ÍNDICE

1. Relevamiento, Análisis y Diseño del Sistema **2**
1.1. Relevamiento 2
1.2. Análisis 2
1.3. Diseño del Sistema 3
1.4. Objetivo General 4
2. DER (Diagrama entidad-relación) **4**
3. DFD (Diagrama de flujo de datos) **5**
3.1 Sistema en general 6
3.2 Módulos 7
3.2.1 Módulo 1 7
3.2.2 Módulo 2 8
3.2.3 Módulo 3 9
3.2.4 Módulo 4 9
3.2.5 Módulo 5 10
3.3 Interpretación global 11
4. Desarrollo de módulos **12**
4.1 Módulo 1: Consulta de disponibilidad 12
4.2 Módulo 2: Registro de reserva 16
4.3 Módulo 3: Confirmación de la reserva 20
4.4 Módulo 4: Preparación de la habitación 23
4.5 Módulo 5: Check-in y atención al huésped 27

1. Relevamiento, Análisis y Diseño del Sistema

1.1. Relevamiento

En el contexto de la **gestión hotelera moderna**, se identificó la necesidad de contar con un sistema informático que optimice los procesos vinculados al ciclo completo de la reserva de habitaciones.
El relevamiento permitió detectar los principales puntos críticos en la operación cotidiana del hotel, como la falta de trazabilidad entre reservas, demoras en la confirmación de pagos y ausencia de un control centralizado sobre la preparación y asignación de habitaciones.

De este estudio preliminar surgieron los **requerimientos funcionales esenciales**:

- Consultar la disponibilidad de habitaciones en función de fechas, tipo y capacidad, sin alterar los datos operativos.

- Registrar nuevas reservas asociadas a huéspedes, con validaciones sobre fechas, capacidad y políticas del establecimiento.

- Confirmar reservas mediante la validación de pagos o garantías de estadía.

- Gestionar la preparación de habitaciones asegurando el cumplimiento de los estándares de limpieza, mantenimiento y confort.

- Controlar el proceso de **check-in**, registrando la ocupación efectiva y habilitando la atención personalizada al huésped.

Asimismo, se identificó la necesidad de establecer **registros históricos** que permitan auditar cada operación, garantizando la transparencia y el seguimiento de las acciones realizadas por el personal.

1.2. Análisis

A partir del relevamiento, se determinaron **cinco módulos funcionales** que reflejan las etapas clave del ciclo de vida de una reserva hotelera:

1. **Consulta de disponibilidad:** permite conocer las habitaciones libres en un rango determinado de fechas, aplicando filtros por tipo y capacidad.

2. **Registro de reserva:** genera una nueva reserva en estado *pendiente*, asociando los datos del huésped y el detalle de la solicitud.

3. **Confirmación de reserva:** actualiza el estado de la reserva a *confirmada* una vez

verificado el pago o garantía correspondiente.

4. **Preparación de la habitación:** coordina las tareas internas previas al check-in, como limpieza, reposición de insumos y control de mantenimiento.

5. **Check-in y atención al huésped:** formaliza el ingreso del huésped, valida su identidad y marca la habitación como *ocupada* dentro del sistema.

Los módulos se integran de manera **secuencial y dependiente**, garantizando que la salida de un proceso constituya la entrada del siguiente.
Esta estructura minimiza errores operativos y refuerza la coherencia interna del sistema.

Se determinó además la importancia de incorporar **reglas de negocio estrictas** (validaciones de fechas, capacidades y estados de reserva) para evitar conflictos como sobreasignaciones, reservas duplicadas o habitaciones no preparadas a tiempo.

1.3. Diseño del Sistema

El sistema fue concebido bajo una **arquitectura modular y escalable**, en la que cada proceso mantiene independencia lógica, pero se integra dentro de un flujo general coherente.

Cada módulo se define por los siguientes componentes:

- **Entradas:** datos necesarios para su ejecución (fechas, ID de habitación, datos del huésped, etc.).

- **Reglas de validación:** restricciones y condiciones de negocio para asegurar la consistencia de la información.

- **Salidas:** resultados del proceso (estado de la reserva, habitación asignada, comprobante, etc.).

- **Algoritmo:** secuencia de pasos operativos con distintos niveles de refinamiento.

- **Pseudocódigo:** representación formal del flujo lógico que facilita la implementación.

El diseño busca garantizar:

- **Trazabilidad completa** entre las distintas etapas de una reserva.

- **Integridad de los datos** relacionados con huéspedes, habitaciones y transacciones.

- **Secuencialidad y coherencia** en la gestión de cada proceso operativo.

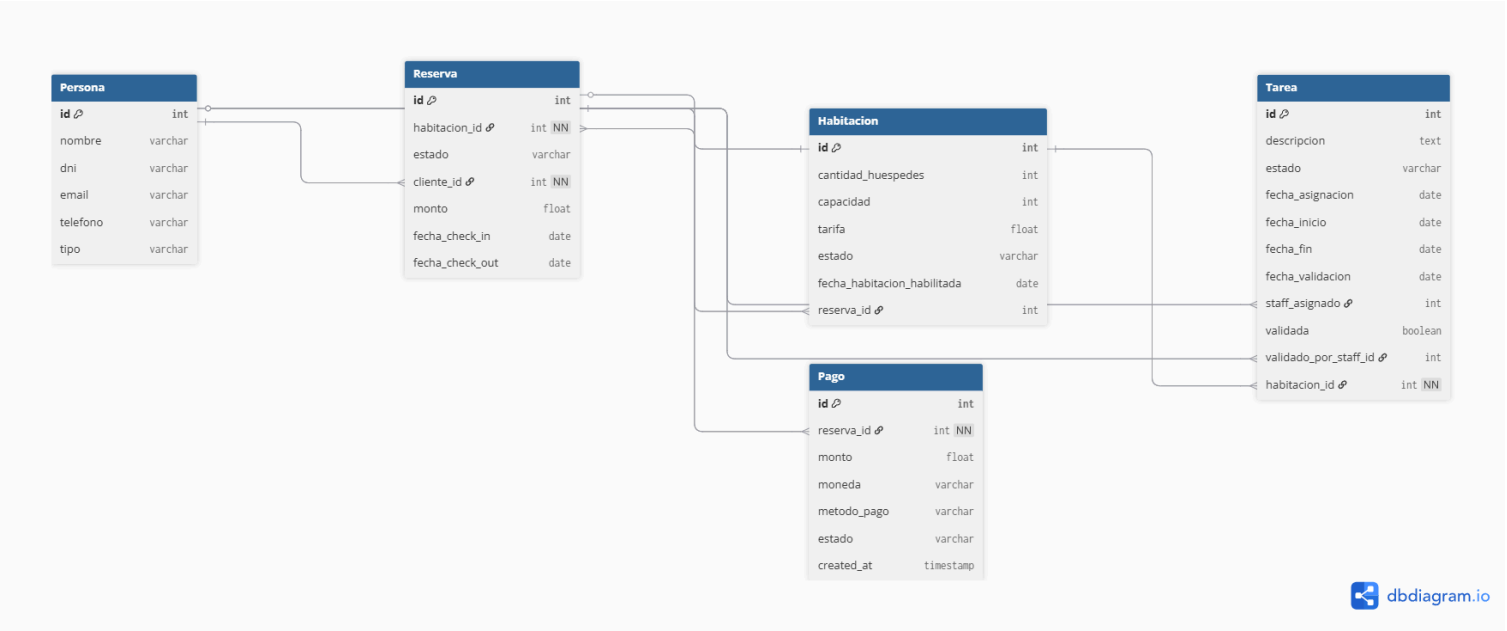
- **Facilidad de mantenimiento y extensión futura**, permitiendo agregar

funcionalidades (check-out, facturación, reportes, etc.) sin alterar la estructura base.

1.4. Objetivo General

Desarrollar un **sistema integral de gestión hotelera** que administre de forma ordenada y eficiente el ciclo completo de una reserva —desde la consulta inicial hasta el check-in—, garantizando la **consistencia de los datos**, la **eficiencia operativa** y una **experiencia fluida** tanto para el huésped como para el personal del hotel.
El sistema busca modernizar la operatoria interna, reducir errores humanos y establecer una base sólida para futuras mejoras tecnológicas, como la integración con plataformas de reservas en línea o sistemas de gestión contable.

2. DER (Diagrama entidad-relación)



El **Diagrama Entidad–Relación (DER)** define la estructura lógica de los datos que componen el sistema de gestión hotelera. Este modelo permite visualizar las entidades principales del sistema, sus atributos y las relaciones existentes entre ellas, asegurando la **integridad y consistencia** de la información a lo largo de todo el proceso operativo.

El diseño presentado a continuación contempla cinco entidades principales:

- **Persona:** almacena los datos personales y de contacto tanto de huéspedes como del personal del hotel. Incluye campos como nombre, DNI, correo electrónico, teléfono y tipo de persona (cliente o staff).
  Esta entidad permite una gestión unificada de usuarios y empleados bajo un mismo

esquema de datos.

- **Reserva:** representa el núcleo del sistema. Registra las operaciones de reserva de habitaciones, vinculando un huésped con una habitación durante un rango de fechas determinado.
  Contiene información como el estado (*pending*, *confirmed*), el monto total, las fechas de check-in/check-out y referencias a la habitación y al cliente.

- **Habitación:** modela las unidades físicas del hotel. Cada registro posee atributos de capacidad, tarifa, estado actual (disponible, ocupada, en preparación) y fecha de habilitación.
  Además, incluye una relación hacia la entidad *Reserva*, permitiendo conocer qué reserva está asociada actualmente a cada habitación.

- **Pago:** almacena la información de los pagos efectuados para confirmar reservas. Cada pago se vincula a una reserva específica e incluye monto, moneda, método de pago, estado (autorizado, capturado, etc.) y la fecha de creación.
  Esta entidad garantiza trazabilidad en las transacciones y permite registrar múltiples pagos por una misma reserva si fuera necesario.

- **Tarea:** representa las actividades operativas relacionadas con la preparación de las habitaciones (limpieza, reposición, mantenimiento).
  Incluye fechas de asignación, inicio, finalización y validación, así como referencias al miembro del *staff* responsable y al validador. Además, está directamente asociada a la *Habitación*, lo que permite gestionar el estado operativo previo al check-in.

Las **relaciones** más relevantes definidas en el modelo son:

- Una **Persona** puede generar **muchas Reservas** (1:N).

- Una **Reserva** está asociada a **una sola Habitación**, pero una habitación puede tener **muchas Reservas** a lo largo del tiempo (N:1).

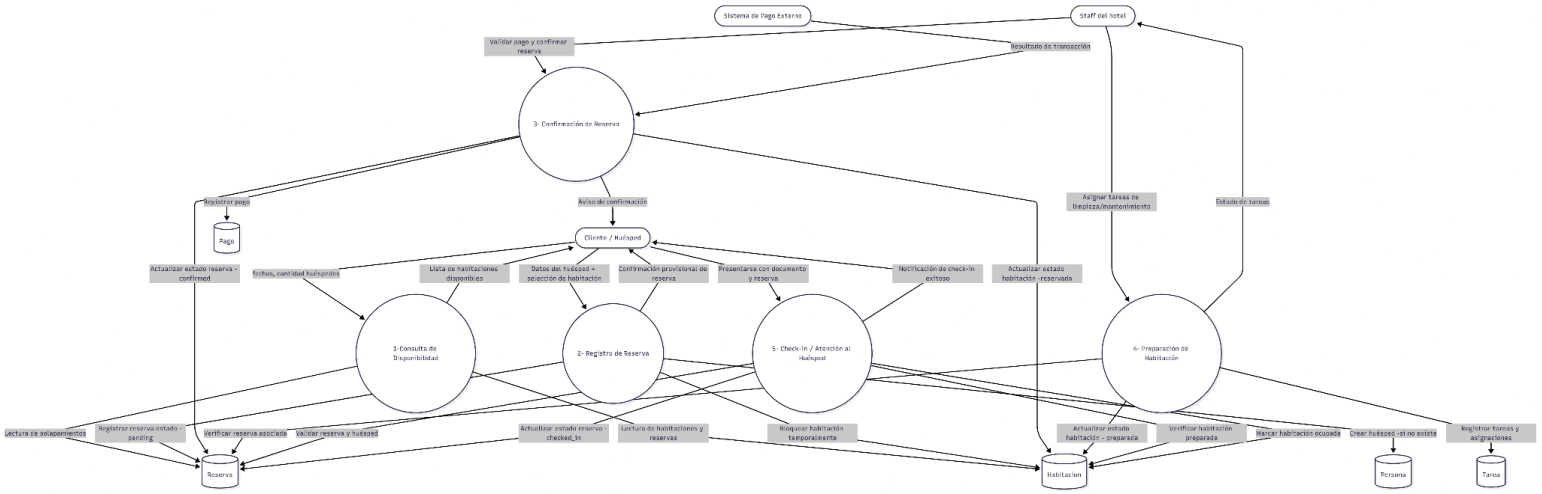
- Una **Reserva** puede estar vinculada con **uno o más Pagos** (1:N).

- Una **Habitación** puede tener **muchas Tareas** asignadas (1:N), las cuales son ejecutadas y validadas por el **staff** (referenciado desde *Persona*).

En conjunto, el modelo refleja un esquema **relacional normalizado**, que favorece la integridad referencial y evita redundancias.
El diseño también permite futuras extensiones, como la incorporación de entidades de *Check-out*, *Facturación*, *Servicios adicionales* o *Reportes estadísticos*, manteniendo la misma lógica estructural y la coherencia de los datos.

3. DFD (Diagrama de flujo de datos)

3.1 Sistema en general



El DFD general (nivel 0) presenta una visión integrada de los cinco procesos que conforman el ciclo operativo del sistema hotelero.
Cada proceso se encuentra interconectado de forma lógica, evidenciando la **dependencia secuencial** entre los módulos: desde la consulta inicial hasta el check-in efectivo del huésped.

Los principales flujos identificados son:

- La **interacción con el huésped**, que inicia el proceso al consultar disponibilidad y culmina con la notificación de check-in exitoso.

- Los **procesos administrativos**, donde el personal del hotel y el sistema de pagos gestionan las operaciones críticas de confirmación, limpieza y ocupación.

- Las **bases de datos centrales** —*Habitación, Reserva, Pago, Persona y Tarea*— que actúan como repositorios intermedios garantizando la persistencia y validación de la información.

Este nivel permite comprender el **funcionamiento global del sistema como un ecosistema integrado**, donde cada módulo realiza una función específica y contribuye a mantener la coherencia del flujo operativo.

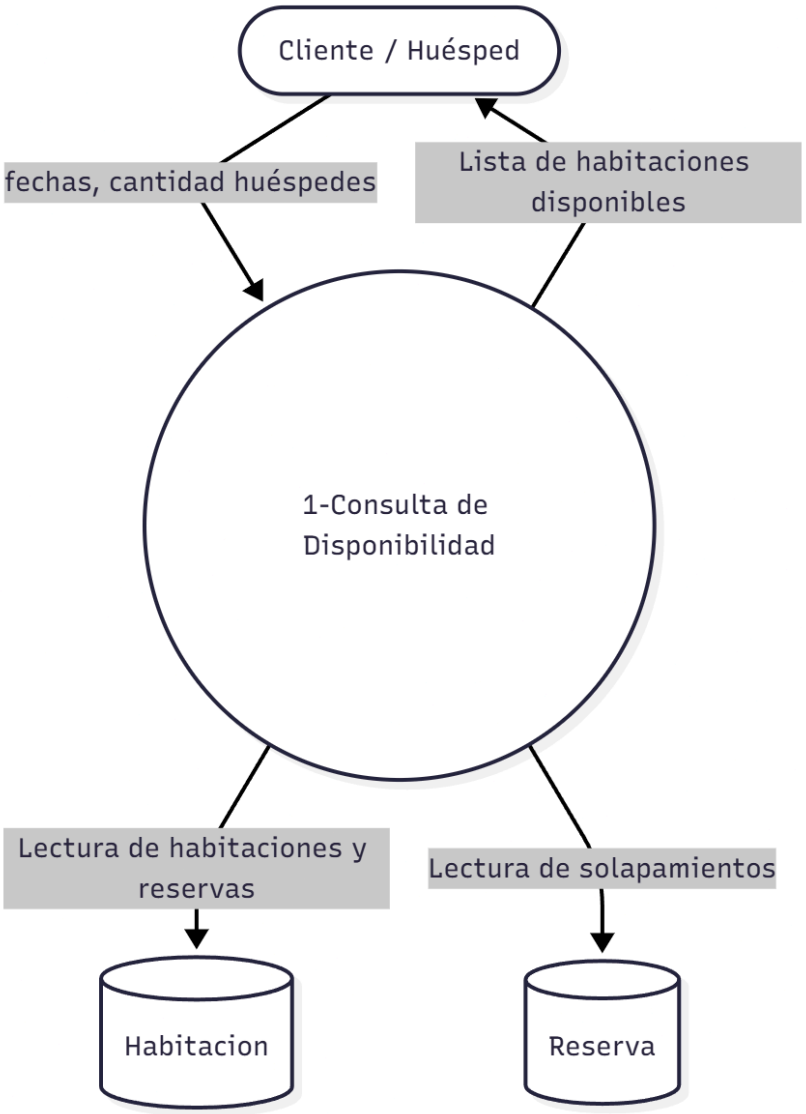
3.2 Módulos

3.2.1 Módulo 1

Este diagrama representa el punto de partida del proceso de reservas.
El **cliente/huésped** ingresa un rango de fechas y la cantidad de huéspedes deseada.
El sistema consulta la base de datos de *Habitaciones* y *Reservas* para identificar solapamientos y devolver una **lista de habitaciones disponibles**.

- **Entradas:** fechas, cantidad de huéspedes.
- **Salidas:** listado filtrado de habitaciones disponibles.
- **Almacenamientos:** Habitación, Reserva.

El proceso no genera modificaciones ni bloqueos de datos, respetando el principio de **no mutación del sistema** en la etapa de consulta.

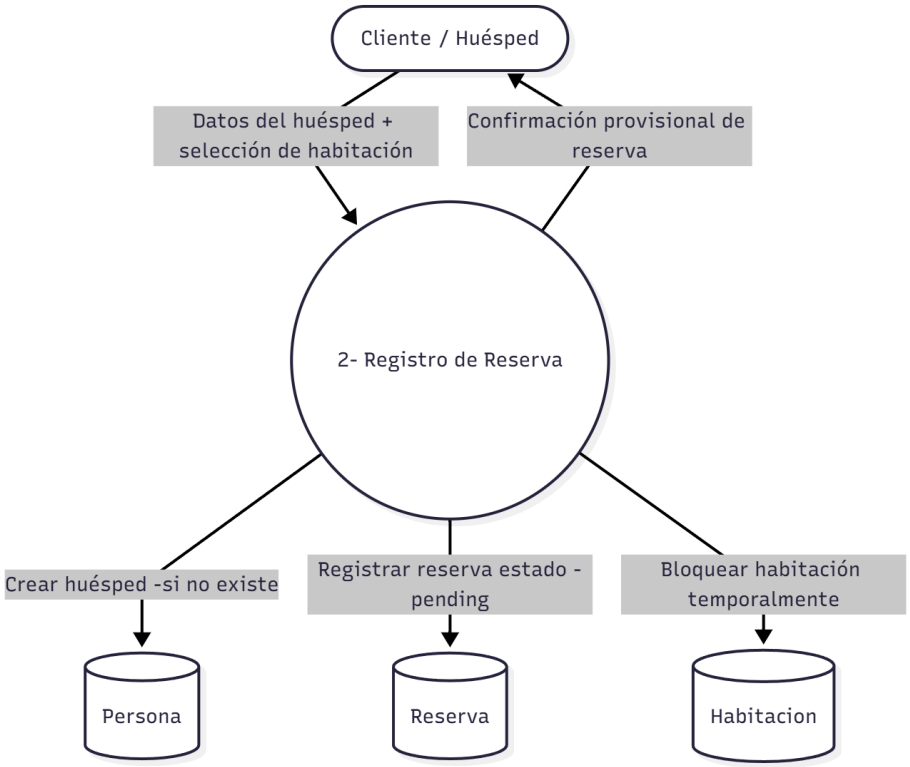


3.2.2 Módulo 2

Este diagrama refleja la creación de una **reserva preliminar (estado pending)**.
El huésped ingresa sus datos y selecciona una habitación disponible. El sistema verifica que no haya solapamientos, registra al huésped si no existe y crea la reserva temporal.

- **Entradas:** datos del huésped, habitación seleccionada.
- **Salidas:** confirmación provisional y número de reserva.
- **Almacenamientos:** Persona, Habitación, Reserva.

Este módulo inicia la **persistencia de datos**, ya que introduce registros nuevos que serán validados en los siguientes procesos.

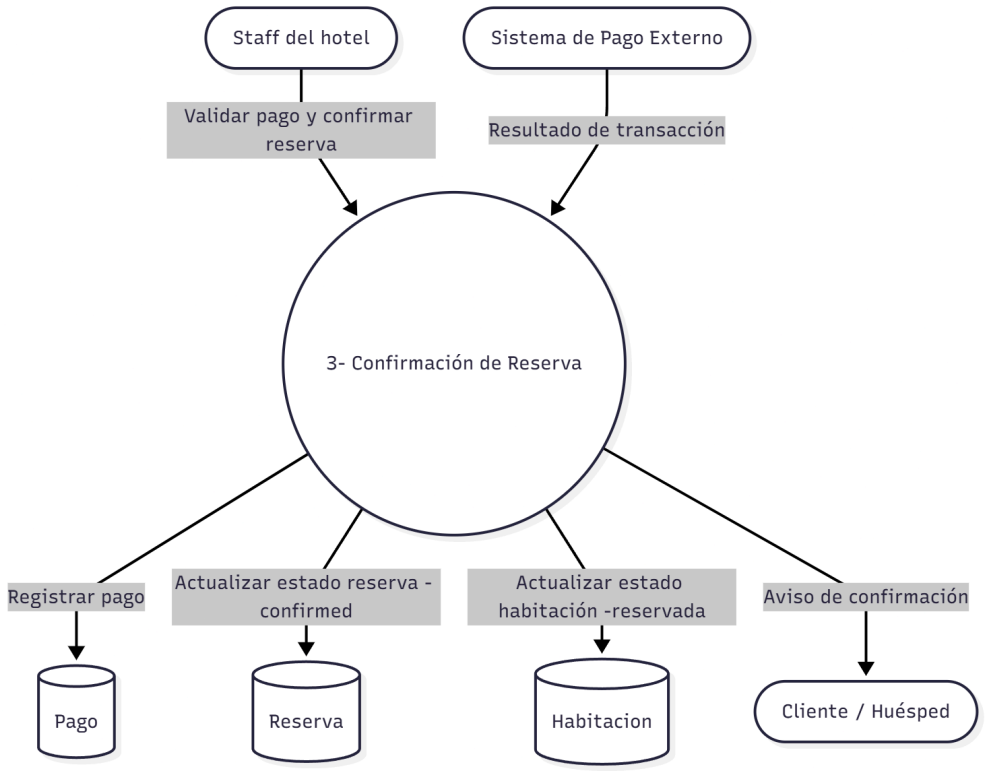


3.2.3 Módulo 3

Aquí se formaliza la reserva mediante la **validación del pago**.
El *staff del hotel* o el *sistema externo de pagos* envía la información de la transacción. Si el pago se encuentra aprobado (estado *captured*), el sistema actualiza la reserva a **confirmed** y cambia el estado de la habitación a **reservada**.

- **Entradas:** datos de pago, reserva pendiente.
- **Salidas:** comprobante de confirmación, actualización de estado.
- **Almacenamientos:** Pago, Reserva, Habitación.

Este módulo es clave porque consolida la **coherencia financiera y operativa** del proceso, garantizando que solo reservas con pago válido avancen al siguiente paso.



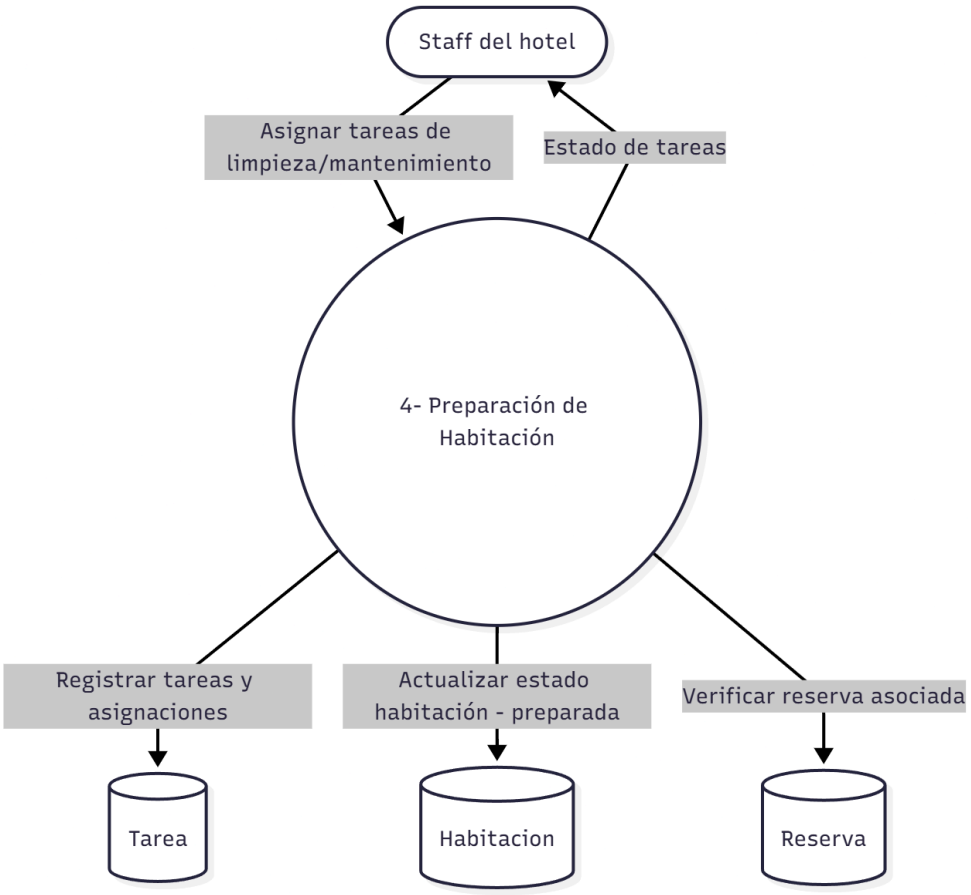
3.2.4 Módulo 4

El DFD de este módulo modela el flujo interno de preparación de habitaciones, gestionado por el *staff del hotel*.
Una vez confirmada la reserva, se generan **tareas operativas** (limpieza, mantenimiento, reposición) que deben ser completadas antes del check-in.

- **Entradas:** confirmación de reserva, personal disponible.

- **Salidas:** habitación marcada como preparada.
- **Almacenamientos:** Tarea, Habitación, Reserva.

El proceso promueve la **gestión de calidad interna**, permitiendo al hotel mantener un registro digital del estado de las tareas y el cumplimiento de los plazos operativos.

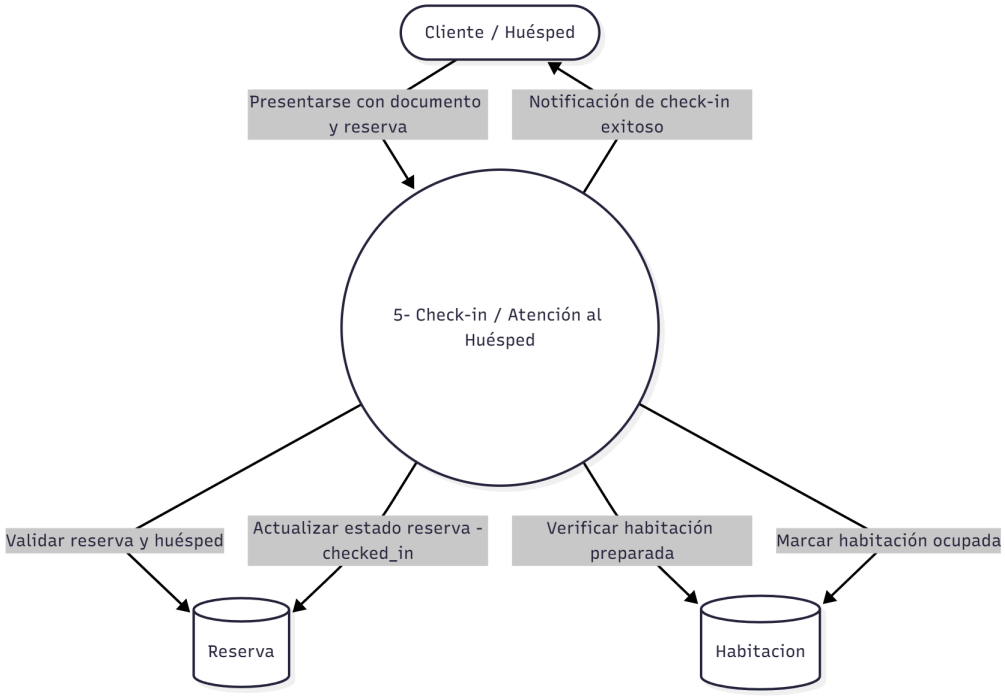


3.2.5 Módulo 5

El último DFD representa la instancia final del ciclo operativo.
El huésped se presenta con su documento y número de reserva; el sistema valida los datos, confirma que la habitación esté preparada y actualiza los estados a **ocupada** y **checked_in**.

- **Entradas:** reserva confirmada, identificación del huésped.
- **Salidas:** notificación de check-in exitoso, habitación ocupada.
- **Almacenamientos:** Reserva, Habitación.

Este proceso cierra el flujo de la reserva garantizando **consistencia entre la información administrativa y la ocupación real**.



3.3 Interpretación global

Los DFD demuestran cómo el sistema opera de forma **modular, segura y controlada**, garantizando la comunicación continua entre los actores externos (huésped, personal, sistema de pago) y las bases de datos.
Cada flujo asegura una **transformación válida de datos**: una entrada bien definida, un procesamiento lógico y una salida coherente con el estado del sistema.

Además, la descomposición progresiva desde el DFD general hasta los DFD específicos de módulo permite:

- Identificar los **puntos de interacción crítica** (por ejemplo, confirmación de pagos o validación de habitaciones).

- Mantener una **visión jerárquica del flujo de información**, facilitando futuras integraciones con otros subsistemas (check-out, reportes, mantenimiento).

- Asegurar una **alineación directa entre el modelo funcional (DFD)** y el modelo de datos (DER), fortaleciendo la consistencia entre la lógica de negocio y la estructura de la base.

4. Desarrollo de módulos

4.1 Módulo 1: Consulta de disponibilidad

Objetivo

Determinar qué habitaciones están libres en un rango de fechas [$check-in, check-out$) sin modificar el estado del sistema ni generar bloqueos o reservas.

Entradas

Las entradas del presente módulo son:

- ***fechaCheckIn*** (***Date***): inclusive, es decir, ese día la habitación debe estar libre.
- ***fechaCheckOut*** (***Date***): exclusivo, ya que ese día el huésped se retirará de la habitación.
- ***cantidadHuespedes*** (***entero*** ≥ **1**): indica el número de huéspedes que albergará la habitación.

Reglas y validaciones

- Fechas válidas: $checkIn$ < $checkOut$ y $checkIn$ > $hoy()$.
- La capacidad de la habitación debe ser igual o mayor a la capacidad ingresada por el usuario.
- No mutación de estado: la consulta no crea, bloquea o actualiza reserva ni tareas.
- Límites del rango: el máximo de noches son 14, por lo tanto, se debe validar el tamaño del intervalo.

Salida

- Lista de habitaciones disponibles, con los campos mínimos
  - ID
  - Capacidad.
  - Tarifa.
- Sin efectos laterales: no crea ni modifica reservas/bloqueos.

Algoritmo

1. Usuario ingresa capacidad, checkIn y checkOut.
2. Validar entradas: fechas, límites de noches (≤ 14) y capacidad (≥ 1 y ≤ 4).
3. Cargar universo de habitaciones activas.
4. Filtrar habitaciones disponibles y sin solapes, cuya capacidad sea mayor o igual a la solicitada por el usuario.
5. Construir la lista de disponibles con las habitaciones sin solapes.

6. Devolver la lista de habitaciones disponibles (sin mutaciones).

Refinamiento - Nivel 1

1. Usuario ingresa capacidad, checkIn y checkOut.
2. Validar entradas: fechas, límites de noches (≤ 14) y capacidad (≥ 1).
    2.1 Validar que la fecha de checkIn sea anterior a la fecha de checkOut.
    2.2 Validad que la cantidad de noches sea menor o igual a 14.
    2.3 Validar que la capacidad sea mayor o igual a 1.
    2.4 Validar que la capacidad sea menor o igual a 4.
3. Cargar universo de habitaciones activas.
4. Filtrar habitaciones disponibles y sin solapes, cuya capacidad sea mayor o igual a la solicitada por el usuario.
    4.1 Para cada habitación:
        4.1.1 Comprobar que la capacidad de la habitación sea mayor o igual que la capacidad ingresada por el usuario.
        4.1.2 Buscar reservas activas (*confirmed o pending*) que se solapen con el rango solicitado.
        4.1.3 Si no hay solapamiento, añadir la habitación a la lista de habitaciones disponibles.
5. Construir la lista de disponibles con las habitaciones sin solapes.
6. Devolver la lista de habitaciones disponibles (sin mutaciones).

Refinamiento - Nivel 2

1. Usuario ingresa fecha de capacidad, checkIn y checkOut.
    1.1 Leer la capacidad ingresada por el usuario.
    1.2 Leer la fecha de checkIn ingresada por el usuario.
    1.3 Leer la fecha de checkOut ingresada por el usuario
.2. Validar entradas: fechas, límites de noches (≤ 14) y capacidad (≥ 1).
    2.1 Validar que la fecha de checkIn sea anterior a la fecha de checkOut.
        2.1.1 Si la fecha de checkIn es menor a la fecha de checkOut, continuar. En caso contrario arrojar error (“La fecha de checkIn debe ser anterior a la fecha de checkOut).
    2.2 Validar que la cantidad de noches sea menor o igual a 14.
        2.2.1 Calcular la diferencia de días entre la fecha de checkOut y la fecha de checkIn.
        2.2.2 Si la diferencia es menor o igual a 14, continuar.
        2.2.3 En caso contrario, arrojar error (“El máximo de noches es 14”).
    2.3 Validar que la capacidad sea mayor o igual a 1.
        2.3.1 Si la capacidad es mayor o igual a 1, continuar.
        2.3.2 En caso contrario, arrojar error (“El mínimo de capacidad es de 1 persona”).
    2.4 Validar que la capacidad sea menor o igual a 4.
        2.4.1 Si la capacidad es menor o igual a 4, continuar.
        2.4.2 En caso contrario, arrojar error (“El máximo de capacidad es de 4 personas”).
3. Cargar universo de habitaciones activas.

3.1 Si no hay ninguna habitación activa, devolver una lista vacía.

4. Filtrar habitaciones disponibles y sin solapes, cuya capacidad sea mayor o igual a la solicitada por el usuario.

4.1 Para cada habitación:

4.1.1 Comprobar que la capacidad de la habitación sea mayor o igual que la capacidad ingresada por el usuario.

4.1.1.1 Si la capacidad de la habitación es mayor o igual que la capacidad ingresada por el usuario, continuar.

4.1.1.2 En caso contrario, continuar con la siguiente habitación.

4.1.2 Buscar reservas activas (*confirmed o pending*) que se solapen con el rango solicitado.

4.1.2.1 Para cada reserva activa en la habitación:

4.1.2.2 Si la fecha de checkIn de la reserva es mayor o igual a la ingresada por el usuario, o la fecha de checkOut es menor o igual a la ingresada por el usuario:

4.1.2.2.1 Establecer solapamiento como verdadero.

4.1.2.2.2 Continuar con la siguiente habitación.

4.1.3 Si no hay solapamiento, añadir la habitación a la lista de habitaciones disponibles.

4.1.3.1 Si solapamiento es falso, añadir la habitación a la lista de habitaciones disponibles.

5. Construir la lista de disponibles con las habitaciones sin solapes.

6. Devolver la lista de habitaciones disponibles (sin mutaciones).

Pseudocódigo

```
PROCESO Modulo 1
        //PASO 1: Usuario ingresa capacidad, checkIn y checkOut

        LEER capacidad
        LEER fechaCheckIn
        LEER fechaCheckOut

        //PASO 2: Validaciones

        SI fechaCheckIn >= fechaCheckOut ENTONCES
                ArrojarError("La fecha de checkIn debe ser anterior a la fecha de checkOut").
                Devolver []
        FIN SI

        noches <-diffDias(fechaCheckOut, fechaCheckIn)
        SI noches > 14 ENTONCES
                ArrojarError("El máximo de noches es 14")
                Devolver []
        FIN SI

        SI capacidad < 1 ENTONCES
                ArrojarError("El mínimo de capacidad es de 1 persona")
                Devolver []
        FIN SI

        SI capacidad > 4 ENTONCES
                ArrojarError("El máximo de capacidad es de 4 personas")
                Devolver []
        FIN SI

        //PASO 3: Cargar universo de habitaciones activas.
        habitaciones <- obtenerHabitacionesActivas()
        SI habitaciones ==  ∅ ENTONCES
                Devolver []
        FIN SI

        disponibles <- []

        //PASO 4: Filtrar por capacidad y solapes.

        PARA CADA hab EN habitaciones HACER
                SI hab.capacidad < capacidad ENTONCES
                        CONTINUAR //pasar a la siguiente habitación.
                FIN SI

                reservasActivas <- listarReservas (hab.id, estados ∈ {"confirmed","pending"})

                solapa <- FALSO
```

```
            PARA CADA r EN reservasActivas HACER
                    SI NO (r.checkOut ≤ fechaCheckIn O r.checkIn ≥ fechaCheckOut) ENTONCES
                            solapa <- VERDADERO
                            SALIR //cortar evaluación de reservas de esta habitación.
                    FIN SI
            FIN PARA
            SI NO solapa ENTONCES
                    (agregar(disponibles, hab)
            FIN SI
    FIN PARA


    //PASO 5: Devolver disponibles.

    DEVOLVER disponibles
FIN PROCESO
```

4.2 Módulo 2: Registro de reserva

Objetivo

Registrar una nueva reserva para una habitación seleccionada por el usuario. Se almacena la información del huésped, el rango de fechas, la cantidad de personas y se genera un número único de reserva. El estado inicial de la reserva es pending.

Entradas

- habitacion_id (número): ID de la habitación seleccionada.
- checkin_date (Date): Fecha de entrada (inclusive).
- checkout_date (Date): Fecha de salida (exclusiva).
- cantidad_huespedes (entero): Número de personas a alojar.
- huesped: objeto con:
    - nombre (string)
    - documento (string)
    - email (string)
    - telefono (string)

Reglas y validaciones

- La fecha de check-in debe ser anterior a la de check-out.
- El rango máximo es de 14 noches.
- La habitación debe existir, estar activa y tener capacidad suficiente.
- La habitación no debe tener solapamiento con otra reserva pending o confirmed.
- Los datos del huésped deben estar completos y válidos.
- Al registrar la reserva, la habitación queda bloqueada para ese rango de fechas (por X minutos).

Salida

- reserva_id: número/código único generado.
- Estado inicial de la reserva: "pending"
- Monto estimado (tarifa base × noches × cantidad huéspedes + impuestos estimados)
- Se registra el huésped si no existía.

- Se asocia la habitación a la reserva.

Algoritmo

1. Validar fechas, cantidad de huéspedes y formato de datos.
2. Verificar disponibilidad actual de la habitación (sin solapamientos).
3. Validar existencia de la habitación y su capacidad.
4. Crear registro de huésped (si no existe).
5. Crear nueva reserva en estado "pending"
6. Asociar habitación a la reserva.
7. Devolver ID de reserva y estado.

Refinamiento - Nivel 1

1. Validar entradas:
    1.1 Check-in < Check-out
    1.2 Diferencia de días ≤ 14
    1.3 Cantidad de huéspedes ≥ 1
    1.4 Datos del huésped no vacíos
2 Verificar habitación:
    2.1 Existe
    2.2 Capacidad ≥ cantidad de huéspedes
    2.3 No tiene reservas activas solapadas
3 Registrar huésped si no existe
4 Crear reserva:
    4.1 Guardar checkin_date, checkout_date, estado = "pending"
    4.2 Asignar huesped_id y habitacion_id
5 Calcular y mostrar monto estimado

Refinamiento - Nivel 2

1. Validar fechas
    1.1 Si checkin ≥ checkout → Error
    1.2 Si noches > 14 → Error
    1.3 Si cantidad huéspedes < 1 Error
2. Validar habitación
    2.1 Buscar habitación por habitacion_id
    2.2 Si no existe → Error
    2.3 Si capacidad < cantidad_huespedes → Error
    2.4 Buscar en reservas activas si hay solapamiento → Si hay → Error
3 Validar huésped
    3.1 Verificar que nombre, documento, email, teléfono ≠ vacío
    3.2 Si huésped con mismo documento ya existe → reutilizar huesped_id
4 Registrar reserva
    4.1 Crear objeto reserva con:
        4.1.1 ID generado
        4.1.2 Estado = "pending"
        4.1.3 Fechas

4.1.4 Habitación y huésped vinculados

4.2 Agregar a la lista de reservas activas

5 Salida

5.1 Mostrar: ID de reserva, estado = "pending", monto estimado

5.2 La habitación queda bloqueada temporalmente

Pseudocódigo

```
INICIO REGISTRAR_RESERVA

LEER habitacion_id, checkin_date, checkout_date, cantidad_huespedes
LEER nombre, documento, email, telefono

// Validar fechas
SI checkin_date ≥ checkout_date ENTONCES
   MOSTRAR "Error: Fechas inválidas"
   TERMINAR
FIN SI

SI DIAS_ENTRE(checkin_date, checkout_date) > 14 ENTONCES
   MOSTRAR "Error: Máximo 14 noches permitidas"
   TERMINAR
FIN SI

// Validar habitación
habitacion ← BUSCAR_HABITACION(habitacion_id)
SI habitacion = NULO ENTONCES
   MOSTRAR "Error: Habitación inexistente"
   TERMINAR
FIN SI

SI habitacion.capacidad < cantidad_huespedes ENTONCES
   MOSTRAR "Error: Capacidad insuficiente"
   TERMINAR
FIN SI
```

```
SI EXISTE_SOLAPAMIENTO(habitacion_id, checkin_date, checkout_date)
ENTONCES
   MOSTRAR "Error: Habitación no disponible en ese rango"
   TERMINAR
FIN SI

// Verificar o crear huésped
huesped ← BUSCAR_HUESPED(documento)
SI huesped = NULO ENTONCES
   huesped ← CREAR_HUESPED(nombre, documento, email, telefono)
FIN SI

// Crear reserva
reserva_id ← GENERAR_ID_UNICO()
reserva ← {
   id: reserva_id,
   huesped_id: huesped.id,
   habitacion_id: habitacion.id,
   checkin_date: checkin_date,
   checkout_date: checkout_date,
   estado: "pending"
}
AGREGAR_A_LISTA_RESERVAS(reserva)

// Calcular monto estimado
monto ← CALCULAR_MONTO(habitacion.tarifa, checkin_date, checkout_date,
cantidad_huespedes)

// Salida
```

4.3 Módulo 3: Confirmación de la reserva

Objetivo

Confirmar una reserva existente mediante el registro de un pago válido. El proceso asegura que la reserva esté activa, que el importe sea correcto, que el pago quede debidamente regis trado y que, si el pago fue efectivamente capturado, la reserva cambie su estado de pending a confirmed.

Entradas

- reserva_id (número): Identificador único de la reserva a confirmar.
- monto(número): Importe abonado por el cliente.
- moneda(string): Tipo de moneda utilizada (por ejemplo, “ARS”).
- metodo_pago (string):
- Medio de pago empleado (tarjeta, efectivo, transferencia, etc.).
- estado_pago (string): Estado del pago (authorized, captured, refunded, voided).
- staff_id (número): Identificador del empleado (tipo staff) responsable de la confirma ción

Reglas y validaciones

- La reserva debe existir y encontrarse en estado pending.
- Solo pueden confirmarse reservas que no estén canceladas ni completadas.
- El monto del pago debe ser mayor que cero.
- El estado del pago debe ser uno de los válidos: authorized, captured, refunded o voided.
- Todo pago registrado debe incluir método, monto, moneda, estado y fecha de creación
- Si el pago fue capturado, la reserva cambia automáticamente a estado confirmed.
- Si el pago no fue capturado, la reserva permanece pending.

Salida

- Estado final de la reserva: confirmed si el pago fue capturado; pending si no.
- Registro del pago con los campos (monto, moneda, método, estado, fecha de creación).
- Fecha y usuario que confirmaron la reserva (solo si fue capturada).
- Mensaje informativo con el resultado del proceso.

Algoritmo

1. Validar la existencia de la reserva y su estado actual.
2. Verificar que el importe del pago sea mayor a 0.
3. Registrar el pago con su monto, moneda, método y estado.
4. Analizar el resultado del pago:
    4.1. Si el pago está en estado captured, actualizar la reserva a confirmed y registrar fecha y staff responsable.

4.2. En caso contrario, mantener la reserva pending.
5. Retornar el estado final de la reserva junto con el registro del pago.

Refinamiento - Nivel 1

1. Validar datos de entrada:
    1.1. Verificar que la reserva exista.
    1.2. Validar que el estado actual sea pending.
    1.3. Comprobar que el monto sea mayor a 0.
2. Registrar el pago:
    2.1. Crear un nuevo pago asociado a la reserva
    2.2. Registrar método, moneda, monto, estado y fecha de creación.
    2.3. Aceptar únicamente estados válidos: authorized, captured, refunded, voided.
3. Confirmar la reserva (si corresponde):
    3.1. Si el pago fue captured, actualizar estado de reserva a confirmed.
    3.2. Registrar fecha y usuario (staff_id) que realizó la confirmación.

Refinamiento - Nivel 2

1. Validar datos:
    1.1. Si la reserva no existe → Error: “Reserva inexistente”.
    1.2. Si la reserva no está en estado pending → Error: “Reserva no disponible para con
    firmar”.
    1.3. Si el monto 0 →Error: “Monto inválido”.
2. Registrar pago:
    2.1. Crear registro de pago con los datos ingresados.
    2.2. Validar que el estado del pago esté dentro de los valores permitidos.
    2.3. Guardar fecha y hora del registro.
3. Confirmar reserva:
    3.1. Si el estado del pago es captured, cambiar la reserva a confirmed.
    3.2. Registrar fecha y usuario (staff_id) que realizó la confirmación.
    3.3. Si no fue capturado, mantener pending.
4. Retornar salida:
    4.1. Estado final de la reserva.
    4.2. Datos del pago asociado.
    4.3. Mensaje de confirmación o aviso de pendiente.

Pseudocódigo

```
INICIO CONFIRMAR_RESERVA

LEER reserva_id, monto, moneda, metodo_pago, estado_pago, staff_id

// Validar reserva
SI RESERVA_NO_EXISTE(reserva_id) ENTONCES
    MOSTRAR "Error: reserva inexistente"
    TERMINAR
FIN SI

SI ESTADO(reserva_id) != "pending" ENTONCES
    MOSTRAR "Error: la reserva no puede ser confirmada"
    TERMINAR
FIN SI

SI monto <= 0 ENTONCES
    MOSTRAR "Error: monto invalido"
    TERMINAR
FIN SI

// Registrar pago
pago :=
CREAR_PAGO(reserva_id, monto, moneda, metodo_pago, estado_pago, FECHA_ACTUAL)

// Confirmar reserva si el pago fue capturado
SI estado_pago = "captured" ENTONCES
    ACTUALIZAR_ESTADO_RESERVA(reserva_id, "confirmed")
    REGISTRAR_CONFIRMACION(reserva_id, staff_id, FECHA_ACTUAL)
    MOSTRAR "Reserva confirmada correctamente"
SINO
    MOSTRAR "Reserva permanece pendiente"
FIN SI

FIN
```

4.4 Módulo 4: Preparación de la habitación

Objetivo

En este módulo se trabaja la preparación de la habitación, previa al check-in e ingreso de un nuevo huésped. Cada actividad involucrada en dejar la habitación preparada deberá realizarse por 1 persona asignada, cumpliendo con un plazo de tiempo definido: debe ser realizada previa al horario en el que se realiza la entrada a la habitación de un nuevo huésped.

Las tareas que se deben realizar para dejar la habitación preparada son:

- Limpiar los pisos.
- Cambiar sábanas y toallas.
- Reponer el frigobar.

Cuando cada tarea presenta *isValidated*:*true* la habitación se habilita para ser ocupada, lo cual se debe cumplir antes del horario de entrada asignado del huésped. Cuando la habitación es desocupada, se resetea por completo el estado de cada tarea asociada a dicha habitación.

El flujo dentro del módulo estaría definido por los siguientes métodos:

- Asignar tarea a personal.
- Realizar tarea.
- Finalizar tarea.
- Validar tarea.
- Establecer habitación como preparada.

Entradas

- reserva_id (número): Identificador único de la reserva a confirmar.
- Estado final de la reserva: confirmed si el pago fue capturado; pending si no.
- habitación_id (número): Identificador único de la habitación asociada a la reserva.
- user_id (número): Identificador del usuario que realiza la reserva.

Reglas y validaciones

- La reserva asociada debe existir y estar confirmada.
- La fecha de check-in debe ser futura respecto a la fecha actual.
- La habitación no debe estar ocupada ni en mantenimiento.
- Las tareas deben existir y reiniciarse correctamente (estado = pendiente).
- Debe haber suficiente personal de limpieza disponible.
- Cada tarea debe tener un único responsable asignado.
- Las fechas de inicio y fin de tareas deben seguir un orden lógico.
- Solo pueden validarse tareas finalizadas y antes del check-in.

- Todas las tareas deben estar validadas para habilitar la habitación.
- Si no todas están validadas, se extiende el check-in y no se habilita la habitación.

Salida

La salida de este módulo es la habitación preparada y lista para ingresar el nuevo huésped.

Algoritmo

1. Comprobar si la habitación asignada ya fue reservada nuevamente.
2. Comprobar si la habitación asignada ya no se encuentra ocupada.
3. Reestablecer estado de preparación de la habitación.
4. Asignar tareas de preparación a personal disponible.
5. Gestionar ejecución de tareas.
6. Verificar que todas las tareas estén validadas para preparar la habitación.

Refinamiento - Nivel 1

1. Comprobar si la habitación asignada ya fue reservada nuevamente.
    1.1 Verificar si existe una nueva reserva válida.
2. Comprobar si la habitación asignada ya no se encuentra ocupada.
    2.1 Verificar si el estado de la habitación asignada ya no es "ocupada".
3. Reestablecer el estado de preparación de la habitación.
    3.1 Establecer como pendiente cada actividad y el estado general de preparación de la habitación.
4. Asignar tareas de preparación a personal disponible.
    4.1. Vincular cada tarea a un personal de limpieza del hotel disponible.
5. Gestionar ejecución de tareas
    5.1 Iniciar tarea.
    5.2 Finalizar tarea.
    5.3 Validar tarea.
6. Verificar que todas las tareas estén validadas para preparar la habitación.
    6.1. Comparar el horario de validación de cada tarea con el del check-in y verificar que este sea mayor.

Refinamiento - Nivel 2

1. Comprobar si la habitación asignada ya fue reservada nuevamente.
    1.1 Verificar si existe una nueva reserva reserva válida
        1.1.1 Si la fecha_check_in asociada a la reserva es mayor a la fecha_actual y las condiciones de reserva son válidas
            1.1.1.1 Mostrar "La habitación tiene una próxima reserva".
        1.1.2 Sino
            1.1.2.1 Mostrar "La habitación no tiene una próxima reserva".

2. Comprobar si la habitación asignada ya no se encuentra ocupada.

2.1 Verificar si el estado de la habitación asignada ya no es "ocupada".

2.1.1 Si el estado de la habitación asociada a la reserva es "ocupada"

2.1.1.1 Mostrar "La habitación se encuentra ocupada, no se procede a preparar".

2.1.2 Sino

2.1.1.2 Mostrar "La habitación se encuentra desocupada, se procede a preparar".

3. Reestablecer el estado de preparación de la habitación.

3.1 Establecer como pendiente cada actividad y el estado general de preparación de la habitación.

3.1.1 Para cada tarea de la lista de tareas asociada a la habitación.

3.1.1.1. Establecer estado en pendiente.

3.1.1.2. Establecer validación en falso.

3.1.2 Cambiar estado de la habitación a "desalojada".

4. Asignar tareas de preparación a personal disponible.

4.1. Vincular cada tarea a un personal de limpieza del hotel disponible.

4.1.1 Obtener una lista de personas de personal de limpieza con estado "disponible"

4.1.2 Para tantas personas de la lista como tareas haya

4.1.2.1 Asignar una tarea en estado “pendiente”.

4.1.2.2 Registrar fecha de asignación con la fecha actual.

4.1.2.3 Registrar la persona responsable de la tarea.

4.1.2.4 Cambiar estado del personal a “ocupado”.

5. Gestionar ejecución de tareas

5.1 Iniciar tarea.

5.1.1 Seleccionar tarea en estado “pendiente”.

5.1.2 Cambiar estado a “enProgreso”.

5.1.3 Registrar fecha_inicio como fecha_actual.

5.2 Finalizar tarea.

5.2.1 Seleccionar tarea en estado “enProgreso”.

5.2.2 Cambiar estado a “finalizado”.

5.2.3 Registrar fecha_fin como fecha_actual.

5.3 Validar tarea.

5.3.1 Seleccionar tarea en estado “finalizado”.

5.3.2 Verificar cumplimiento de condiciones de validación.

5.3.3 Registrar fecha_validación como fecha_actual.

5.3.4 Cambiar atributo validada a VERDADERO.

6. Verificar que todas las tareas estén validadas para preparar la habitación.

6.1. Comprobar validez de cada tarea y verificar que la fecha de validación no sea mayor que la fecha de check-in.

6.1.1 Obtener la lista de tareas asociadas a la habitación.

6.1.2 Para cada tarea dentro de la lista

6.1.2.1 Si la tarea no está validada o su FECHA_VALIDACIÓN supera FECHA_CHECK_IN

6.1.2.1.1 Establecer que no todas las tareas están validadas

6.1.3 Si todas las tareas están validadas

6.1.3.1 Ejecutar EstablecerHabitacionPreparada(habitaciónId)

6.1.4 Sino

6.1.4.1 Ejecutar ExtenderFechaCheckIn(habitaciónId, 1 hora)

Pseudocódigo

```
PROCESO PrepararHabitación(habitacion_id, reserva_id, estado_reserva, user_id)
        ComprobarReserva(reserva_id, estado_reserva)
        ComprobarOcupación(habitacion_id)
        ReiniciarTareas(habitacion_id)
        AsignarTareas(habitacion_id)
        RegistrarHorarios(habitacion_id)
        ValidarTareas(reserva_id, habitacion_id)
FIN PROCESO

PROCESO ComprobarReserva(reserva_id, estado_reserva, user_id)
        reserva <- obtenerReservaPorId(reserva_id)
        SI reserva.fecha_check_in > fecha_actual Y estado_reserva == "confirmed" Y reserva.cliente_id == user_id ENTONCES
                imprimir("La habitación tiene una próxima reserva")
        SINO
                imprimir("La habitación no tiene una próxima reserva")
        FIN SI
FIN PROCESO
```

```
PROCESO ComprobarOcupacion(habitacion_id)
        habitacion <- obtenerHabitacionPorId(habitacion_id)
        SI habitacion.estado == "ocupada" ENTONCES
                imprimir("La habitación se encuentra ocupada, no se procede a preparar")
                retornar
        SINO
                imprimir("La habitación se encuentra desocupada, se procede a preparar")
        FIN SI
FIN PROCESO

PROCESO ReestablecerEstado(habitacion_id)
        habitacion <- obtenerHabitacionPorId(habitacion_id)
        listaDeTareas <- habitacion.tareas
        PARA cada tarea en listaDeTareas:
                tarea.estado <- "pendiente"
                tarea.validad <- falso
        FIN PARA
        habitacion.estado <- "desalojada"
        habitacion.tareas <- listaDeTareas
FIN PROCESO
```

```
PROCESO AsignarTareas(habitacion_id)
        habitacion <- obtenerHabitacionPorId(habitacion_id)
        listaDeTareas <- habitacion.tareas
        personal <- obtenerPersonasDeLimpiezaDisponibles()
        SINO longitud(personal) < longitud(listaDeTareas) ENTONCES
                imprimir("No hay personal suficiente para realizar todas las tareas")
                retornar
        SINO
                PARA i DESDE 0 HASTA minimo(longitud(personal), longitud(listaDeTareas)):
                        tarea <- listaTareas[i]
                        persona <- listaPersonal[i]
                        tarea.estado <- "pendiente"
                        tarea.fecha_asignacion <- fecha_actual
                        tarea.staff_asignado <- persona.id
                        persona.estado <- "ocupado"
                FIN PARA
        FIN SI
FIN PROCESO

PROCESO GestionarTareas(habitacion_id)
        IniciarTarea(habitacion_id)
        FinalizarTarea(habitacion_id)
        ValidarTarea(habitacion_id)
FIN PROCESO
```

```
PROCESO VerificarValidación(reserva_id, habitacion_id)
        todasValidadas <- VERDADERO
        habitacion <- obtenerHabitacionPorId(habitacion_id)
        reserva <- obtenerReservaPorId(reserva_id)

        PARA cada tarea en habitación.tareas HACER
                SI tarea.validada == FALSO O tarea.fecha_validacion > reserva.fecha_check_in ENTONCES
                        todasValidadas <- FALSO
                        notificar("Tarea " + tarea.id + " no validada a tiempo.")
                FIN SI
        FIN PARA
        SI todasValidadas == VERDADERO ENTONCES
                EstablecerHabitaciónPreparada(habitacion_id)
                imprimir("Habitación listra para check-in")
        SINO
                ExtenderFechaCheckIn(reserva_id, 1 hora)
                imprimir("Check-in extendido por tareas pendientes")
        FIN SI
FIN PROCESO

PROCESO ExtenderFechaCheckIn(reserva_id, tiempo)
    reserva <- obtenerReservaPorId(reserva_id)
    nuevaFecha <- reserva.fecha_check_in + tiempo
    SI nuevaFecha < reserva.fecha_check_out ENTONCES
        reserva.fecha_check_in <- nuevaFecha
        imprimir("El check-in se extendió " + tiempo)
    SINO
        imprimir("No se puede extender el check-in: supera la fecha de salida.")
    FIN SI
FIN PROCESO
```

```
PROCESO EstablecerHabitaciónPreparada(habitacionId)
        habitacion <- obtenerHabitacionPorId(habitacion_id)
        habitacion.estado <- "preparada"
        habitacion.fecha_habitacion_habilitada <- fechaActual
        imprimir("Habitación lista para check-in")
FIN PROCESO

PROCESO IniciarTarea(habitacionId)
    tarea <- seleccionarTarea(habitacionId, "pending")
    SI tarea /= NULL ENTONCES
        tarea.estado ← "inProgress"
        tarea.fecha_inicio ← fechaActual
        imprimir("Tarea " + tarea.id + " iniciada.")
    SINO
        imprimir("No hay tareas pendientes para iniciar.")
    FIN SI
FIN PROCESO
```

```
PROCESO FinalizarTarea(habitacionId)
    tarea <- seleccionarTarea(habitacionId, "inProgress")
    SI tarea ≠ NULL ENTONCES
        tarea.estado <- "finished"
        tarea.fecha_fin ← fechaActual
        imprimir("Tarea " + tarea.id + " finalizada.")
    SINO
        imprimir("No hay tareas en progreso para finalizar.")
    FIN SI
FIN PROCESO

PROCESO ValidarTarea(habitacionId)
    tarea <- seleccionarTarea(habitacionId, "finished")
    SI tarea /= NULL ENTONCES
        condicionesCumplidas <- verificarCondicionesValidación(tarea.id)

        SI condicionesCumplidas = VERDADERO ENTONCES
            tarea.validada <- VERDADERO
            tarea.fecha_validacion <- fechaActual
            imprimir("Tarea " + tarea.id + " validada correctamente.")
        SINO
            imprimir("La tarea " + tarea.id + " no cumple condiciones de validación.")
        FIN SI
    SINO
        imprimir("No hay tareas finalizadas para validar.")
    FIN SI
FIN PROCESO
```

4.5 Módulo 5: Check-in y atención al huésped

Objetivo

Formalizar la llegada del huésped y marcar la ocupación real de la habitación. Este proceso debe asegurar que la reserva esté confirmada, la habitación esté preparada y se realice dentro del rango de la fecha de check-in.

Entradas

- reserva_id (número): Identificador único de la reserva para realizar el check-in.
- identificacion_huesped (string): Documento de identidad presentado por la persona que se presenta para el check-in.
- fecha_actual (Date): Fecha y hora en la que se realiza la operación.

Reglas y validaciones

- La reserva identificada por ‘reserva_id‘ debe existir.
- La reserva debe estar en estado ”confirmed”.
- La habitación asociada a la reserva debe estar en estado ”preparada”.
- El check-in debe realizarse en la fecha de check-in registrada en la reserva o en una fecha cercana (dentro de un rango de tolerancia, por ejemplo, el mismo día).
- Se debe vincular al huésped de la reserva con la habitación al establecer la ocupación (actualizar el campo ‘reserva_id‘ en la entidad ‘Habitacion‘ y el campo ‘huespedId‘ en la habitación, si existiera).
- La identificación presentada (identificacion_huesped) debe coincidir con el documento del huésped titular de la reserva.
- El proceso debe modificar el estado de la habitación a ”ocupada”.

Salida

- La reserva asociada a la habitación queda ocupada (se registra la fecha y hora de la ocupación real).
- La habitación cambia su estado a ”ocupada”.
- Se registra el huésped como ocupante de la habitación (actualización de ‘Habitacion.reserva_id‘ y potencialmente ‘Habitacion.huesped_id‘).
- Mensaje informativo con el resultado del proceso.

Algoritmo

1. Validar la existencia de la reserva y su estado (debe ser ”confirmed”).
2. Verificar la identidad del huésped, comparando la identificación presentada con la registrada en la reserva.
3. Verificar que la habitación asociada esté en estado ”preparada”.
4. Comprobar que la operación se realice dentro de la fecha de check-in permitida.
5. Actualizar el estado de la habitación a ”ocupada” y registrar la fecha/hora del check-in efectivo.
6. Vincular la reserva a la habitación como la ocupación actual.
7. Retornar el estado de la operación y la confirmación de la ocupación.

Refinamiento - Nivel 1

1. Validar reserva y habitación:
    1.1. Buscar la reserva por ‘reserva_id‘. Si no existe, arrojar error.
    1.2. Verificar que el estado de la reserva sea ”confirmed”. Si no lo es, arrojar error.
    1.3. Comparar la identificacion_huesped presentada con el documento del titular de la reserva. Si no coinciden, arrojar error.
2. Validar Habitacion:
    2.1. Obtener la habitación vinculada a la reserva.
    2.2. Verificar que el estado de la habitación sea ”preparada”. Si no lo es, arrojar error.
3. Validar fecha de check-in:
    3.1. Comprobar que la ‘fecha_actual‘ (fecha de la operación) sea igual o posterior a la ‘fecha_check_in‘ de la reserva. Si no, arrojar advertencia o error.
4. Registrar ocupación:
    4.1. Actualizar el campo ‘estado‘ de la habitación a ”ocupada”.
    4.2. Registrar la ‘reserva_id‘ en la habitación (campo ‘reserva_id‘ de ‘Habitacion‘).
    4.3. Registrar la fecha y hora de la ocupación efectiva (ej. en la reserva o en la habitación, usando un nuevo campo como ‘fecha_check_in_efectivo‘).
5. Devolver confirmación.

Refinamiento - Nivel 2

1. Validar Reserva y Habitación:
    1.1. ‘reserva = BUSCAR_RESERVA(reserva_id)‘
        1.1.1. Si ‘reserva‘ es NULO, arrojar error: ”Reserva inexistente.”
    1.2. Si ‘reserva.estado ≠ ”confirmed”, arrojar error: ”La reserva no está confirmada.”
    1.3. huesped = BUSCAR_HUESPED(reserva.huesped_id)
        1.3.1. Si huesped.documento != identificacion_huesped, arrojar error: "La identificación no coincide con el titular de la reserva".
2. Validar Habitacion
    2.1. ‘habitacion = BUSCAR_HABITACION(reserva.habitacion_id)‘
    2.2. Si ‘habitacion.estado ≠ ”preparada”, arrojar error: ”La habitación no está lista para el check-in.”
3. Validar Fecha y Hora:
    3.1. Si ‘FECHA(fecha_actual)‘ es anterior a ‘FECHA(reserva.fecha_check_in)‘, arrojar error:
”Check-in anticipado no permitido, la fecha es FECHA(reserva.fecha_check_in).”
4. Registrar Ocupación:
    4.1. ACTUALIZAR_HABITACION (‘habitacion.id‘, ‘estado‘ = ”ocupada”, ‘reserva_id‘ = ‘reserva.id‘, fecha_habitacion_habitada = fecha_actual)

5. Salida:
    5.1. MOSTRAR”Check-in realizado. Habitación habitacion.id ocupada por la Reserva reserva_id.”

Pseudocódigo

```
PROCESO CHECK_IN (reserva_id, identificacion_huesped, fecha_actual)
// PASO 1: Validar Reserva y Habitación
reserva <- BUSCAR_RESERVA(reserva_id)
SI reserva = NULO ENTONCES
    MOSTRAR "Error: Reserva inexistente"
    TERMINAR
FIN SI

SI reserva.estado != "confirmed" ENTONCES
    MOSTRAR "Error: La reserva no está confirmada"
    TERMINAR
FIN SI

huesped <- BUSCAR_HUESPED_POR_RESERVA(reserva.id)
    SI huesped.documento != identificacion_huesped ENTONCES
        MOSTRAR "Error: La identificación no coincide con el titular de la reserva"
        TERMINAR
    FIN SI

// PASO 2: Validar Habitacion
habitacion <- BUSCAR_HABITACION(reserva.habitacion_id)
SI habitacion.estado != "preparada" ENTONCES
    MOSTRAR "Error: La habitación no está preparada para el check-in"
    TERMINAR
FIN SI

// PASO 3: Validar Fecha de Check-in
SI FECHA(fecha_actual) < FECHA(reserva.fecha_check_in) ENTONCES
    MOSTRAR "Error: Check-in anticipado no permitido. La fecha es " +
    FECHA(reserva.fecha_check_in)
    TERMINAR
FIN SI

// PASO 4: Registrar Ocupación
ACTUALIZAR_HABITACION (
    habitacion.id,
    estado: "ocupada",
    reserva_id: reserva.id
    fecha_habitacion_habitada: fecha_actual
)


// PASO 5: Devolver Salida
MOSTRAR "Check-in realizado. Habitación " + habitacion.id +
" ocupada por la Reserva " + reserva_id + "."
DEVOLVER ÉXITO
FIN PROCESO
```