# **Formulario: Vista Lógica del Proyecto de Software**

**Proyecto:** Reserva BHAS  
**Grupo:**06  
**Fecha:** 22/10/2024

**1. Descripción General del Sistema**

**Descripción del sistema:**

El sistema de reserva del salón "Agenda Salón BHAS" es una plataforma web diseñada para gestionar la reserva de citas en un salón de belleza. Permite a los clientes seleccionar estilistas, elegir servicios, verificar la disponibilidad de horarios y hacer reservas. El sistema incluye módulos para la gestión de citas, estilistas, servicios, reseñas y un panel administrativo que facilita el control sobre el funcionamiento del salón. La plataforma cuenta con diferentes interfaces para administradores, estilistas y clientes, cada uno con funcionalidades específicas, como la gestión de disponibilidad para los estilistas y la revisión de citas para los administradores.

Los principales componentes lógicos del sistema son:

* Gestión de usuarios: Clientes, administradores y estilistas.
* Sistema de reservas: Control de citas y disponibilidad.
* Sistema de reseñas: Permite a los clientes dejar comentarios y calificaciones sobre el servicio.
* Notificaciones por correo: Envío de confirmaciones y recordatorios.
* Panel administrativo: Funciones avanzadas para la gestión de servicios, estilistas y citas.

**2. Principales Clases y Entidades del Sistema**

Describe las clases o entidades principales del sistema, sus atributos y métodos principales, y su propósito dentro del sistema.

**Clase 1 (Cliente):   
Atributos:**

* **id:** String - ID único del cliente (compuesto por iniciales del nombre y un contador).
* **nombre:** String - Nombre completo del cliente.
* **email:** String - Correo electrónico del cliente.
* **telefono:** String - Número de teléfono.
* **direccion:** String - Dirección del cliente.
* **comentarios:** String - Comentarios adicionales.
* **created\_at:** Date - Fecha de registro.
* **updated\_at:** Date - Fecha de la última actualización.

**Métodos:**

* **registrar\_cliente() -** Registra un nuevo cliente.
* **actualizar\_perfil() -** Actualiza los datos del perfil del cliente.
* **ver\_citas() -** Muestra la lista de citas del cliente.
* **iniciar\_sesion() -** Autentica al cliente en el sistema.

**Propósito:** La clase Cliente representa a los usuarios del sistema que realizan reservas en el salón BHAS. Su principal objetivo es gestionar la información personal de los clientes, como su nombre, correo, teléfono y dirección, permitiendo mantener un registro detallado de cada cliente y sus interacciones con el salón.

**Clase 2 (Estilista):   
Atributos:**

* **id:** String - ID único del estilista.
* **nombre:** String - Nombre del estilista.
* **foto\_url:** String - URL de la imagen del estilista.
* **telefono:** String - Número de teléfono del estilista.
* **correo:** String - Correo electrónico del estilista.
* **activo:** Boolean - Indica si el estilista está activo.
* **disponibilidad:** List[Disponibilidad] - Lista de objetos de disponibilidad.
* **created\_at:** Date - Fecha de creación del registro.
* **updated\_at:** Date - Fecha de la última actualización.

**Métodos:**

* **ver\_disponibilidad() -** Devuelve las horas disponibles del estilista.
* **agregar\_disponibilidad(fecha: Date, hora\_inicio: Time, hora\_fin: Time) -** Añade nuevas horas de disponibilidad.
* **modificar\_disponibilidad(disponibilidad: Disponibilidad) -** Modifica la disponibilidad existente.
* **ver\_citas() -** Muestra las citas del estilista.

**Propósito:** Representa a los estilistas del salón y gestiona su disponibilidad de horarios.

**Clase 3 (Cita):   
Atributos:**

* **id:** String - ID único de la cita.
* **cliente\_id:** String - ID del cliente que reserva la cita.
* **estilista\_id:** String - ID del estilista que atiende la cita.
* **fecha:** Date - Fecha de la cita.
* **hora:** Time - Hora de la cita.
* **servicio:** String - Servicio seleccionado (ejemplo: corte, color).
* **estado:** String - Estado de la cita (pendiente, confirmada, completada, cancelada).
* **comentarios:** String - Comentarios adicionales sobre la cita.

**Métodos:**

* **crear\_cita(cliente: Cliente, estilista: Estilista, fecha: Date, hora: Time, servicio: String) -** Crea una nueva cita.
* **cancelar\_cita() -** Cancela la cita.
* **confirmar\_cita() -** Confirma la cita.
* **enviar\_recordatorio() -** Envía un recordatorio antes de la cita.

**Propósito:** Gestiona el proceso de reserva de citas, su creación y cancelación.

**Clase 4 (Disponibilidad):   
Atributos: id: String - ID único de la disponibilidad.**

* **estilista\_id:** String - ID del estilista.
* **fecha:** Date - Fecha en la que el estilista está disponible.
* **hora\_inicio:** Time - Hora de inicio de la disponibilidad.
* **hora\_fin:** Time - Hora de fin de la disponibilidad.

**Métodos:**

* **ver\_disponibilidad(estilista\_id: String, fecha: Date) -** Devuelve las horas disponibles de un estilista en una fecha específica.
* **agregar\_hora\_disponible(hora: Time) -** Añade una hora específica a la disponibilidad.
* **eliminar\_hora\_disponible(hora: Time) -** Elimina una hora de la disponibilidad.

**Propósito:** Gestiona los horarios en los que los estilistas del salón están disponibles para atender a los clientes. Esta clase permite visualizar, agregar y eliminar horas disponibles para un estilista en una fecha determinada, facilitando la organización y administración de las citas

**Clase 5 (Administrador):   
Atributos:**

* **id:** String - ID único del administrador.
* **idestilista:** String - ID del estilista (si aplica).
* **pass:** String - Contraseña de acceso.
* **rol:** String - Rol del administrador (puede ser 'admin' o 'estilista').

**Métodos:**

* **gestionar\_citas() -** Permite al administrador gestionar las citas (ver, modificar, eliminar).
* **gestionar\_disponibilidad() -** Permite gestionar la disponibilidad de los estilistas.
* **gestionar\_usuarios() -** Gestiona la información de los usuarios (clientes y estilistas).
* **asignar\_permisos(permiso: String) -** Asigna permisos a los usuarios según su rol.

**Propósito:** La clase Administrador tiene como objetivo centralizar la gestión y el control del sistema de reserva del salón. Este administrador puede tener acceso total al sistema (como un administrador del salón) o un acceso más limitado (como un estilista con permisos restringidos), según su rol.

**Clase 6 (Email):   
Atributos:**

* **destinatario:** String - Dirección de correo electrónico del destinatario.
* **asunto:** String - Asunto del correo.
* **mensaje:** String - Contenido del mensaje.

**Métodos:**

* **enviar\_confirmacion\_cita(cliente: Cliente, cita: Cita) -** Envía un correo de confirmación de cita.
* **enviar\_recordatorio\_cita(cliente: Cliente, cita: Cita) -** Envía un recordatorio de la cita.
* **enviar\_bienvenida(cliente: Cliente) -** Envía un correo de bienvenida tras el registro del cliente.

**Propósito:** La clase *Email* se encarga de gestionar el envío de correos electrónicos automatizados a los clientes del sistema. Esta clase es responsable de facilitar la comunicación entre el sistema y los clientes, enviando correos de confirmación, recordatorios de citas y mensajes de bienvenida.

**3. Relaciones entre clases**

Especifica cómo están relacionadas las clases entre sí, indicando relaciones de herencia, composición, agregación o dependencias.

**Relaciones entre clases:**

* Cliente y Cita: Un cliente puede tener múltiples citas (relación 1 a N).
* Estilista y Cita: Un estilista puede atender múltiples citas (relación 1 a N).
* Estilista y Disponibilidad: Un estilista tiene varias disponibilidades (relación 1 a N).
* Administrador y Permiso: Un administrador puede tener varios permisos asignados (relación 1 a N).
* Administrador y Estilista: Un administrador puede gestionar a varios estilistas (relación 1 a N).

**4. Interfaces y dependencias**

**Descripción de interfaces y dependencias:**

**1. Clase: Cliente**

* Interfaz: IClienteRepositorio: Esta interfaz define los métodos necesarios para interactuar con la base de datos en lo que respecta a los clientes. Facilita la abstracción de la lógica de acceso a datos, permitiendo que la clase Cliente interactúe con el módulo de base de datos a través de esta interfaz.
* Métodos de la interfaz:
  + obtener\_cliente\_por\_id(id: String): Cliente
  + guardar\_cliente(cliente: Cliente): void
  + actualizar\_cliente(cliente: Cliente): void
* Interacción con otros módulos: La clase Cliente interactúa con el módulo de base de datos a través de la interfaz IClienteRepositorio, lo que permite recuperar y almacenar información de clientes sin depender directamente de una implementación específica de la base de datos. Esto es útil para la inversión de dependencias, facilitando cambios en la infraestructura (por ejemplo, cambiar de MySQL a MongoDB) sin modificar la lógica del sistema.

**2. Clase: Estilista**

* Interfaz: IEstilistaRepositorio: Similar a la clase Cliente, esta interfaz define los métodos para interactuar con la base de datos en relación con los estilistas. La interfaz permite la abstracción del almacenamiento y recuperación de los estilistas.
* Métodos de la interfaz:
  + obtener\_estilista\_por\_id(id: String): Estilista
  + guardar\_estilista(estilista: Estilista): void
  + actualizar\_disponibilidad(estilista: Estilista, disponibilidad: Disponibilidad): void
* Interacción con otros módulos: La clase Estilista se comunica con el módulo de base de datos a través de la interfaz IEstilistaRepositorio para recuperar y almacenar información relacionada con los estilistas, manteniendo la lógica desacoplada de la implementación específica de la base de datos.

**3. Clase: Cita**

* Interfaz: ICitaRepositorio: Esta interfaz maneja el acceso a los datos de las citas. Similar a las interfaces de Cliente y Estilista, permite la abstracción en el acceso a datos para las citas.
* Métodos de la interfaz:
  + obtener\_cita\_por\_id(id: String): Cita
  + guardar\_cita(cita: Cita): void
  + actualizar\_cita(cita: Cita): void
  + cancelar\_cita(cita\_id: String): void
* Interacción con otros módulos: La clase Cita interactúa con el módulo de base de datos a través de la interfaz ICitaRepositorio, lo que facilita la gestión de citas sin acoplar la lógica a la base de datos específica.

**4. Clase: Email**

* Interfaz: IEmailService: Esta interfaz define los métodos necesarios para el envío de correos electrónicos. Asegura que diferentes servicios de correo (Gmail, SendGrid, etc.) puedan implementarse de manera intercambiable.
* Métodos de la interfaz:
  + enviar\_correo(destinatario: String, asunto: String, mensaje: String): void
  + enviar\_confirmacion\_cita(cliente: Cliente, cita: Cita): void
  + enviar\_recordatorio\_cita(cliente: Cliente, cita: Cita): void
* Interacción con otros módulos: La clase Email se comunica con el módulo de envío de correos a través de la interfaz IEmailService, lo que facilita el envío de correos electrónicos sin acoplar el sistema a una implementación específica. Esto permite, por ejemplo, cambiar el proveedor de correo electrónico sin modificar la lógica del sistema.

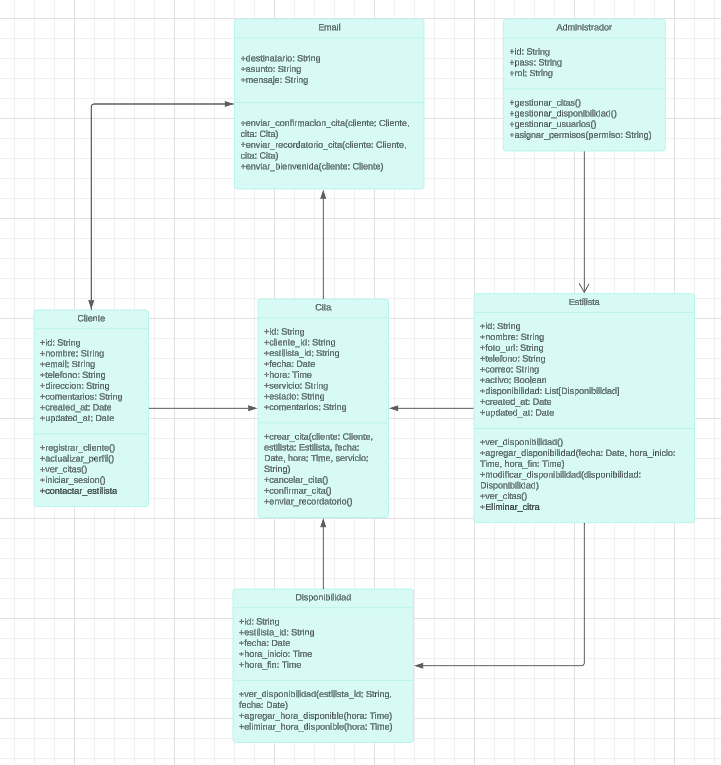
**5. Clase: Administrador**

* Interfaz: IAdministradorRepositorio: Esta interfaz gestiona el acceso a datos para los administradores y permite que la lógica de negocio no dependa directamente de la base de datos.
* Métodos de la interfaz:
  + obtener\_admin\_por\_id(id: String): Administrador
  + guardar\_administrador(admin: Administrador): void
  + asignar\_permiso(admin\_id: String, permiso: String): void
* Interacción con otros módulos: La clase Administrador interactúa con el módulo de base de datos a través de la interfaz IAdministradorRepositorio, lo que permite realizar la gestión de administradores y permisos

**6. Clase: Disponibilidad**

* Interfaz: IDisponibilidadRepositorio: Define la interacción para gestionar la disponibilidad de los estilistas.
* Métodos de la interfaz:
  + obtener\_disponibilidad\_por\_estilista(estilista\_id: String, fecha: Date): List[Disponibilidad]
  + guardar\_disponibilidad(disponibilidad: Disponibilidad): void
* Interacción con otros módulos: La clase Disponibilidad utiliza la interfaz IDisponibilidadRepositorio para interactuar con el módulo de base de datos, lo que facilita la gestión de los horarios disponibles para los estilistas sin depender de la implementación específica de almacenamiento.

**5. Diagrama de Clases del Sistema**

**Diagrama de clases:****