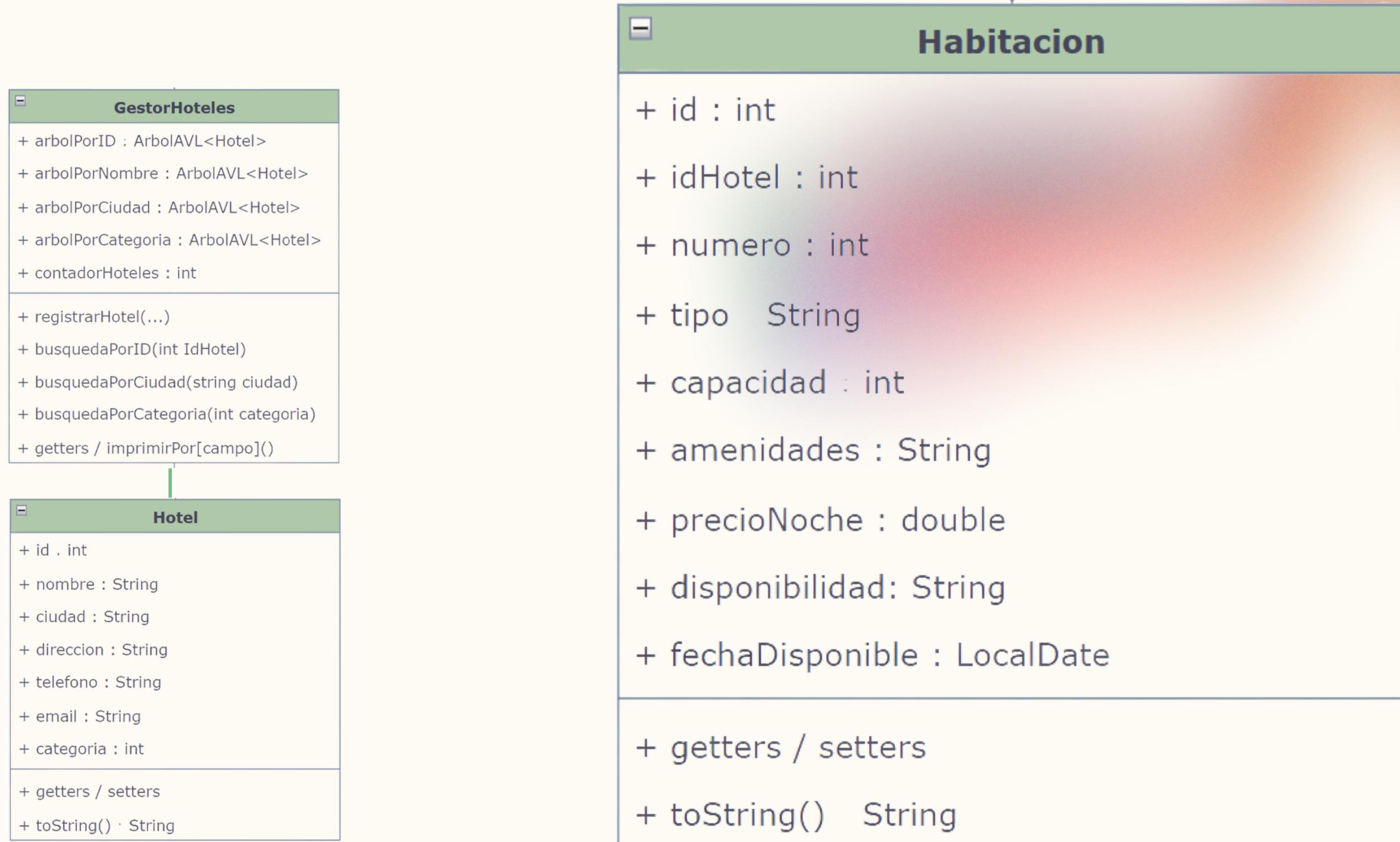


Diagrama UML

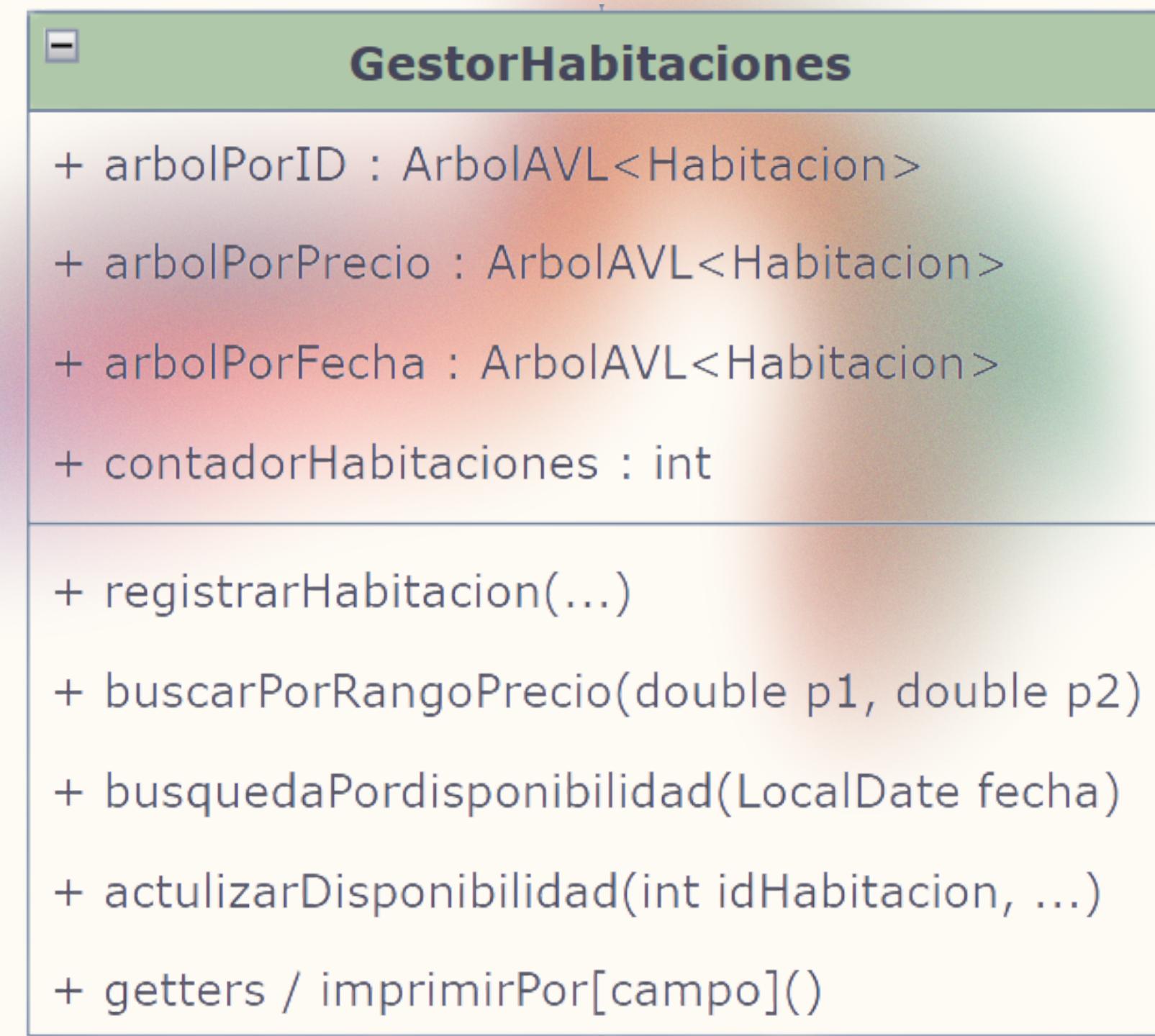
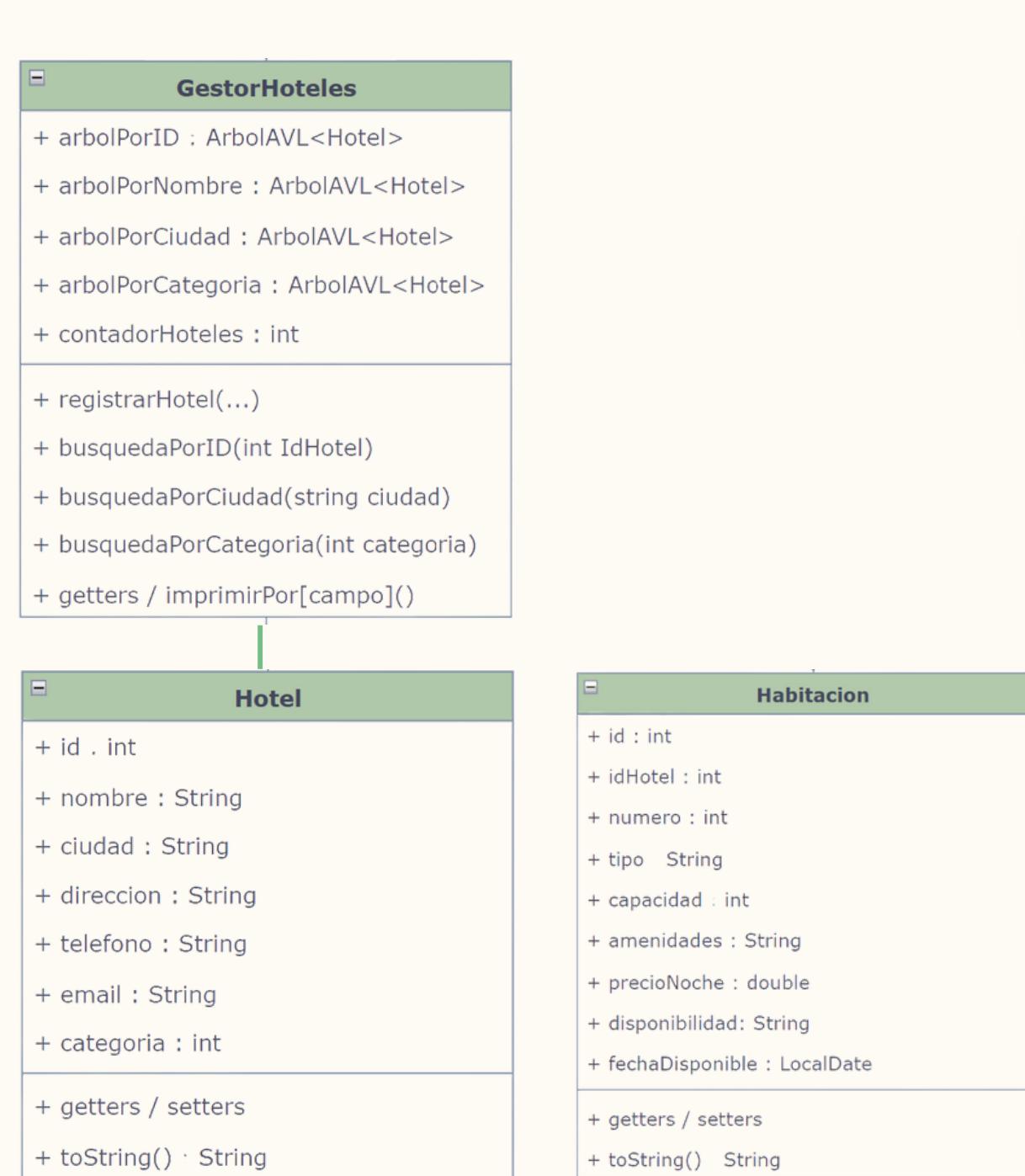


Diagrama UML

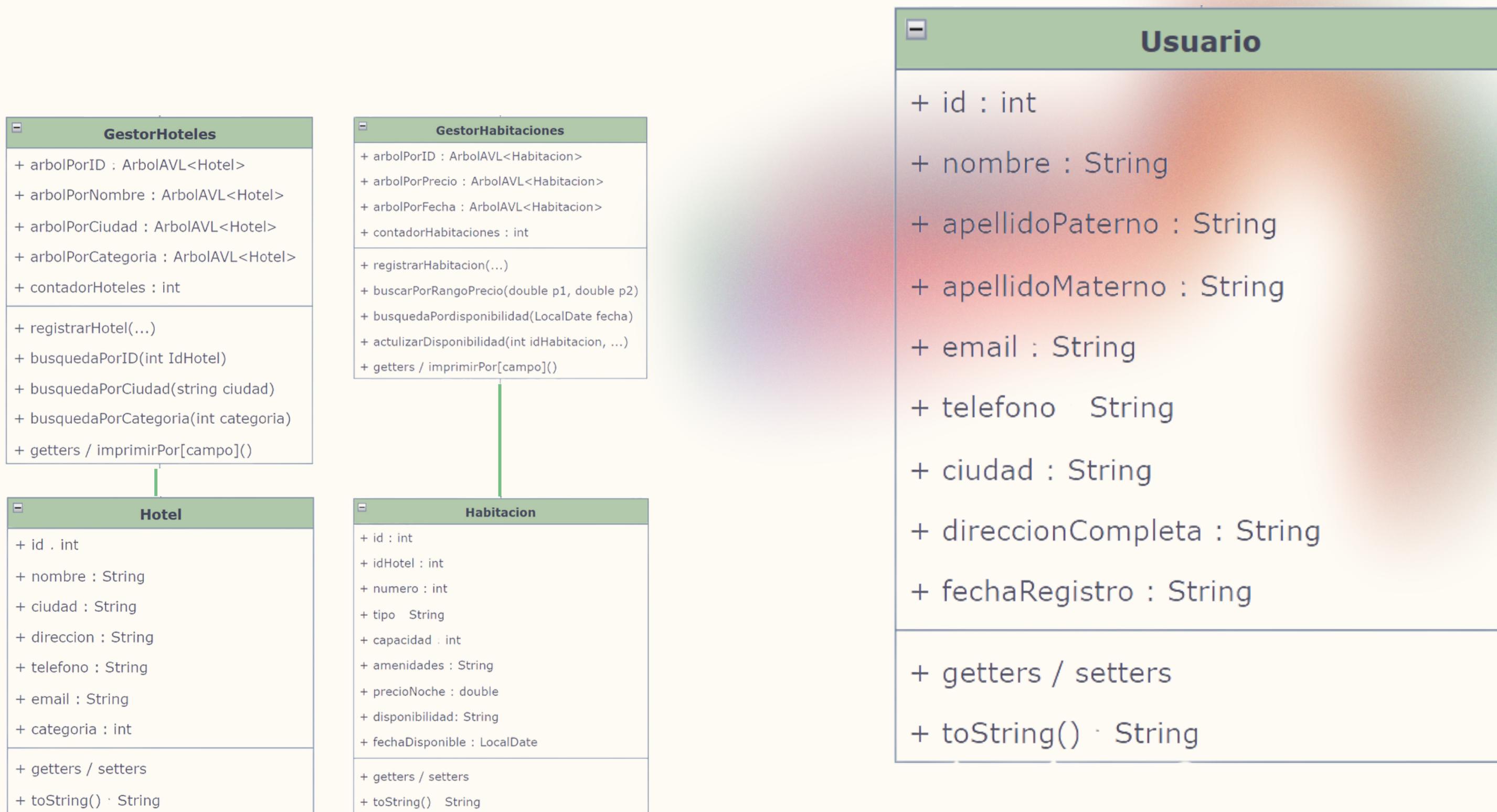


Diagrama UML

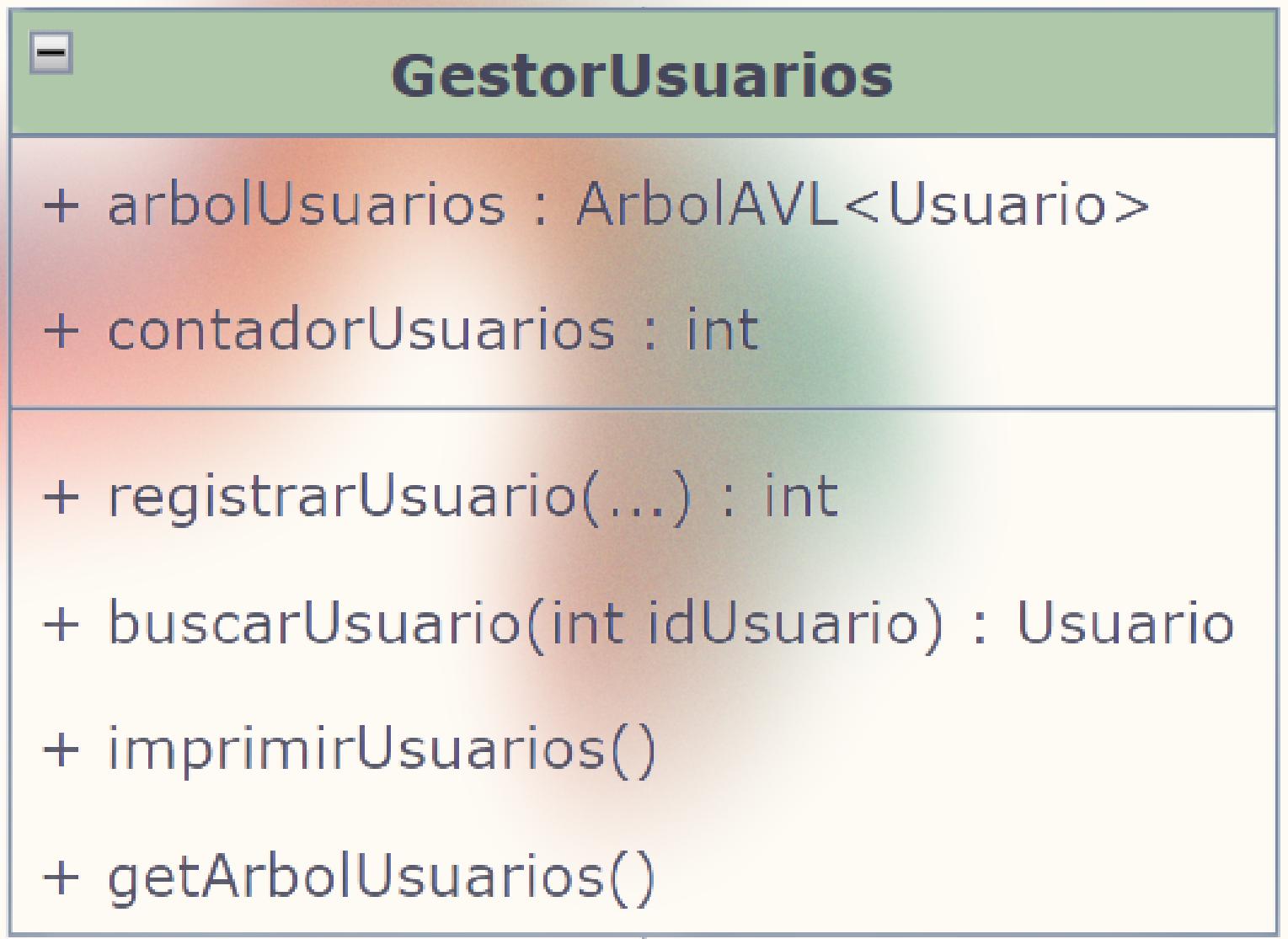
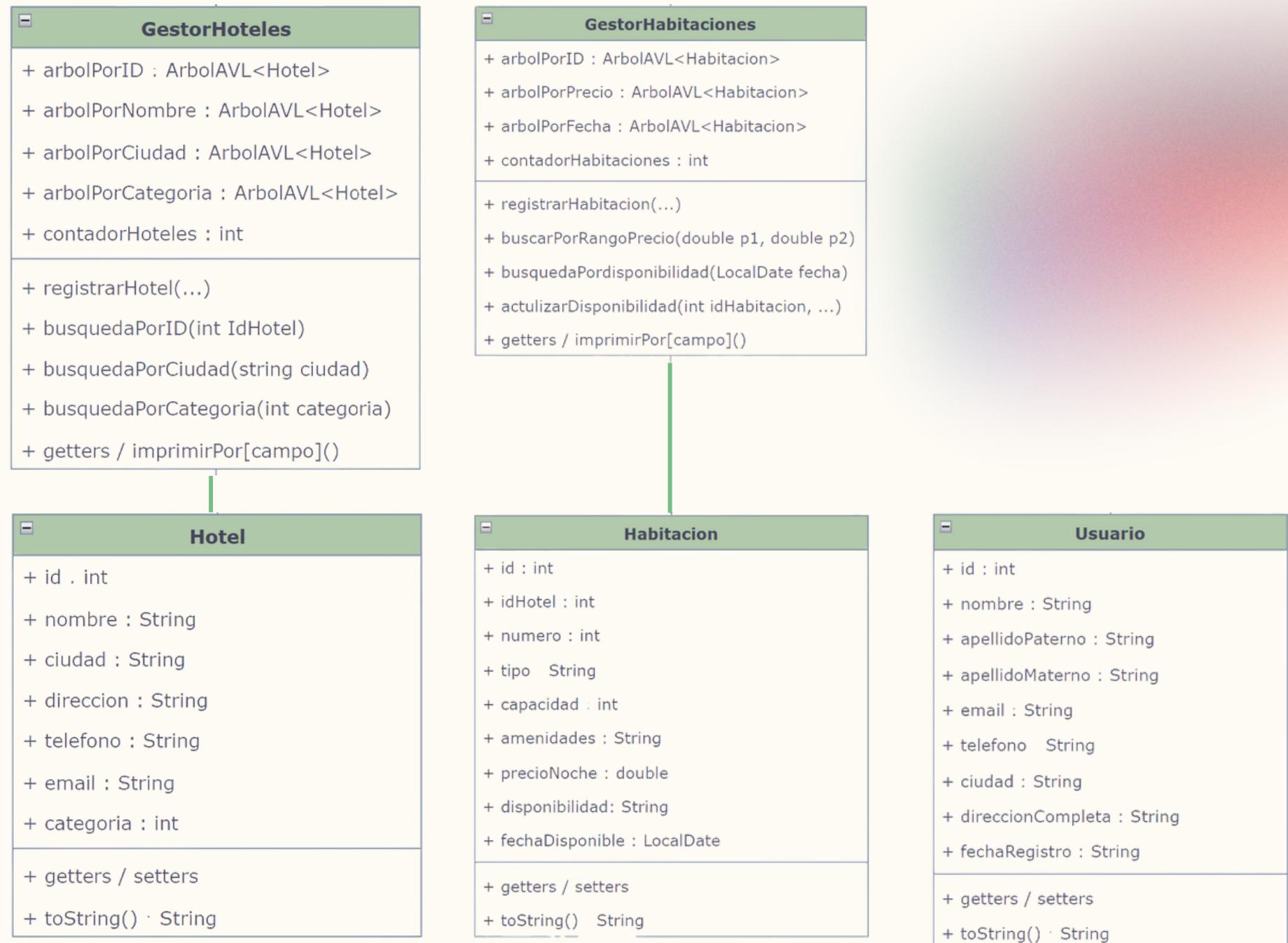


Diagrama UML

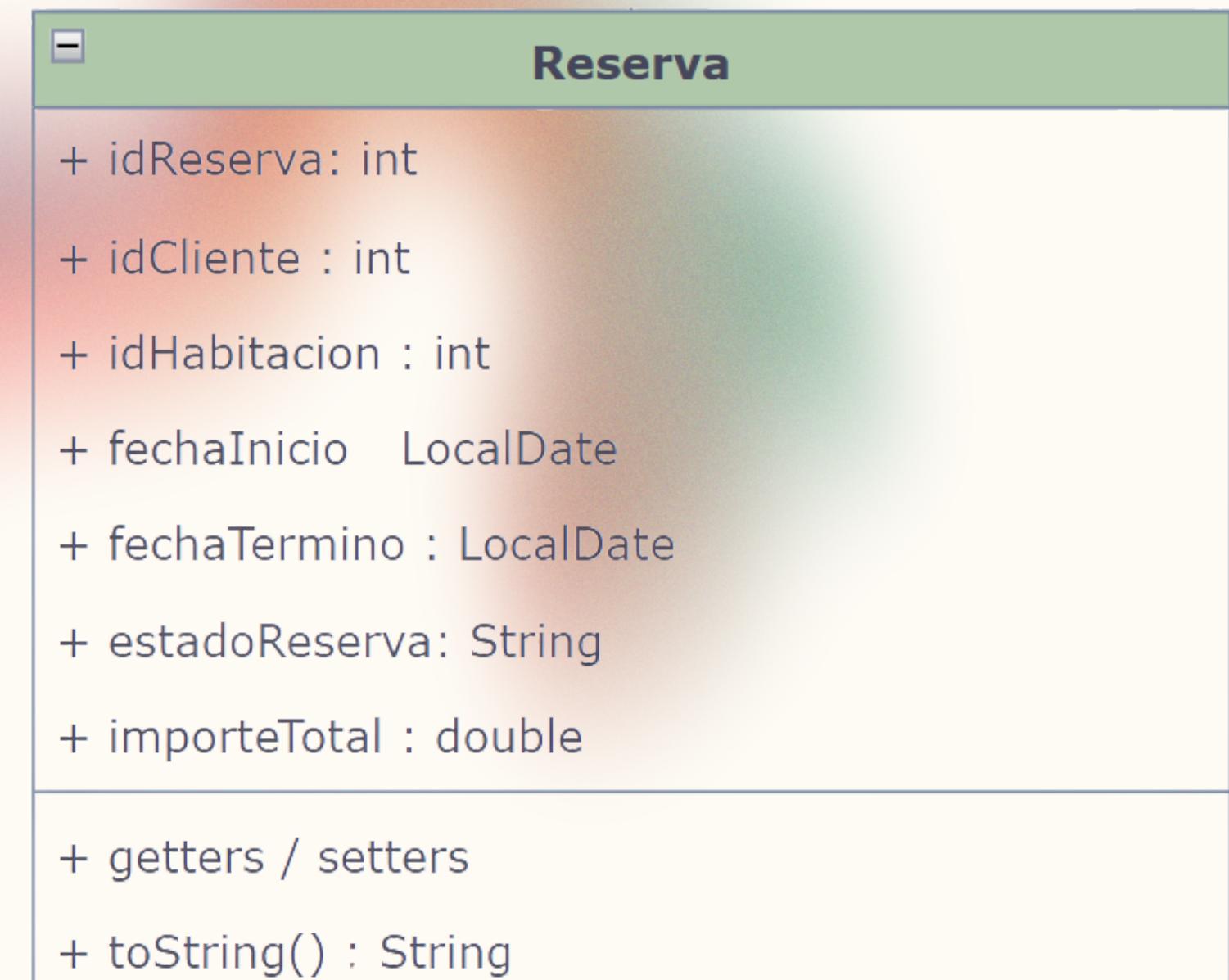
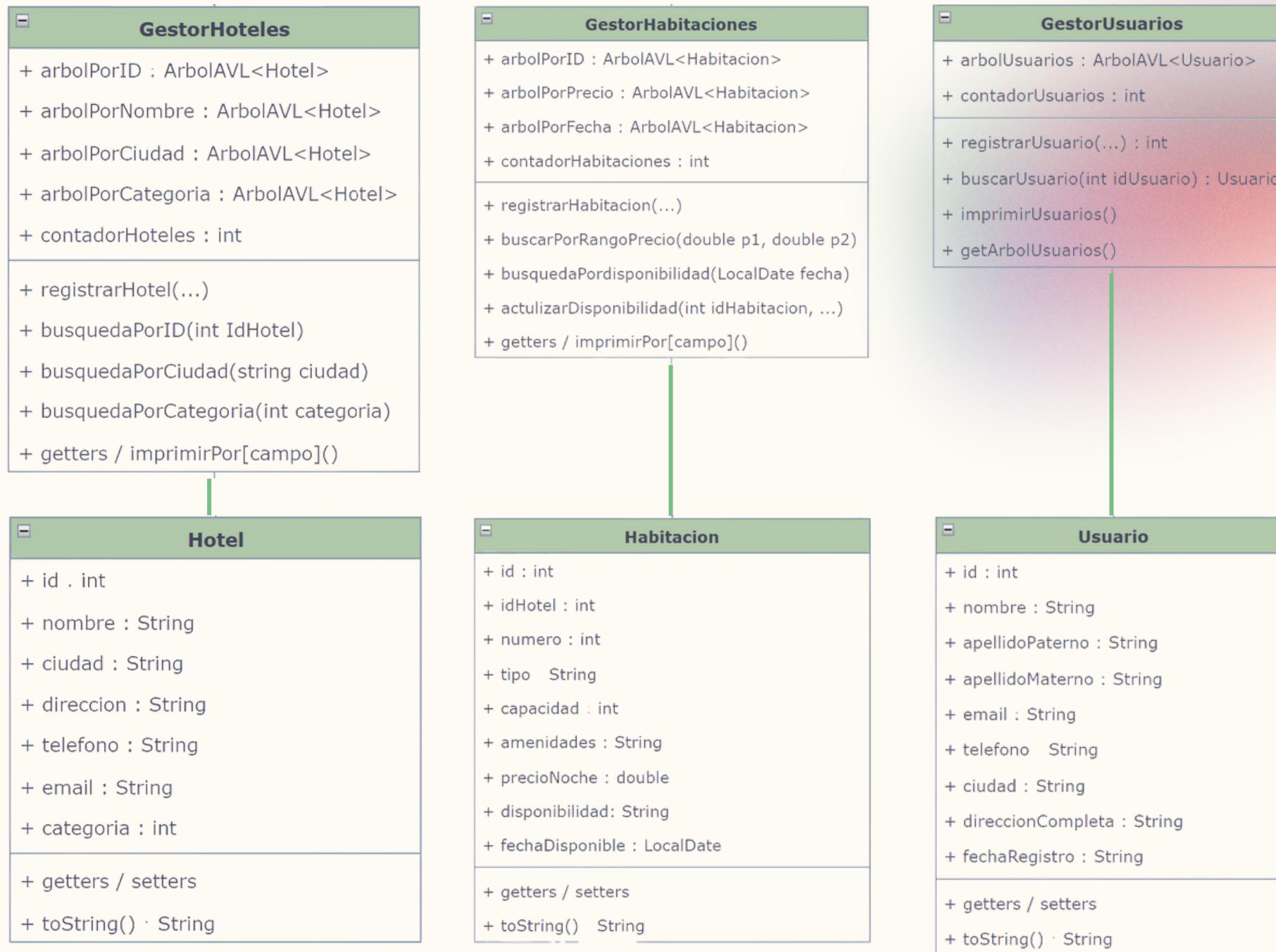


Diagrama UML

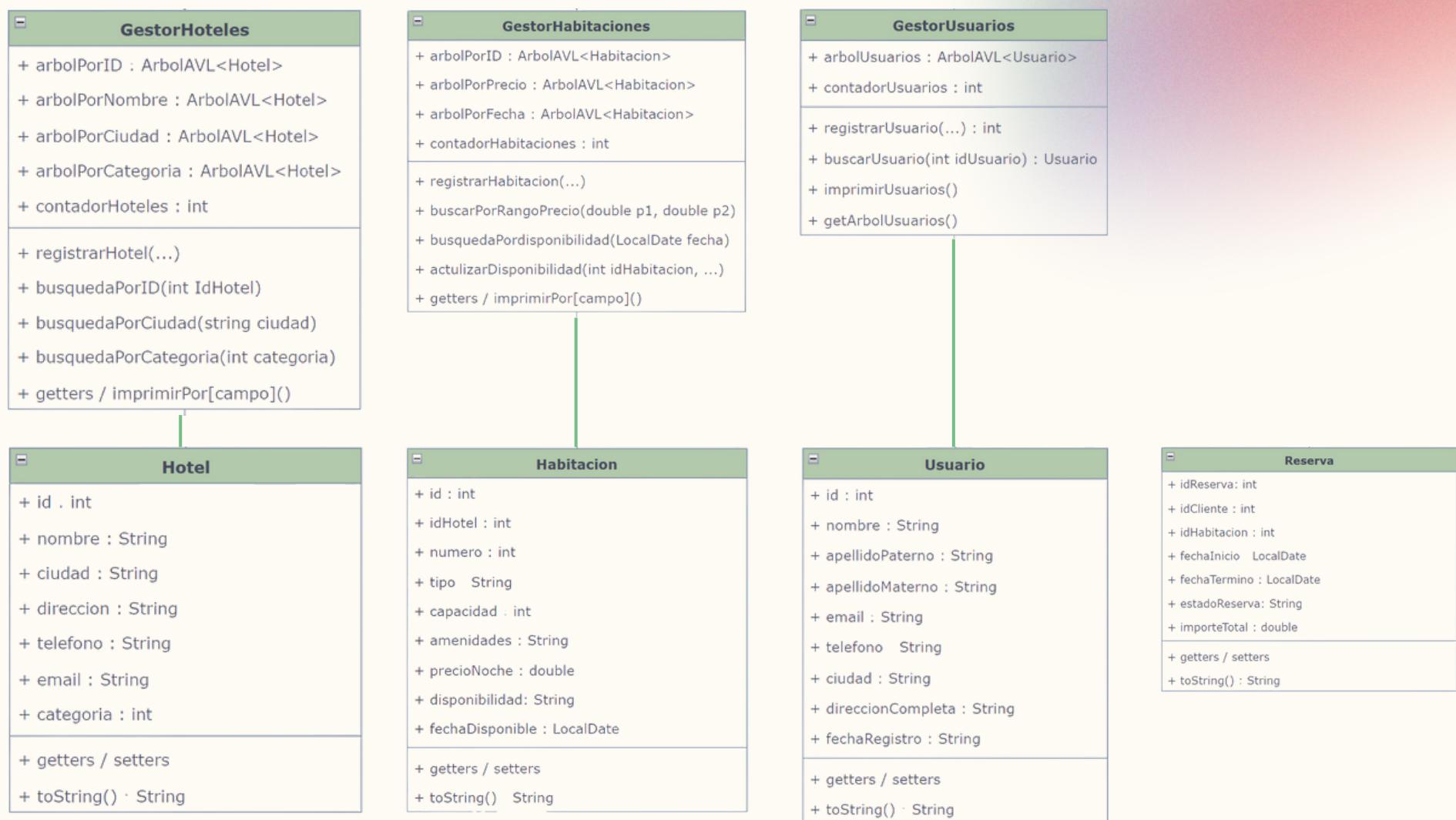


Diagrama UML

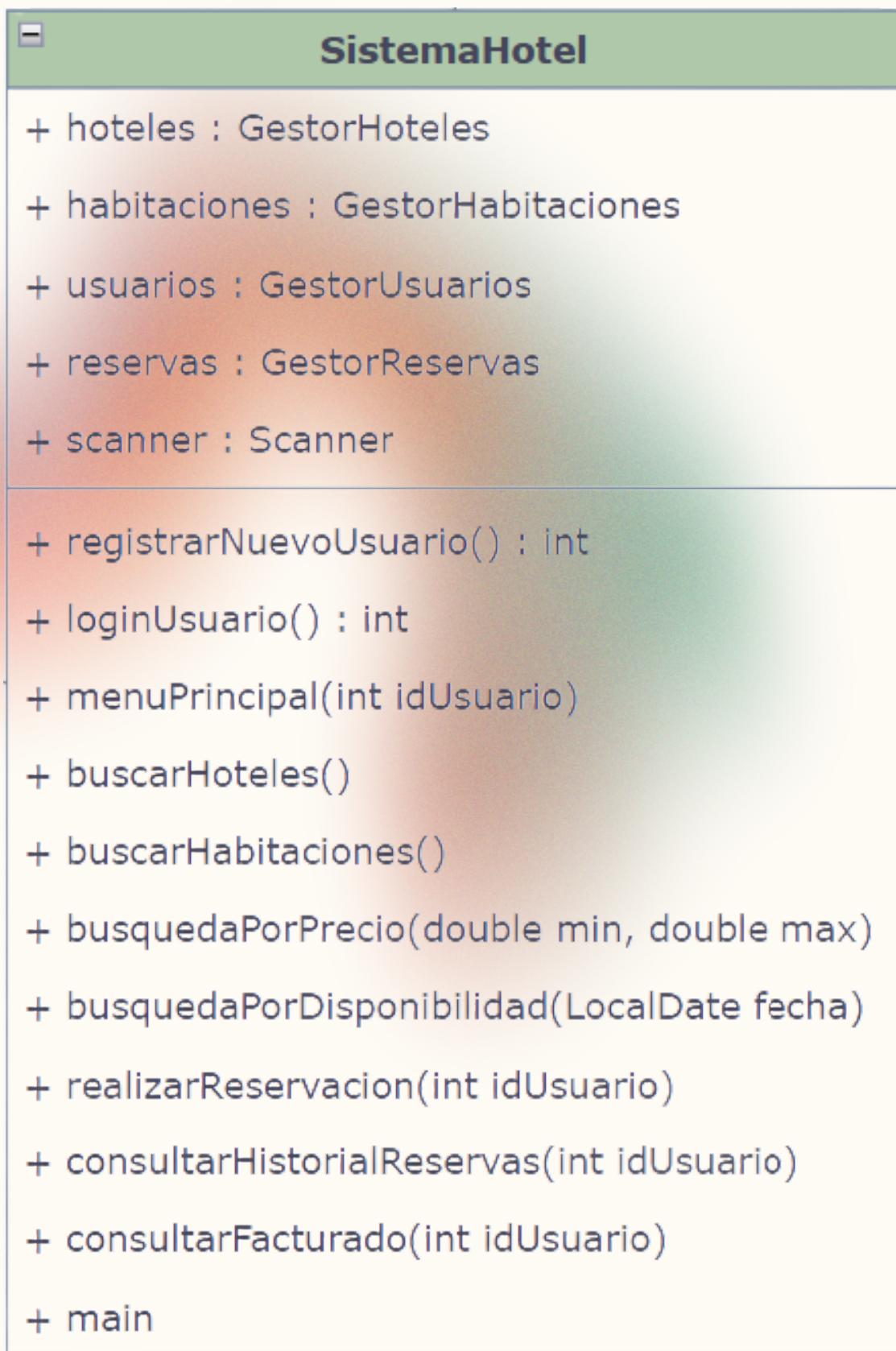
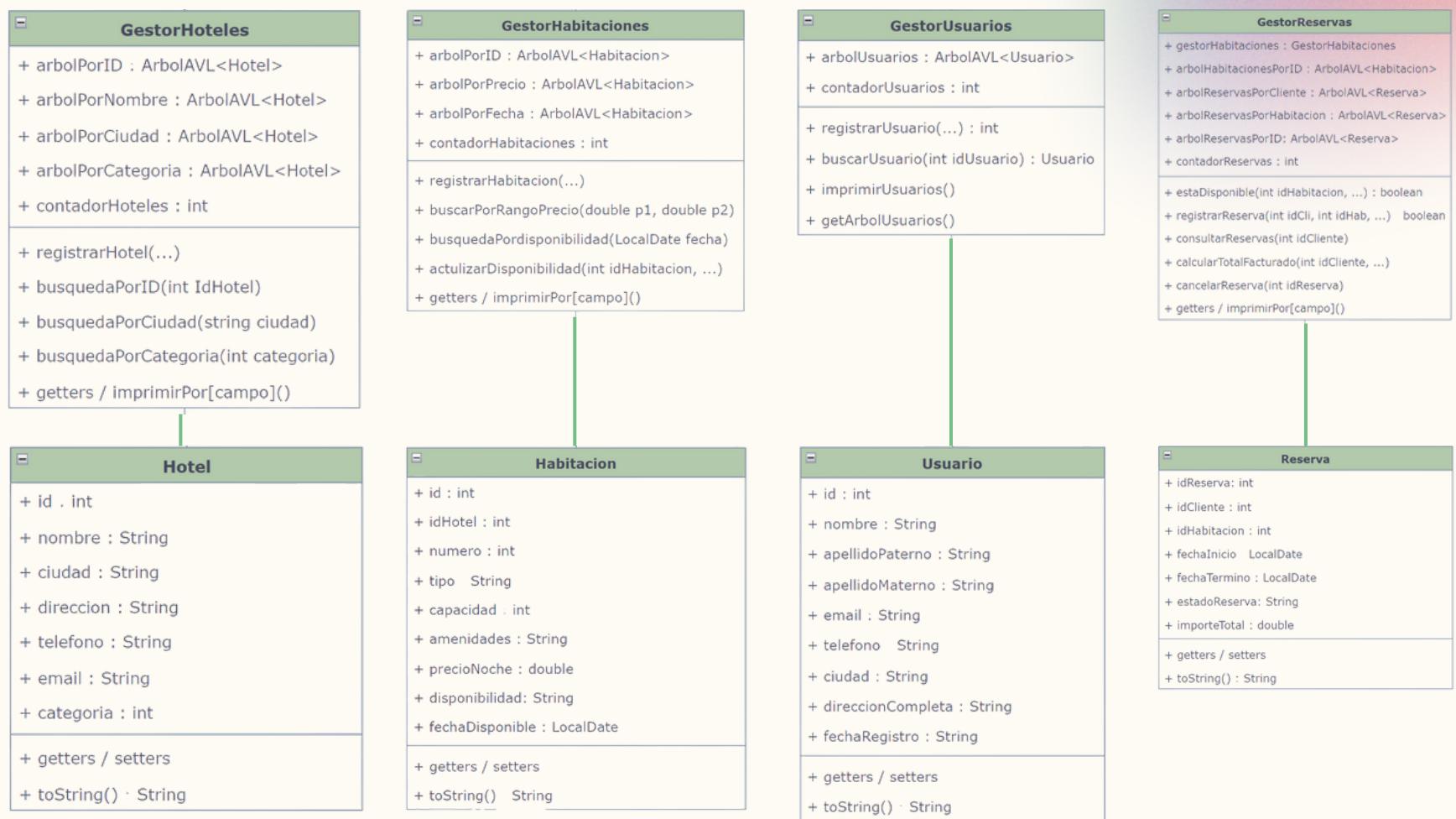


Diagrama UML

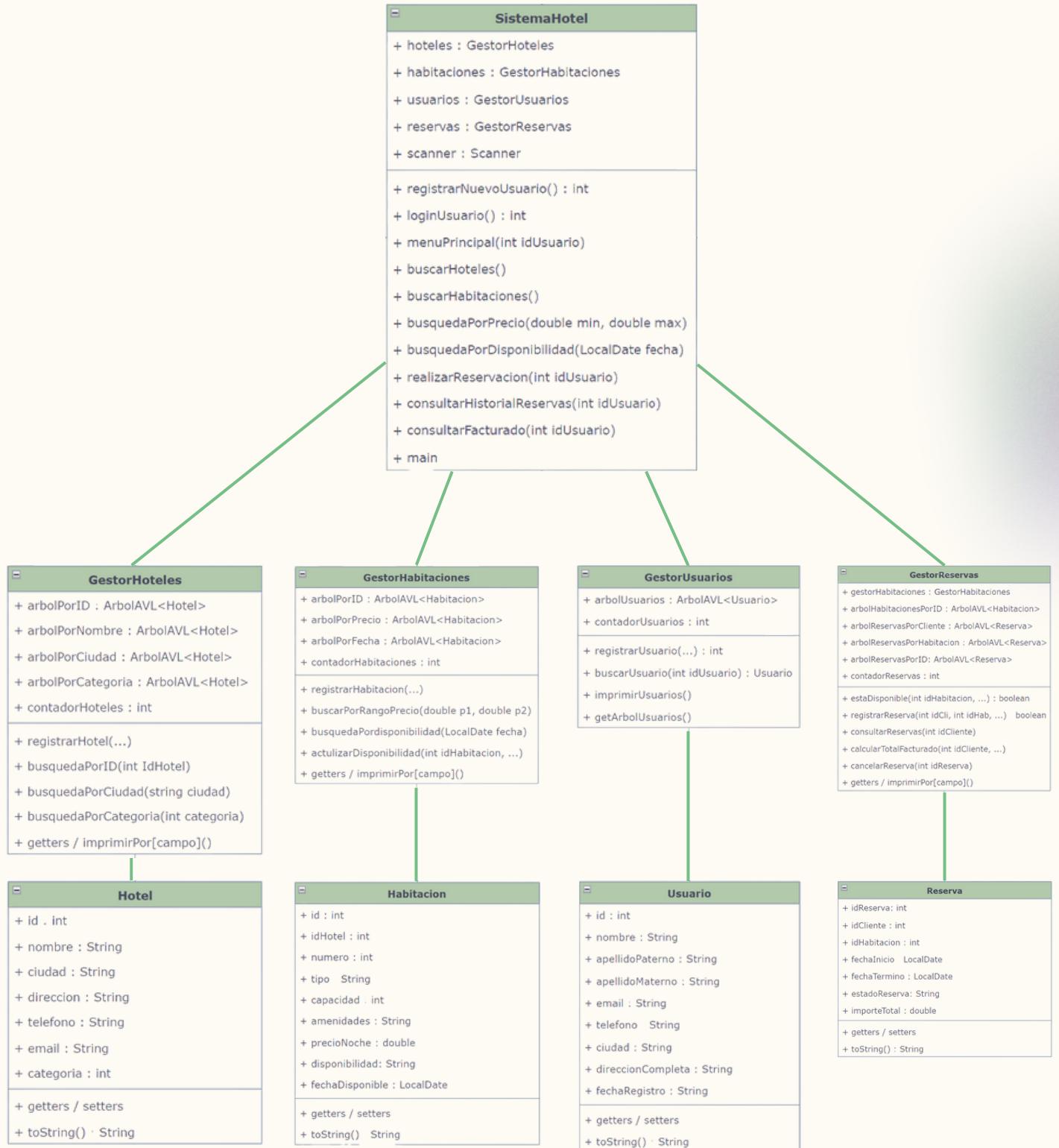


Diagrama UML

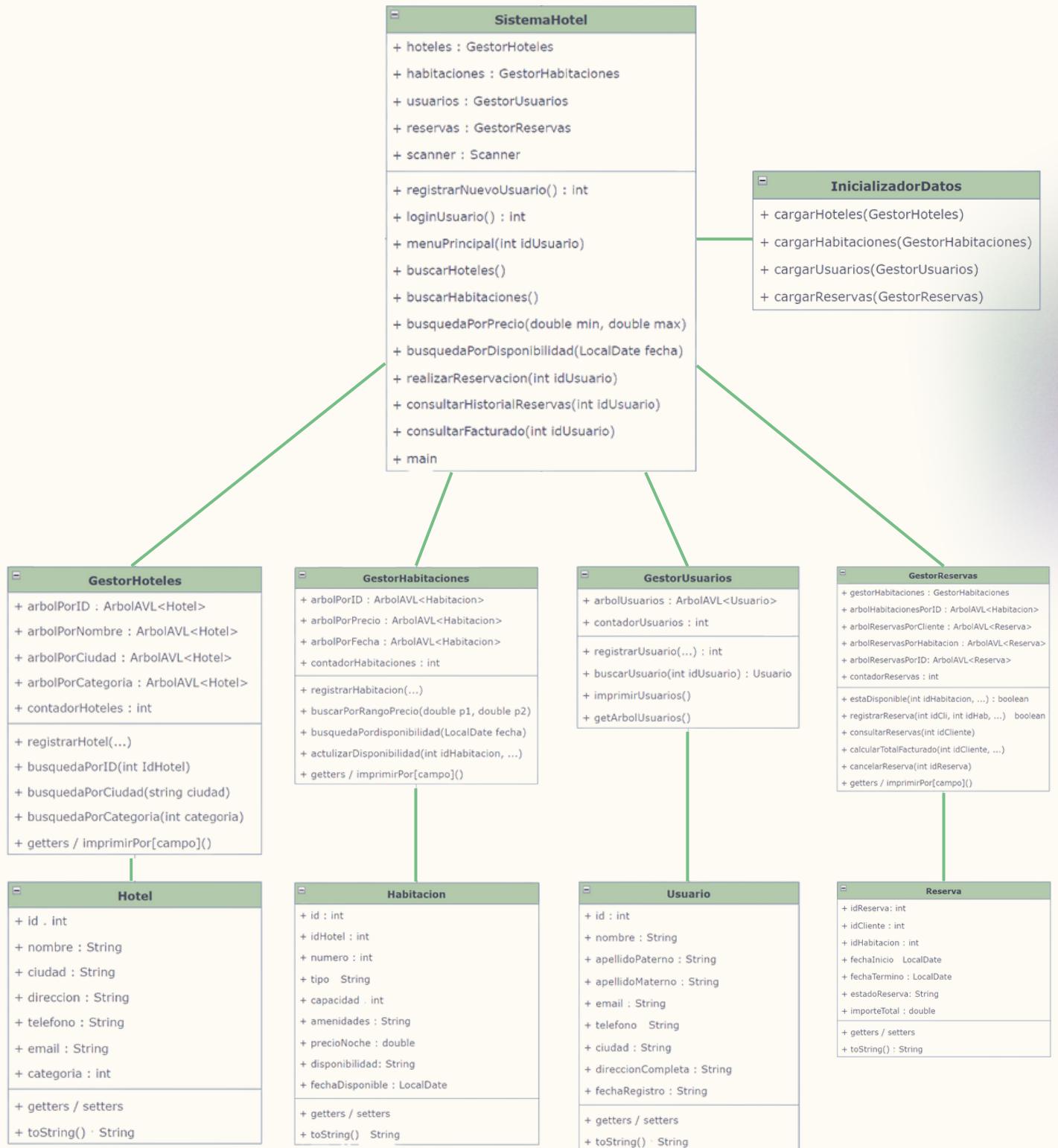


Diagrama UML

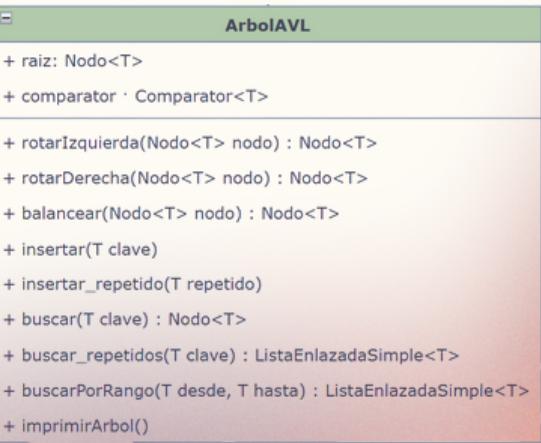
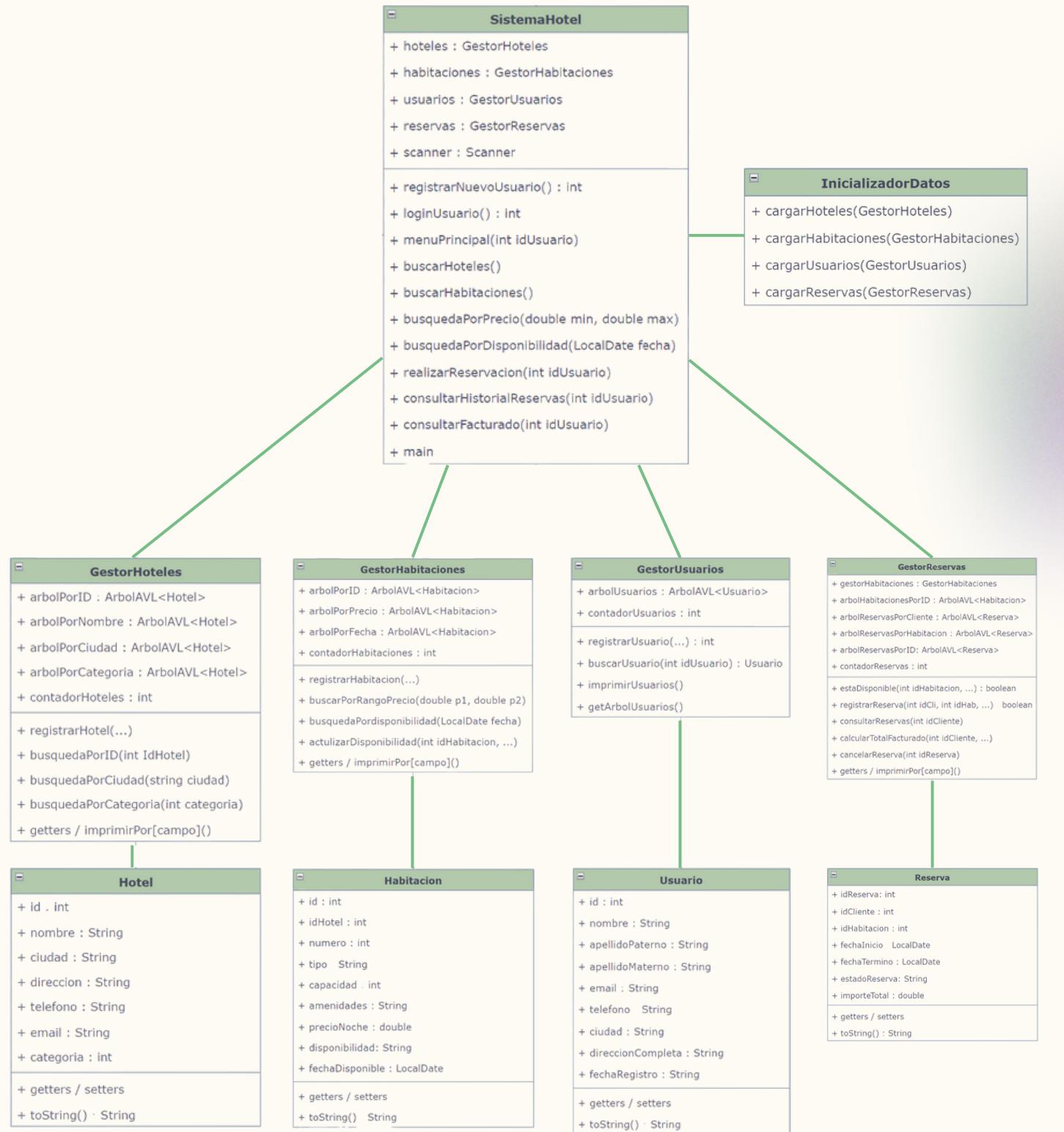
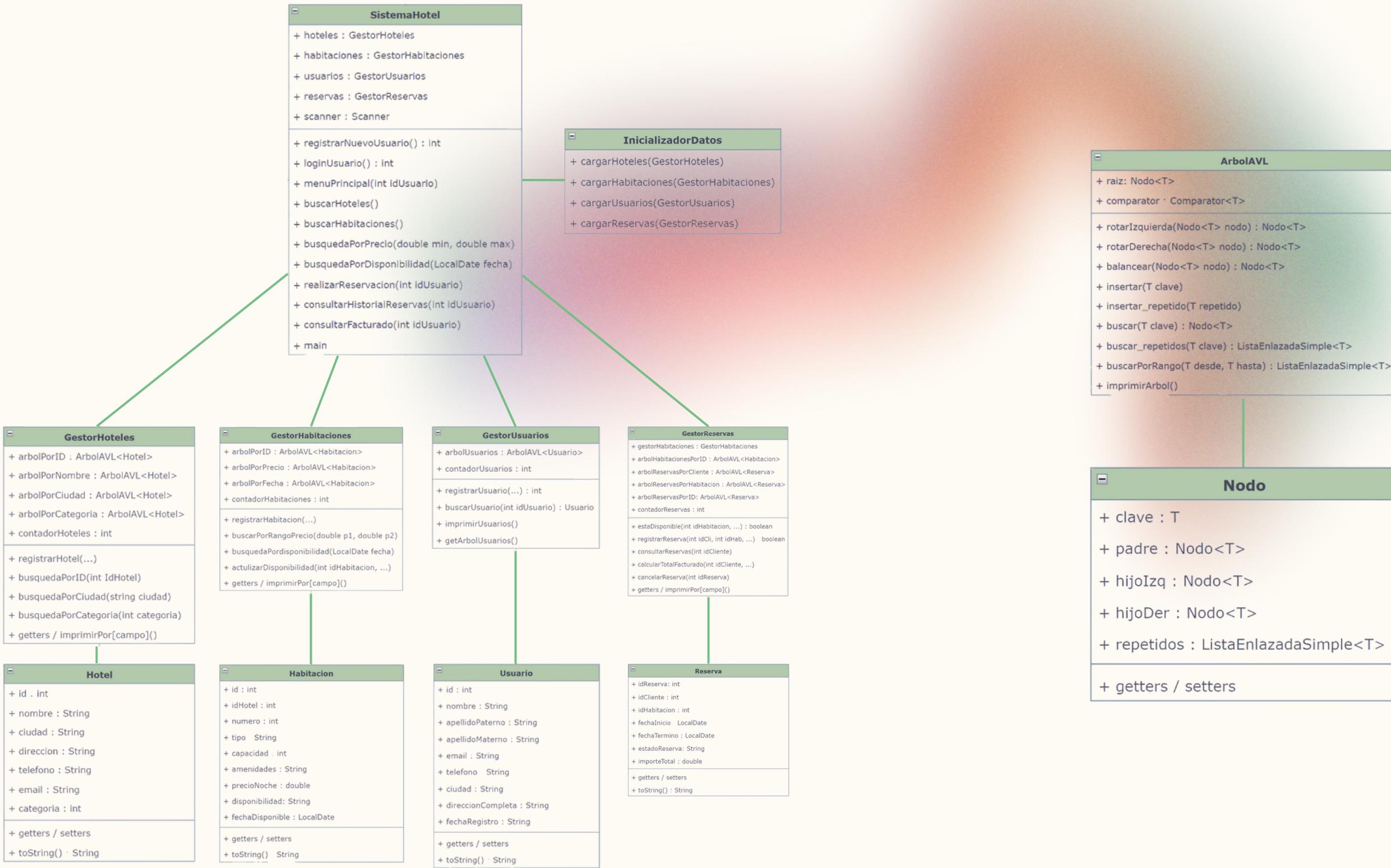


Diagrama UML



Organización de los archivos

```
|--- src
  |--- app
    |--- InicializadorDatos.java
    |--- SistemaHotel.java
  |--- comparadores
    |--- [...]
  |--- estructuras
    |--- ArbolAVL.java
    |--- ListaEnlazadaSimple.java
    |--- Nodo.java
    |--- TDABinaryTree.java
    |--- TDALista.java
  |--- modelo
    |--- GestorHabitaciones.java
    |--- GestorHoteles.java
    |--- GestorReservas.java
    |--- GestorUsuarios.java
    |--- Habitacion.java
    |--- Hotel.java
    |--- Reserva.java
    |--- Usuario.java
  |--- pruebas
    |--- [...]
```

Referencias

- Peláez, C. (2018). *Estructuras de Datos con Java Moderno*. UNAM.
- Gosling, J., et al. (2024). *The Java Language Specification*. Oracle America, Inc.
- Oracle Docs: [Interface Comparator<T>](#).
- Brass, P. (2008). *Advanced Data Structures*. Cambridge University Press.

Sesión de preguntas

¡Gracias por escuchar!