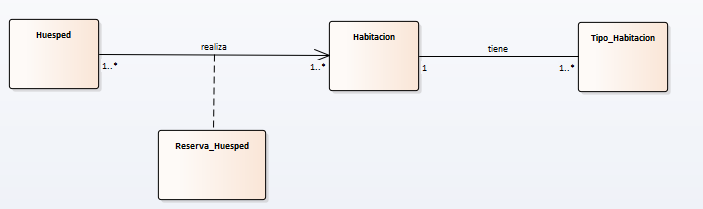
1. **HOTEL IDEAL**

**Relaciones**

GRUPO 1

* El *Hotel* tiene un conjunto fijo *de* **Habitaciones** *numeradas del 1 al 200 en donde los huéspedes pueden alojarse. Se encuentran en un piso, y tienen un estado Ocupada (1) o Libre (0).*
* Una habitación es de una categoría o **Tipo de Habitación** específico. A este *Tipo de Habitación* *(Estándar simple, doble, Triple, o Suite Lujo) pertenecen muchas habitaciones*.
* Un **huésped** hace muchas **reservas**, y cada reserva corresponde a una habitación. Asi una reserva vincula un huésped con una habitación, y tiene fechas inicio y fin, días, un monto a pagar, un estado.

**Clases**

* Los Huéspedes son quienes podrán realizar sus reservas según disponibilidad en las fechas que correspondan. **ABM *Huésped***tiene sus datos personales (nombre, DNI, Domicilio, correo, celular).
* *Crear la* ***Clase Tipo de Habitación*** tendrá sus propias características, las relevantes para nuestro sistema de gestión de reservas son: código, *cantidad de personas* (maxima), cantidad de camas, tipo de camas (Simples, Queen, King Size), precio por noche, que por simplicidad se supondrá único (sin promociones o tarifas distintas para temporada alta o baja).
* Un método cambiarPrecio, busca un Tipo de Habitación y actualiza su precio.
* **ABM de Habitaciones**: Se deben poder gestionar el alta baja y modificación de Habitaciones de un tipo. Ejemplo. Si antes tenía 30 de tipo estándar, luego de agregar las habitaciones 201 al 205 de ese tipo, el conteo sumará 35 en total.
* **ABM Reserva**. En cada *Reserva* se toma nota de: *datos del Huesped,* *Tipo de Habitación, cantidad de personas*, fechas de entrada y salida, importe total de estadía. Metodos Buscar una Reserva por Huesped, Cancelar reserva

**Calcular monto estadía**: en la reserva, se calcula como precio por noche (de Tipo Habitación) por cantidad de días de estadía ingresado. Devuelve el monto\*.

1. Se construye la reserva con fechaCheckin, fechaCheckout, Huesped, Habitacion, monto\*, estado=1.
2. La Habitación se marca Ocupada(1), en la fecha de salida vuelve a su estado Libre.

# 💼 Gestión de Alcance

## 📈 Alcance del Producto

El producto a desarrollar es un sistema de gestión hotelera tipo intranet que permitirá a los administradores del hotel gestionar habitaciones, tipos de habitaciones, huéspedes, reservas, y usuarios del sistema. El sistema incluirá funcionalidades para realizar altas, bajas, modificaciones, y consultas (ABMC) de habitaciones, tipos de habitaciones, huéspedes, reservas, y usuarios. Además, se desarrollará una conexión segura con la base de datos para asegurar la persistencia y seguridad de la información.

## 📈 Alcance del Proyecto

## 🎯 Objetivos Generales

El proyecto se dividirá en tres versiones, cada uno enfocado en diferentes módulos del sistema, con el objetivo de proporcionar una herramienta interna eficiente para la gestión del hotel.

El objetivo principal será desarrollar un sistema de gestión hotelera del tipo intranet que permita a los administradores realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) de habitaciones, huéspedes, reservas, y usuarios, además de mantener la información actualizada y disponible .

## 🎯 Objetivos Específicos

* Implementar un sistema de ABMC para los huéspedes, habitaciones, tipos de habitaciones y reservas.
* Permitir la creación y cancelación de reservas, vinculando a los huéspedes con las habitaciones según disponibilidad por fecha de reserva.
* Calcular el monto total de la estadía basado en las fechas de entrada y salida, el tipo de habitación seleccionado y la cantidad de huéspedes por habitación.
* Desarrollar un módulo para la gestión de huéspedes, incluyendo la capacidad de registrar, modificar y eliminar información de los huéspedes.
* Crear un módulo de usuarios para la administración de credenciales de acceso al sistema, roles y permisos.
* Administrar el estado de las habitaciones (ocupada o libre) automáticamente en función de las reservas.
* Incorporar un método para actualizar los precios de las habitaciones por tipo.
* Desarrollar una interfaz de login amigable que incluye funcionalidad para recuperación de contraseñas, asegurando el control de acceso al sistema.

## 📌 Supuestos

* **Equipo de Trabajo:** Se cuenta con un equipo de analistas, desarrolladores, diseñadores y testers capacitados para el desarrollo de aplicaciones de escritorio.
* **Entrevistas con el Cliente:** Ya se realizaron reuniones preliminares con el cliente para capturar los requisitos iniciales y definir las funcionalidades clave.
* **Punto de Partida:** El desarrollo partirá desde cero, no se cuenta con un sistema previo.
* **Lenguaje de Programación y Tecnologías:** La aplicación será desarrollada en Java utilizando Java Swing para la interfaz de usuario y MySQL para la gestión de la base de datos.
* **Conectividad:** el servicio de internet está disponible y es estable en el entorno de producción.
* **Usuarios:** tendrán conocimientos básicos de informática y gestión hotelera, el sistema será utilizado principalmente por personal administrativo del hotel.
* **Instalación:** se realizará la instalación del programa por única vez en los equipos conectados a la intranet del hotel.

## ⚠️ Restricciones

### 🕒 Tiempo:

* + El proyecto será desarrollado siguiendo el modelo evolutivo en un total de 3 incrementos.
  + Se dispone de un total de 6 meses, con 2 meses asignados a cada incremento.

### 💵 Presupuesto:

* + El presupuesto es de cincuenta mil dólares y se asignará de manera equitativa entre los incrementos.El presupuesto asignado para el proyecto no debe exceder los recursos asignados.

## 🚧 Límites

* **Actualizaciones:** Las actualizaciones y mejoras del sistema posterior a la entrega del proyecto no están incluidas y se deberán contratar aparte.
* **Manual de Usuario:** El manual de usuario básico no estará incluido.
* **Capacitación del Personal:** No se incluye la capacitación del personal del hotel en el uso de la aplicación. Esta capacitación deberá ser organizada y financiada por el cliente.
* **Mantenimiento:** Se incluye un mantenimiento por un periodo de tres meses. Cualquier mantenimiento adicional o soporte técnico fuera de este periodo deberá ser contratado adicionalmente.
* **Hosting de la Base de Datos:** No se incluye el hosting de la base de datos.
* **Intranet**:La aplicación estará disponible solo para uso interno a través de la intranet del hotel; no se desarrollarán aplicaciones móviles ni acceso desde fuera de la red interna del hotel.

🧩 CICLO DE VIDA ITERATIVO

# 📈 Primer Versión

Habitación - Tipo de Habitación- Huésped - Conexión - Reserva

## 🎯 Captura de requisitos

* Se realizan las clases modelo y las clase main.
* Diagrama de clases Habitacion, Tipo Habitacion, Huesped, conexión y Reserva.
* Diagrama de bases de datos:Habitacion, Tipo Habitacion, Huesped y Reserva, Usuario.

## 💻 Desarrollo:

### 📊 Análisis: *(diagrama de clases)*

* + **Habitación**: id, número, piso, estado.
  + **tipo Habitación**: id, tipo, descripción, precioXNoche, cantCama, cantPersonas, tipoCama.
  + **Huésped** : Id, nombre, apellido, DNI, domicilio, correo, celular.
  + **Reserva**: id, Habitacion habitacion, Huesped huesped, fechaCheckin, fechaCheckout, montoApagar, estado.
  + **Usuario:** id, name, password, tipo.

### 🎨 Diseño:

#### 📐 Diagrama de clases detallado:

* + - **Usuario**: constructores, métodos y UML.
    - **Habitación:** constructores, métodos y UML.
    - **Tipo de habitación**: constructores, métodos y UML.
    - **Huésped**: constructores, métodos y UML.
    - **Reserva**: constructores, métodos y UML.
  + 🛢️**Diagrama de Base de Datos:**
    - **Huésped :** id(PK), nombre, apellido, dni, domicilio, correo, celular.
    - **tipoHabitación**: id(PK), tipo, descripción, precioXNoche, cantCama, cantPersonas, tipoCama.
    - **Habitación**: id(PK), tipoHabitacionId(FK), número, piso, estado.
    - **Reserva:** id(PK), huesped\_id(FK), habitacion\_id(FK), fechaCheckin, fechaCheckout, montoApagar, estado
    - **Usuario:** id(PK), nombre(UNQ), contraseña, tipo.

### 🔧 Implementación:

#### 👲🏻 Huésped:

* + - **Constructores**: vacío, con id
    - **Métodos**: get(), set()

#### 🏨 Habitación:

* + - **Constructores**: vacío, con id
    - **Métodos**: get(), set()

#### 📋 Tipo Habitación:

* + - **Constructores**: vacío, con id
    - **Métodos**: get(), set()

#### 🛎️ Reserva:

* + **Constructores**: vacío, con id
  + **Métodos**: get(), set()

#### 🌐 Conexión:

* + **Parámetros:** url, host, password, usuario
  + **Métodos**: getConnection()

### 🔍 Testing:

#### 📝 Testing en Caja Blanca (Debug):

* + - Probar conexión
    - Probar las clases desde el main
    - Consultas de la base de datos

#### 🧪 Testing en Caja Negra (Validación):

* + - Pruebas básicas desde el main.

## 🚀 Entrega de la primer versión:

### 📋 Entregado:

* + Diagrama de clases Habitacion, Tipo Habitacion, Huesped, Conexión y Reserva.
  + Esquema de base de datos de las entidades.

### 📋 Falta entregar:

* + Clases Data
  + Vistas

# 📈 Segunda Versión

HabitaciónData - TipodeHabitaciónData - HuéspedData**-**ReservaData - UsuarioData

## 🎯 Captura de requisitos

* El admin gestiona las clases data.
* Se crean las clases DATA para el manejo de la base de datos
* ABMC habitación, tipoHabitacion, usuario, huésped, reserva.

## 💻 Desarrollo:

### 📊 Análisis: (atributos)

* + **Habitación Data**: Habitacion habitacion
  + **tipo Habitación Data**: TipoHabitacion tipoHabitacion
  + **Huesped Data:** Huesped huesped.
  + **Reserva Data**: Reserva reserva
  + **Usuario Data:** Usuario usuario

### 🎨 Diseño:

#### 📐 Diagrama de clases detallado:

* + - **Habitación Data**: ABMC de habitación.
    - **Tipo de habitación Data**: ABMC de tipo de habitación.
    - **Huésped Data:**Alta de Huésped, Baja de Huésped, Modificación de Huésped y Consulta de los huéspedes.
    - **Reserva Data:** Alta de Reserva, Baja de Reserva, Modificación de Reserva,Consulta de reservas.
    - **Usuario Data:**Alta de Usuario, Baja de Usuario, Modificación de Usuario,Consulta de Usuario.

### 🔧 Implementación:

#### 🛠️ Habitación Data:

* + - **Métodos**:
      * void altaHabitación(Habitacion habitacion){}
      * void bajaHabitación(int id){}
      * void modificarHabitacion(Habitacion habitacion){}
      * Habitacion consultarHabitacion(int id){}
      * ArrayList<Habitacion> listarHabitaciones(){}
      * Validaciones: excepción (try/catch)

#### 🛠️ Tipo Habitación Data:

* + - **Métodos:**
      * void altaTipoHabitacion(TipoHabitacion tipoHabitacion){}
      * void bajaTipoHabitacion(int id){}
      * void modificarTipoHabitacion(TipoHabitacion tipoHabitacion){}
      * TipoHabitacion consultarTipoHabitacion(int id){}
      * ArrayList <TipoHabitacion> listarTipoHabitacion(){}
      * void cambiarPrecio(String tipo){}
      * Validaciones: excepción (try/catch)

#### 🛠️ Huésped Data:

* + - **Métodos:**
      * void altaHuesped(Huesped huesped){}
      * void bajaHuesped(int id){}
      * void modificarHuesped(Huesped huesped){}
      * Huesped consultaHuesped(int id){}
      * validaciones
      * excepción *(try/catch)*

#### 🛠️ Reserva Data:

* + - **Método/Eventos:**
      * void altaReserva(Reserva reserva){}
      * void bajaReserva(int id){}
      * void modificarReserva(Reserva reserva){}
      * Reserva consultaReserva(int id){}
      * void calcularMontoAPagar(int tipoHabitacion, int cantidadPersonas){}
      * Reserva buscarReservaPorHuesped(int idHuesped){}
      * void cambiarEstadoReserva(int idReserva, boolean estado){}
      * Validaciones
      * Excepciones *(try/catch)*

#### 🛠️ Usuario Data:

* + - **Método/Eventos:**
      * void altaUsuario(Usuario usuario){}
      * void bajaUsuario(int id){}
      * void modificarUsuario(Usuario usuario){}
      * Usuario consultarUsuario(int id){}
      * void autenticacionUsuario(Usuario usuario){}{}
      * void recuperarConstraseña(String email){}
      * Validaciones
      * Excepciones *(try/catch)*

### 🔍 Testing:

#### 📝 Testing en Caja Blanca (Debug):

* + - Prueba de consultas de ABMC

#### 🧪 Testing en Caja Negra (Validación):

* + - Realizar pruebas de entradas.

#### 🔗Testing integrado: versión anterior + versión actual.

## 

## 🚀 Entrega de la segunda versión:

### 📋 Entregado:

* + Clases Data

### 📋 Falta entregar:

* + Vistas

# 📈 Tercer Versión

VISTAS: Habitación - Tipo de Habitación - Huésped - Reserva

## 🎯 Captura de requisitos

* Se crean las vistas de huesped, habitacion, tipo de habitacion, huesped, reserva, usuario, login y menú principal.
* Se implementarán los métodos de cada vista.
* Todas las funcionalidades restantes para la gestión de todas las entidades anteriormente nombradas, fueron implementadas y validadas en la primera y segunda iteración.

## 💻 Desarrollo:

### 📊 Análisis: (diagrama de clases):

* + Se completo en la iteración anterior

### 🎨 Diseño:

#### 📄 Vista (estático):

* + - Reserva
    - Login
    - Menú principal
    - Usuario
    - Formulario Habitación.
    - Formulario tipoHabitación.

### 🔧 IMPLEMENTACIÓN:

#### 👁️ Vista Habitacion:

* + - **Métodos/Eventos:**
      * void consultarHabitacion(){}
      * void guardarHabitacion(){}
      * void modificarHabitacion(){}
      * void eliminarHabitacion(){}
      * void limpiarFormulario(){}
      * void listarHabitaciones(){}
      * void salir(){}
      * Validaciones: Excepciones (try/catch)

#### 👁️ Vista tipo Habitación:

* + - **Métodos/Eventos:**
      * void consultarTipoHabitacion(){}
      * void guardarTipoHabitacion(){}
      * void modificarTipoHabitacion(){}
      * void eliminarTipoHabitacion(){}
      * void limpiarFormulario(){}
      * void listarTipoHabitacion(){}
      * void salir(){}
      * Validaciones: Excepciones (try/catch)

#### 👁️ Vista Reserva:

* + - **Método/Eventos:**
      * void listarReservas(){}
      * void buscarReservaPorHuesped(){}
      * void guardarReservas(){}
      * void modificarReservas(){}
      * void eliminarReserva(){}
      * void limpiarFormulario(){}
      * void salir(){}
      * Validaciones
      * Excepciones *(try/catch)*

#### 👁️ Vista Huesped:

* + - **Metodos/Eventos:**
      * void consultarHuesped(){}
      * void guardarHuesped(){}
      * void modificarHuesped(){}
      * void eliminarHuesped(){}
      * void salir(){}
      * void limpiarFormulario(){}
      * Validaciones
      * Excepciones *(try/catch)*

#### 👁️ Vista Usuario:

* + - **Método/Eventos:**
      * void consultarUsuario(){}
      * void hacerUsuario(){}
      * void modificarUsuario(){}
      * void eliminarUsuario(){}
      * void limpiarFormulario(){}
      * void salir(){}
      * Validaciones
      * Excepciones *(try/catch).*

#### 🔐 Vista Login:

* + - Método/Eventos:
      * void ingresarUsuario(){}
      * void registrarUsuario(){}
      * void recuperarContraseña(){}
      * void salir(){}
      * Validaciones
      * Excepciones *(try/catch)*

#### 🖥️ Vista Menú Principal:

* + - **Método/Eventos:**
      * void vistaUsuario(){}
      * void vistaReserva(){}
      * void vistaHuesped(){}
      * void vistaHabitacion(){}
      * void vistaTipoHabitacion(){}

### 🔍 Testing:

#### 📝 Testing en Caja Blanca (Debug):

* + - Revisar el código línea por línea.
    - Verificar la lógica del código.

#### 🧪 Testing en Caja Negra (Validación):

* + - Probar la funcionalidad del software
    - Realizar pruebas de entradas.
    - Probar vistas.

#### 🔗Testing integrado: versión anterior + versión actual

## Instalación:

Se realiza la instalación del sistema en los equipos conectados al intranet del hotel.

## 🚀 Entrega de la segunda versión:

### 📋 Entregado:

* + Vistas
  + Finalización de proyecto

# 🔗LINK G-Drive:

[1 - EL GRAN HOTEL - Documentos de Google](https://docs.google.com/document/d/1OZv-PentEt2-d-yox40QkPXul6a1w_VcWpuvUUVs5ks/edit)

## 👥 GRUPO 6:

Ayala, Ariel

Kaminski, Nicolás

Martinez, Calipto

Ponce, Bárbara

Prez, Lucas

Román, Nahuel