PROGRAMACIÓN ORIENTADA A OBJETOS + INTEGRACIÓN WEB

Agencia ViajandoAndo

Aldo Paz Mendiola - A01802995
Lulio Derek Ruiz Carrillo - A01825096

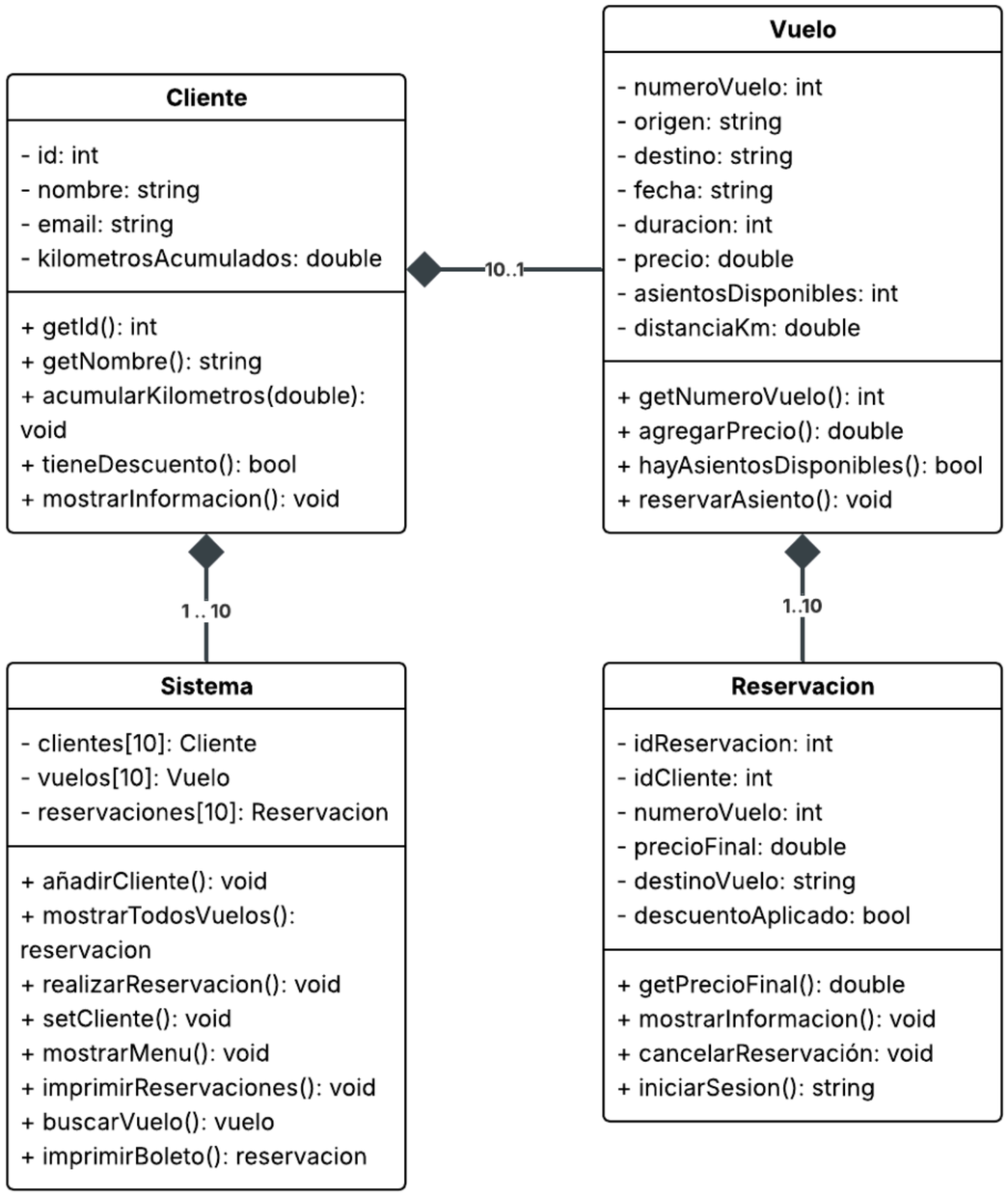
En esta presentación

Se detallará lo superficial del código generado con posible integración web, e impresión de un ticket

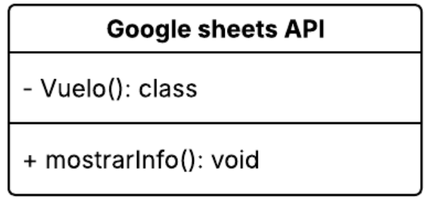
01 Diagrama de clases

02 Codigos

03 Muestra



- Encapsulamiento: Atributos privados con getters/setters



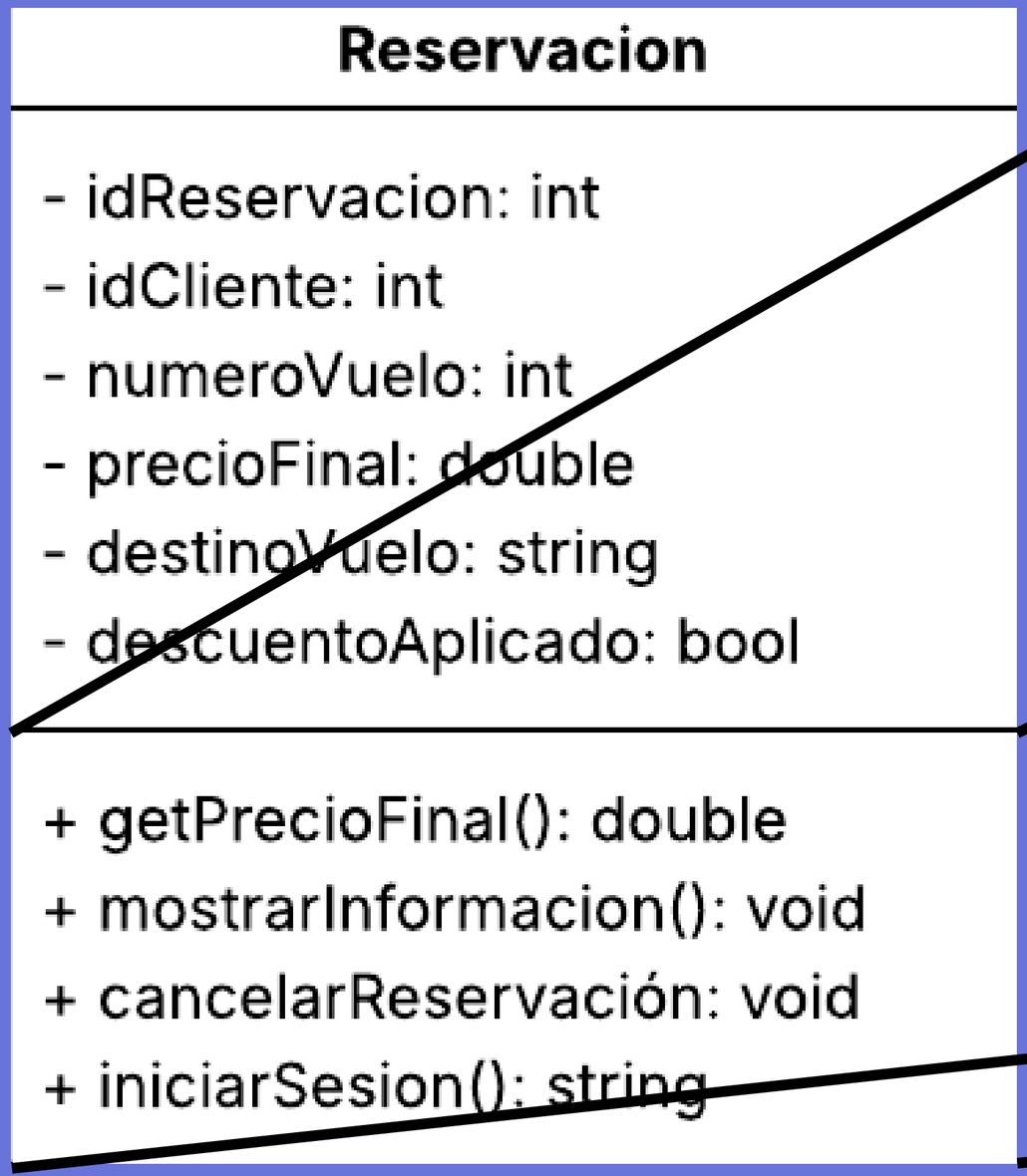
- Abstracción: Cada clase representa una entidad del mundo real
- Modularidad: Cada clase en su propio archivo

Reservacion

- idReservacion: int
- idCliente: int
- numeroVuelo: int
- precioFinal: double
- destinoVuelo: string
- descuentoAplicado: bool

+ getPrecioFinal(): double
+ mostrarInformacion(): void
+ cancelarReservación: void
+ iniciarSesion(): string

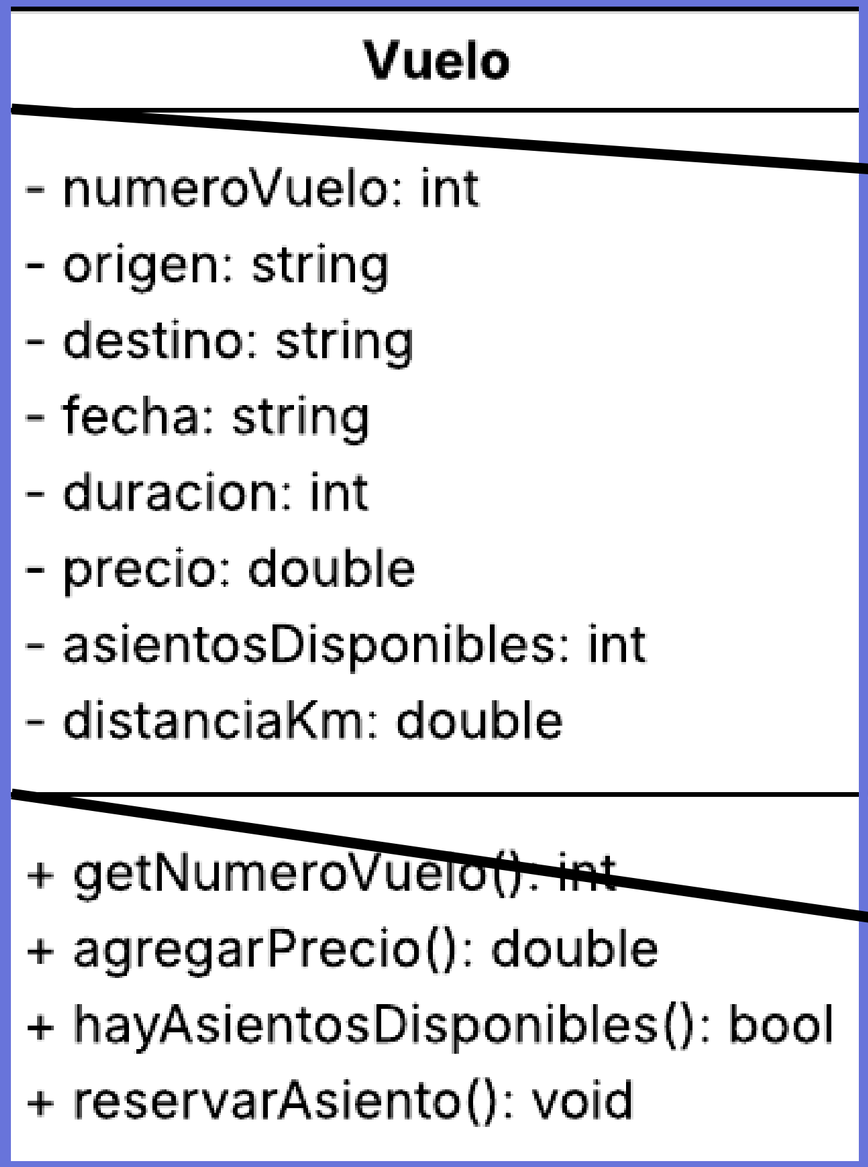
```
class Reservacion {
private:
    int idReservacion;
    int idCliente;
    int numeroVuelo;
    double precioFinal;
    string destinoVuelo;
    bool descuentoAplicado;
```



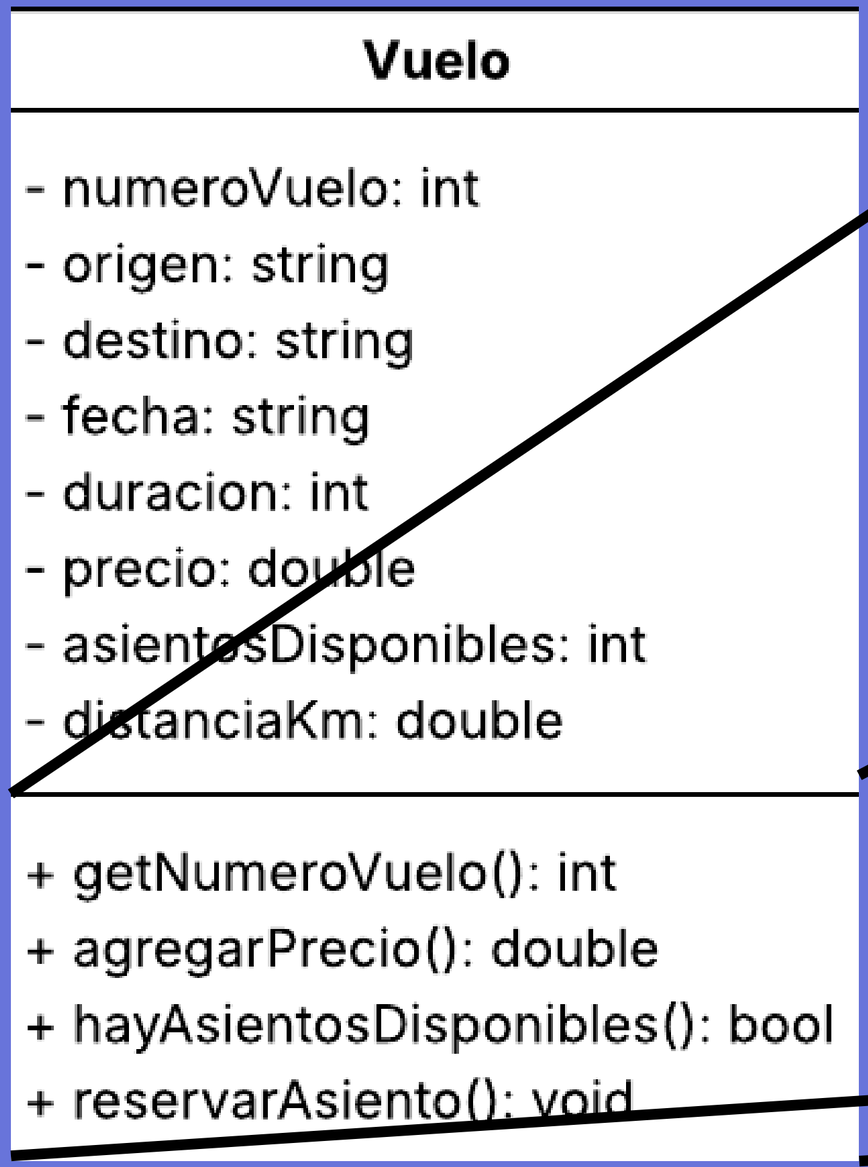

```
public:
    // Constructor
    Reservacion();
    Reservacion(int idReservacion, int idCliente, int numeroVuelo,
                double precioFinal, string destinoVuelo, bool descuentoAplicado);

    // Getters
    int getIdReservacion() const;
    int getIdCliente() const;
    int getNumeroVuelo() const;
    double getPrecioFinal() const;
    string getDestinoVuelo() const;
    bool getDescuentoAplicado() const;

    // Setters
    void setIdReservacion(int id);
    void setIdCliente(int idCliente);
    void setNumeroVuelo(int numeroVuelo);
    void setPrecioFinal(double precio);
    void setDestinoVuelo(string destino);
    void setDescuentoAplicado(bool descuento);

    // Métodos específicos
    void mostrarInformacion() const;
    void cancelarReservacion();
    string iniciarSesion();
};
```

```cpp
// Constructor por defecto
Reservacion::Reservacion() {
    idReservacion = 0;
    idCliente = 0;
    numeroVuelo = 0;
    precioFinal = 0.0;
    destinoVuelo = "";
    descuentoAplicado = false;
}

// Constructor con parámetros
Reservacion::Reservacion(int idReservacion, int idCliente, int numeroVuelo,
                         double precioFinal, string destinoVuelo, bool descuentoAplicado) {
    this->idReservacion = idReservacion;
    this->idCliente = idCliente;
    this->numeroVuelo = numeroVuelo;
    this->precioFinal = precioFinal;
    this->destinoVuelo = destinoVuelo;
    this->descuentoAplicado = descuentoAplicado;
}
```

```cpp
// Métodos específicos
void Reservacion::mostrarInformacion() const {
    cout << "\n=== RESERVACION #" << idReservacion << " ===" << endl;
    cout << "ID Cliente: " << idCliente << endl;
    cout << "Numero de Vuelo: " << numeroVuelo << endl;
    cout << "Destino: " << destinoVuelo << endl;
    cout << "Precio Final: $" << precioFinal << endl;
    if (descuentoAplicado) {
        cout << "*** DESCUENTO DE 40% APLICADO ***" << endl;
    }
    cout << "================================\n" << endl;
}

void Reservacion::cancelarReservacion() {
    cout << "\nReservacion #" << idReservacion << " ha sido cancelada." << endl;
    cout << "Se ha procesado el reembolso de $" << precioFinal << endl;
}

string Reservacion::iniciarSesion() {
    return "Sesion iniciada para cliente con ID: " + to_string(idCliente);
}
```



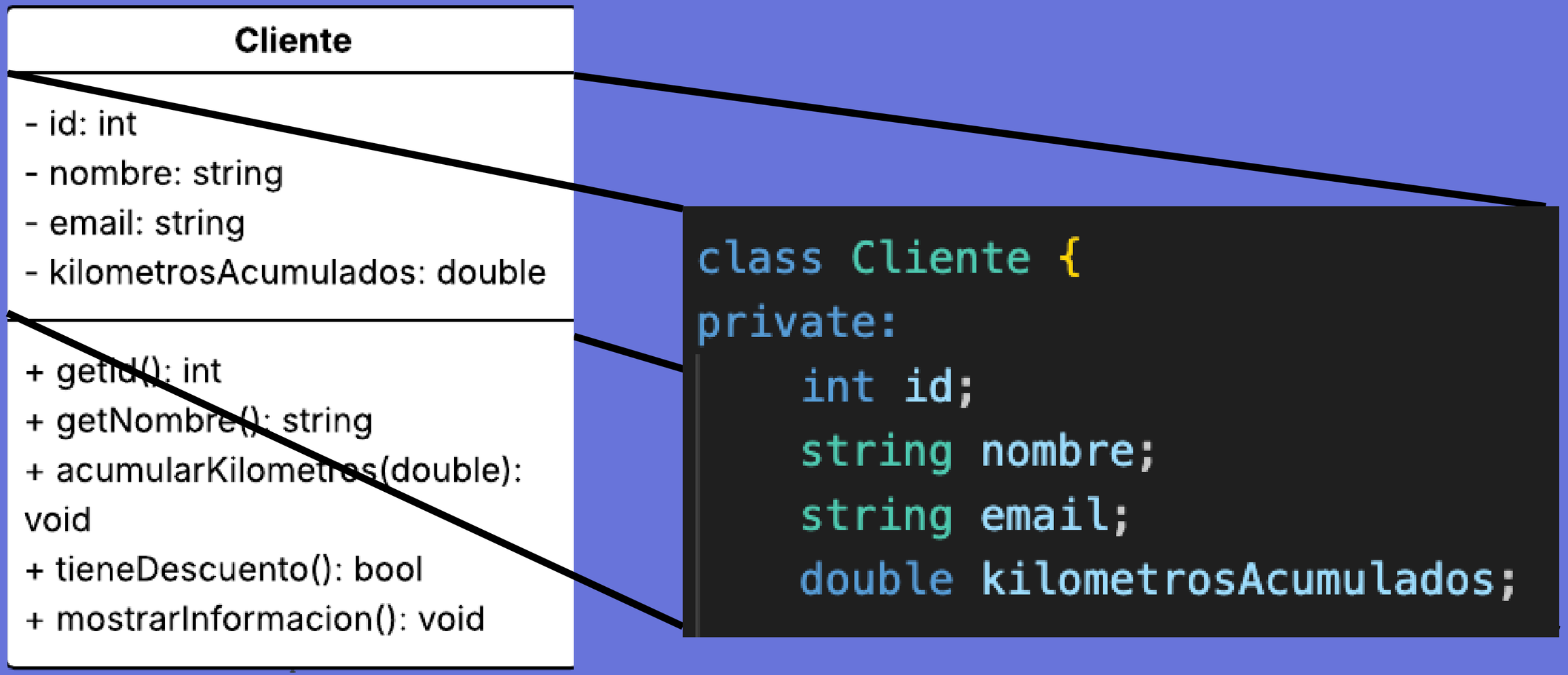

```
class Vuelo {
private:
    int numeroVuelo;
    string origen;
    string destino;
    string fecha;
    int duracion;
    double precio;
    int asientosDisponibles;
    double distanciaKm;
```

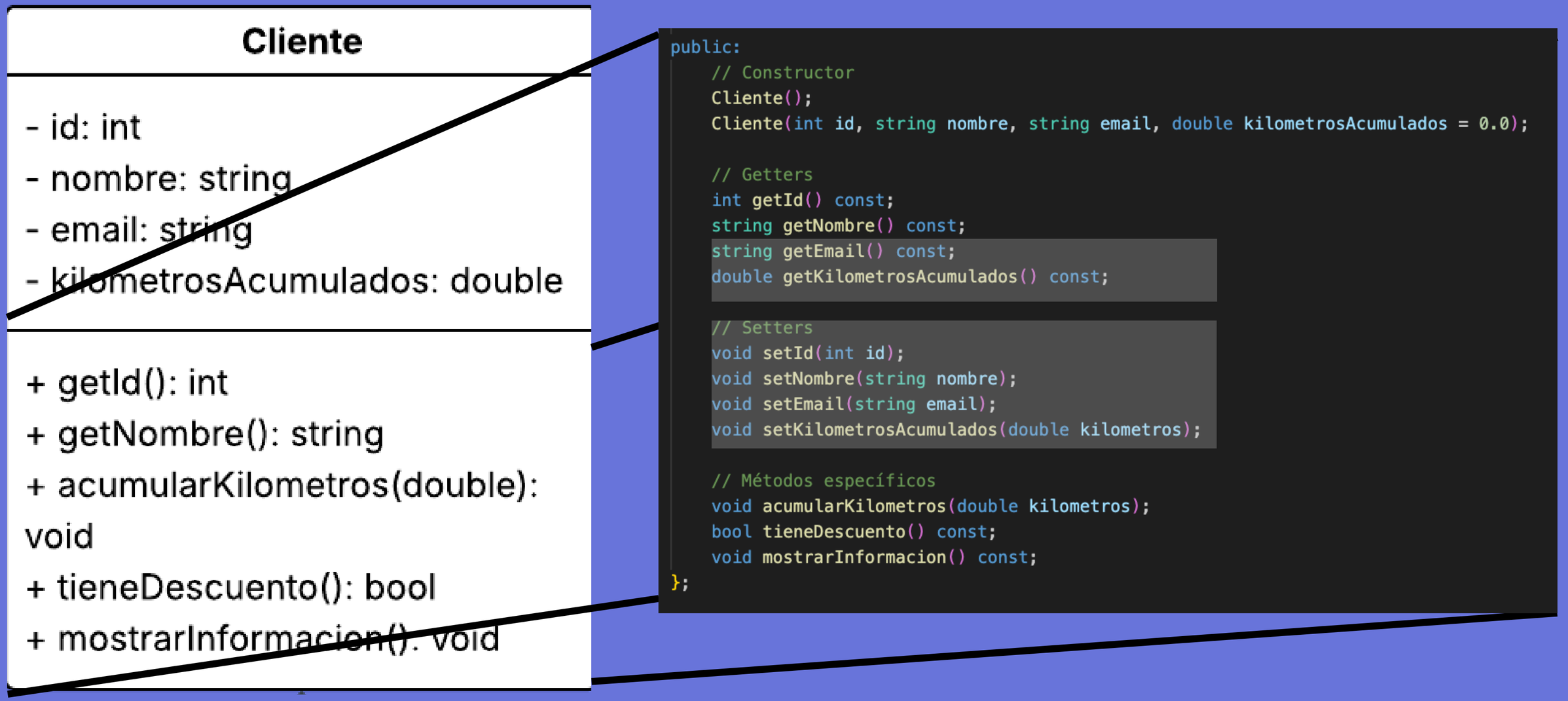



```cpp
public:
    // Constructor
    Vuelo();
    Vuelo(int numeroVuelo, string origen, string destino, string fecha,
          int duracion, double precio, int asientosDisponibles, double distanciaKm);

    // Getters
    int getNumeroVuelo() const;
    string getOrigen() const;
    string getDestino() const;
    string getFecha() const;
    int getDuracion() const;
    double getPrecio() const;
    int getAsientosDisponibles() const;
    double getDistanciaKm() const;

    // Setters
    void setNumeroVuelo(int numeroVuelo);
    void setOrigen(string origen);
    void setDestino(string destino);
    void setFecha(string fecha);
    void setDuracion(int duracion);
    void setPrecio(double precio);
    void setAsientosDisponibles(int asientos);
    void setDistanciaKm(double distancia);

    // Métodos específicos
    double agregarPrecio() const;
    bool hayAsientosDisponibles() const;
    void reservarAsiento();
    void mostrarInformacion() const;
};
```

```cpp
// Constructor por defecto
Vuelo::Vuelo() {
    numeroVuelo = 0;
    origen = "";
    destino = "";
    fecha = "";
    duracion = 0;
    precio = 0.0;
    asientosDisponibles = 0;
    distanciaKm = 0.0;
}

// Constructor con parámetros
Vuelo::Vuelo(int numeroVuelo, string origen, string destino, string fecha,
             int duracion, double precio, int asientosDisponibles, double distanciaKm) {
    this->numeroVuelo = numeroVuelo;
    this->origen = origen;
    this->destino = destino;
    this->fecha = fecha;
    this->duracion = duracion;
    this->precio = precio;
    this->asientosDisponibles = asientosDisponibles;
    this->distanciaKm = distanciaKm;
}
```

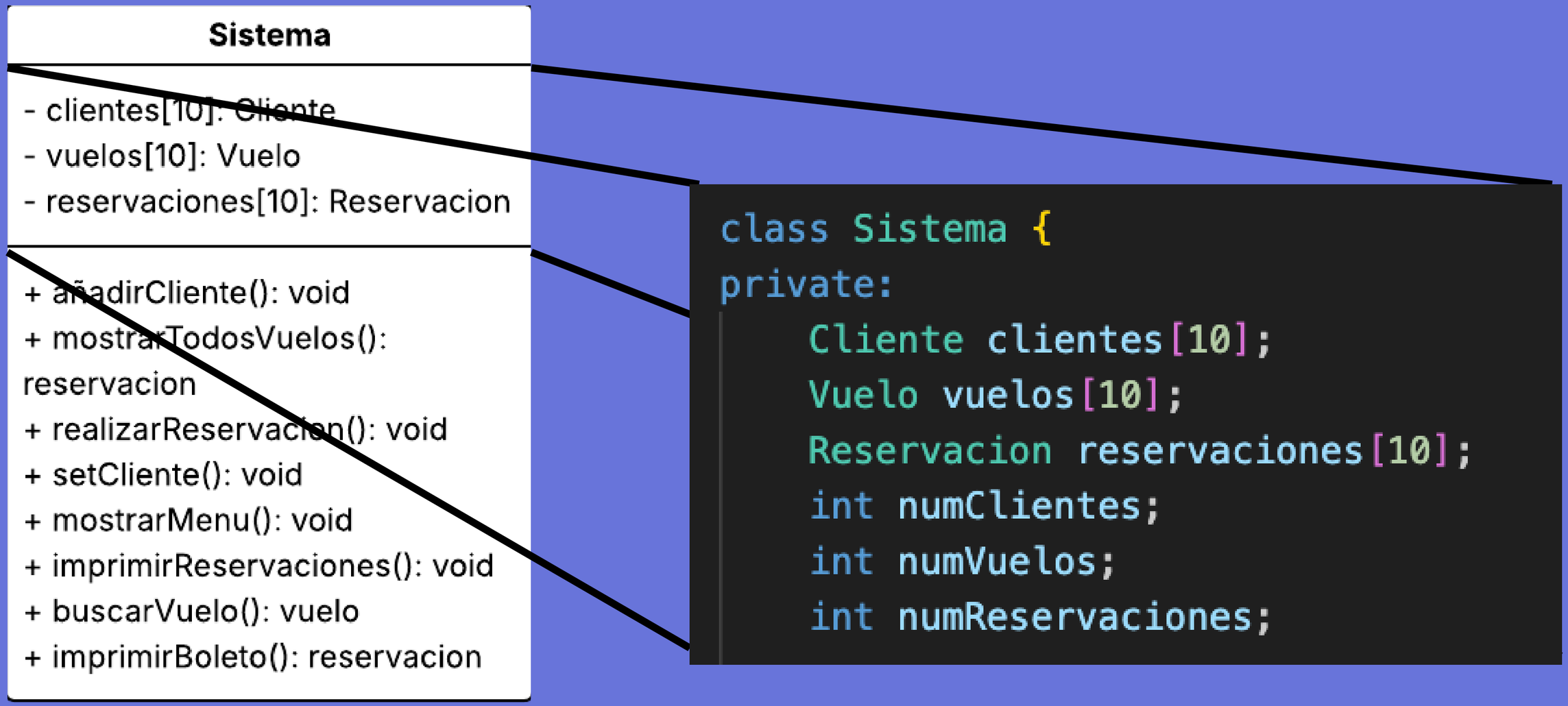
```cpp
// Métodos específicos
double Vuelo::agregarPrecio() const {
    return precio;
}

bool Vuelo::hayAsientosDisponibles() const {
    return asientosDisponibles > 0;
}

void Vuelo::reservarAsiento() {
    if (hayAsientosDisponibles()) {
        asientosDisponibles--;
        cout << "Asiento reservado exitosamente. Asientos restantes: "
             << asientosDisponibles << endl;
    } else {
        cout << "No hay asientos disponibles para este vuelo." << endl;
    }
}

void Vuelo::mostrarInformacion() const {
    cout << "\n--- Vuelo #" << numeroVuelo << " ---" << endl;
    cout << "Origen: " << origen << " -> Destino: " << destino << endl;
    cout << "Fecha: " << fecha << endl;
    cout << "Duracion: " << duracion << " horas" << endl;
    cout << "Precio: $" << precio << endl;
    cout << "Distancia: " << distanciaKm << " km" << endl;
    cout << "Asientos disponibles: " << asientosDisponibles << endl;
    cout << "------------------------\n" << endl;
}
```

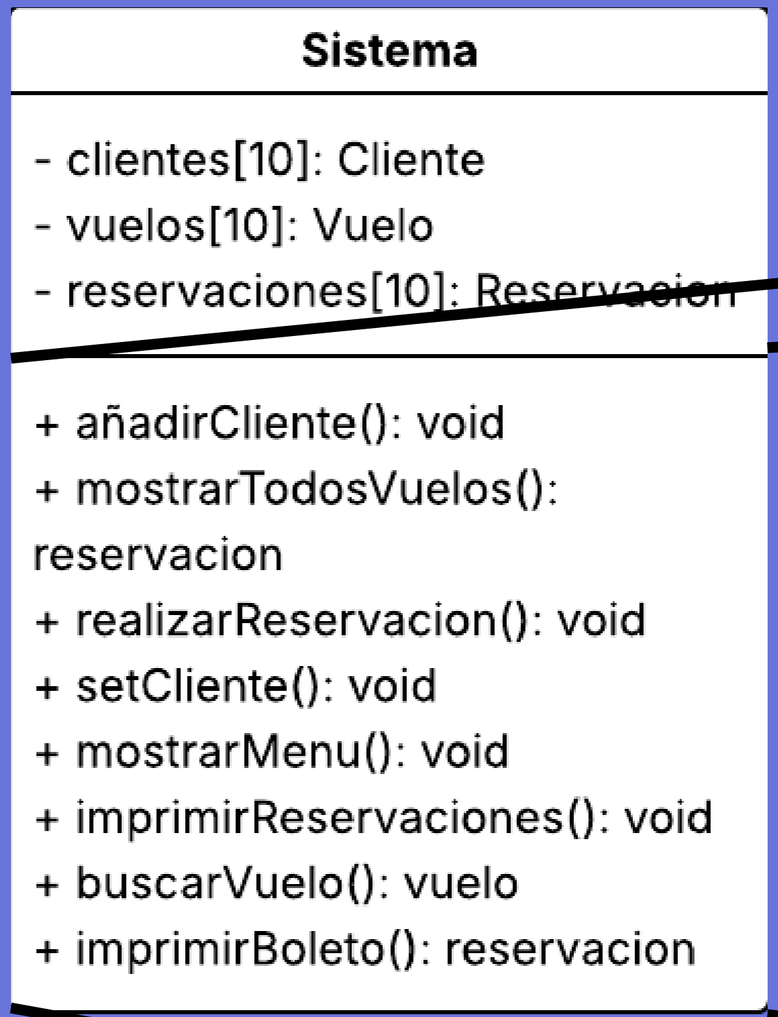

Cliente
- id: int
- nombre: string
- email: string
- kilometrosAcumulados: double
+ getId(): int
+ getNombre(): string
+ acumularKilometros(double): void
+ tieneDescuento(): bool
+ mostrarInformacion(): void
class Cliente {
private:
int id;
string nombre;
string email;
double kilometrosAcumulados;


Cliente
- id: int
- nombre: string
- email: string
- kilometrosAcumulados: double
+ getId(): int
+ getNombre(): string
+ acumularKilometros(double): void
+ tieneDescuento(): bool
+ mostrarInformacion(): void
public:
// Constructor
Cliente();
Cliente(int id, string nombre, string email, double kilometrosAcumulados = 0.0);
// Getters
int getId() const;
string getNombre() const;
string getEmail() const;
double getKilometrosAcumulados() const;
// Setters
void setId(int id);
void setNombre(string nombre);
void setEmail(string email);
void setKilometrosAcumulados(double kilometros);
// Métodos específicos
void acumularKilometros(double kilometros);
bool tieneDescuento() const;
void mostrarInformacion() const;
};

```cpp
#include "Cliente.h"
#include <iostream>
using namespace std;

// Constructor por defecto
Cliente::Cliente() {
    id = 0;
    nombre = "";
    email = "";
    kilometrosAcumulados = 0.0;
}

// Constructor con parámetros
Cliente::Cliente(int id, string nombre, string email, double kilome
    this->id = id;
    this->nombre = nombre;
    this->email = email;
    this->kilometrosAcumulados = kilometrosAcumulados;
}
```

```cpp
// Métodos específicos
void Cliente::acumularKilometros(double kilometros) {
    kilometrosAcumulados += kilometros;
    cout << "Se han acumulado " << kilometros << " kilometros. Total: "
         << kilometrosAcumulados << " km" << endl;
}

bool Cliente::tieneDescuento() const {
    return kilometrosAcumulados >= 50000.0;
}

void Cliente::mostrarInformacion() const {
    cout << "\n=== INFORMACION DEL CLIENTE ===" << endl;
    cout << "ID: " << id << endl;
    cout << "Nombre: " << nombre << endl;
    cout << "Email: " << email << endl;
    cout << "Kilometros Acumulados: " << kilometrosAcumulados << " km" << endl;
    if (tieneDescuento()) {
        cout << "CLIENTE FRECUENTE: Tiene 40% de descuento" << endl;
    }
    cout << "================================\n" << endl;
}
```


Sistema
- clientes[10]: Cliente
- vuelos[10]: Vuelo
- reservaciones[10]: Reservacion
+ añadirCliente(): void
+ mostrarTodosVuelos():
reservacion
+ realizarReservacion(): void
+ setCliente(): void
+ mostrarMenu(): void
+ imprimirReservaciones(): void
+ buscarVuelo(): vuelo
+ imprimirBoleto(): reservacion
class Sistema {
private:
Cliente clientes[10];
Vuelo vuelos[10];
Reservacion reservaciones[10];
int numClientes;
int numVuelos;
int numReservaciones;



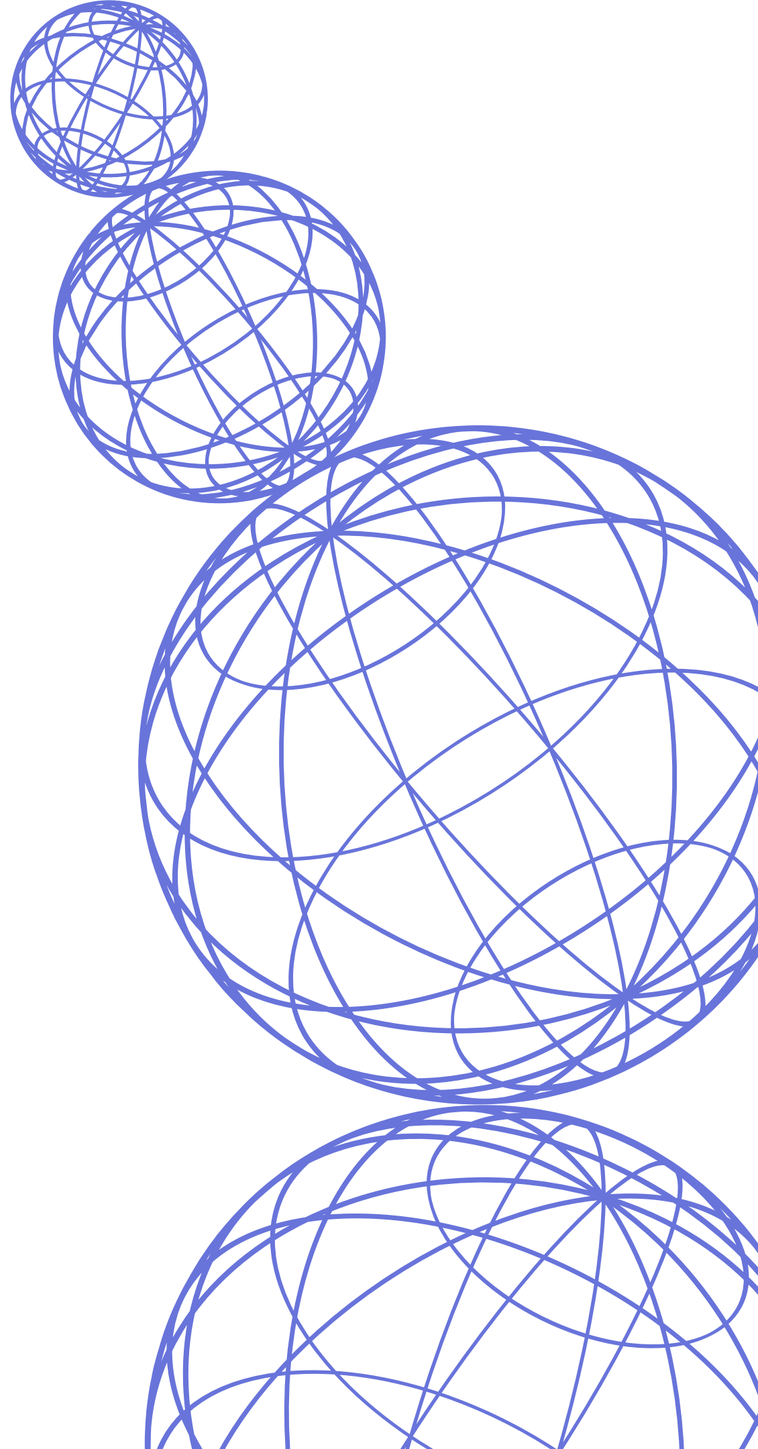

```
public:
    // Constructor
    Sistema();

    // Métodos de gestión
    void anadirCliente();
    void mostrarTodosVuelos();
    void realizarReservacion();
    void setCliente();
    void mostrarMenu();
    void imprimirReservaciones();
    Vuelo buscarVuelo(string origen, string destino, string fecha);
    void imprimirBoleto(int idReservacion);

    // Métodos auxiliares
    void inicializarDatos();
    Cliente* buscarClientePorId(int id);
    Vuelo* buscarVueloPorNumero(int numeroVuelo);
    bool iniciarSesionCliente(int idCliente);
};
```

[Mostrar este en la muestra]

(netlifly)

Muchas gracias

Ahora el código



Fuentes

- Project overview | bucolic-beijinho-5b673b | Netlify. (n.d.). https://app.netlify.com/projects/bucolic-beijinho-5b673b/overview
- Vuelos ViajandoAndo. (n.d.). https://bucolic-beijinho-5b673b.netlify.app/
- https://chatgpt.com/
- https://claude.ai.com/