# 

# 

# 

# 

# 

# 

# 

# 

# 

# Narrativa, problemas enfrentados y diagrama de estructuras

[Qué hace nuestro programa 2](#_Toc138352745)

[Estructuras 2](#_Toc138352746)

[Sobre las librerías 5](#_Toc138352747)

[Problemáticas 5](#_Toc138352748)

[Resolución con las pilas 5](#_Toc138352749)

[Arreglo de palabras 6](#_Toc138352750)

[Pull request 6](#_Toc138352751)

[Aspectos organizativos 7](#_Toc138352752)

# Qué hace nuestro programa

Con nuestro programa, pueden acceder de manera segura a través de un proceso de inicio de sesión personalizado. Cada equipo tiene su propia contraseña exclusiva para garantizar la privacidad y seguridad de los datos. Ya sea que trabajen como mozos, recepcionistas, personal de limpieza o administradores, nuestro sistema se adapta a sus roles y necesidades específicas.

Una de las características destacadas de nuestro sistema es la gestión de reservas de habitaciones. Pueden realizar y administrar fácilmente las reservas de sus huéspedes, así como monitorear el estado de limpieza de las habitaciones en tiempo real.

Para los mozos, hemos desarrollado un menú especializado que les permite presentar deliciosas opciones de comidas a los huéspedes, brindándoles una experiencia culinaria excepcional. Además, podrán gestionar de manera eficiente el servicio de botellas, asegurándose de que los huéspedes disfruten de una selección de bebidas de calidad durante su estancia.

Los recepcionistas tienen acceso a un conjunto de herramientas diseñadas específicamente para facilitar la gestión de las reservas de los huéspedes. Mediante un menú intuitivo, pueden asignar habitaciones, administrar disponibilidad y generar pagos de forma segura y conveniente. Esto simplifica el proceso tanto para los huéspedes como para el personal del hotel.

El personal de limpieza también se beneficia de nuestro sistema, ya que pueden utilizar un menú especializado que garantiza que todas las habitaciones estén limpias y en óptimas condiciones para los huéspedes. Además, podrán gestionar el servicio de botellas para asegurarse de que esté debidamente abastecido y disponible para los huéspedes.

Los administradores tienen acceso a un conjunto completo de funciones de gestión. Desde la administración de empleados hasta la supervisión de las reservas y el estado de las habitaciones, nuestro sistema les permite tomar decisiones estratégicas y eficientes para asegurar el éxito operativo del hotel. Tienen el control total y pueden gestionar todos los aspectos importantes de la industria hotelera.

# Estructuras

Utilizamos cuatro estructuras de datos con sus respectivos requisitos: alta, baja, modificación, listado y ordenaciones. Consideramos que la persistencia de datos de estos archivos es imprescindible, cuidar los datos del cliente. por lo que tener la posibilidad de conservarlos, afuera del programa, es una gran idea. Estas cuatro estructuras son:

La estructura **stEmpleado** representa a un empleado y contiene los siguientes campos:

* nombreyApellido: una cadena de caracteres que almacena el nombre y apellido del empleado.
* edad: un entero que guarda la edad del empleado.
* genero: una cadena de caracteres que indica el género del empleado.
* telefono: un entero que almacena el número de teléfono del empleado.
* dni: un entero que representa el número de documento de identidad del empleado.
* correo: una cadena de caracteres que guarda la dirección de correo electrónico del empleado.
* nacionalidad: una cadena de caracteres que indica la nacionalidad del empleado.
* oficio: una cadena de caracteres que describe la ocupación o el trabajo realizado por el empleado.
* fechaAlta: una variable de tipo tiempo que registra la fecha de alta del empleado.
* fechaBaja: una variable de tipo tiempo que registra la fecha de baja del empleado.
* bajaPasiva: un entero que indica si la baja del empleado fue pasiva (1) o no (0).

La estructura **strHabitacion** representa una habitación y contiene los siguientes campos:

* ocupado: un entero que indica si la habitación está ocupada (1) o no (0).
* reservado: un entero que indica si la habitación está reservada (1) o no (0).
* numeroDeHabitacion: un entero que identifica el número de la habitación.
* capacidadDeHuespedes: un entero que especifica la capacidad máxima de huéspedes permitidos en la habitación.
* frigobar: un entero que indica si la habitación cuenta con un frigobar (1) o no (0).
* cantBanios: un entero que representa la cantidad de baños disponibles en la habitación.
* aireAcondicionado: un entero que indica si la habitación tiene aire acondicionado (1) o no (0).
* precio: un entero que indica el precio de la habitación.
* opcionPunto6: un entero que representa una opción específica relacionada con la habitación (su significado depende del contexto).
* habilitada: un entero que indica si la habitación está habilitada (1) o no (0).

La estructura **huesped** representa a un huésped y contiene los siguientes campos:

* nombreApellido: una cadena de caracteres que almacena el nombre y apellido del huésped.
* edad: un entero que guarda la edad del huésped.
* genero: un entero que indica el género del huésped (0 para masculino, 1 para femenino, 2 para no especificar).
* telefono: un entero que almacena el número de teléfono del huésped.
* dni: un entero que representa el número de documento de identidad del huésped.
* numeroDeHabitacion: un entero que identifica el número de la habitación asignada al huésped.
* bajaPasiva: un entero que indica si el huésped tiene una baja pasiva (1) o no (0).
* estadoReserva: un entero que indica el estado de la reserva del huésped.
* llegada: una variable de tipo tiempo que registra la fecha y hora de llegada del huésped.
* salida: una variable de tipo tiempo que registra la fecha y hora de salida del huésped.

La estructura **tiempo** representa una fecha y hora específicas y contiene los siguientes campos:

* anio: un entero que indica el año.
* mes: un entero que representa el mes.
* dia: un entero que especifica el día.
* hora: un entero que indica la hora.
* minuto: un entero que representa los minutos.

Se puede dar de alta, baja, modificación y listado de las primeras tres, ya que la última es un manejo interno.

# Sobre las librerías

Las 16 librerías que engloban nuestro proyecto fueron producto de una decisión: modularizar lo mejor posible nuestra aplicación. La industria del software hoy dejo de ser una construcción de servicios monolíticos a lograr una diversificación de microservicios, nosotros apuntamos a los estándares de la industria.

Es por eso por lo que, a cada usuario, mediante un juego de case en los switches, puede elegir cada menú y volver al anterior. Cabe destacar que el sistema posee un funcionamiento a través de usuario e inicio de sesión.

# Problemáticas

Además de tener problemas con el manejo de fechas, tuvimos inconvenientes que pudimos resolver, a continuación, se detallan.

## tiempo.h

En el proceso de desarrollar el sistema de reservas y empleados nos surgió un problema: el manejo de las fechas, ¿qué podíamos implementar para administrar esos datos? entre esas ideas estaba la de utilizar time.h, conocida librería de C, pero fue idea de uno de los ayudantes de cátedra, Nahuel Bianca, quién sugirió hacer nuestra propia time.h, lo cual, si bien era reinventar la rueda, era cómodo porque solo queríamos administrar años/meses/días/horas/minutos y sus respectivas funciones de carga, comprobación y lectura. Esta es otra de las problemáticas que nos surgió: ¿cómo íbamos a hacer todas las comprobaciones? Sabemos que febrero tiene 28 días, otros con 31 y también con 30. Además, el gran desafío fue el cálculo de años bisiestos, Bianca y Bruno Fabrizio también nos ayudaron en tales validaciones.

Esta será publicada en GitHub de forma externalizada.

## Resolución con las pilas

Observamos que pila.h, librería provista por la Universidad Tecnológica Nacional, daba ciertas complicaciones a la hora del manejo de datos con archivos, por lo que la deliberación de cuándo utilizarlas fue algo significante: un hotel trabaja las 24 horas, por lo que no precisamente el sistema se abre y se cierra, puede correr continuamente, es por eso que en el algoritmo diseñamos que el sistema de limpieza se borre periódicamente porque las habitaciones se vuelven a ensuciar: de una problemática de persistencia de pilas obtuvimos un beneficio, que esos datos se borren automáticamente.

Originalmente íbamos a utilizar para botellas, pero no dimos cuenta que los datos que se pueden almacenar son solo enteros, por lo que no nos servía para nuestro propósito. Tampoco era óptimo usar pilas dentro de structs. Otra dificultad fue el muestreo de pilas de mejor forma.

La solución de todo eso fue reunirnos con el equipo y reorganizar qué tipo de dato iba a ir para cada menú, sea primitivo o no.

## Arreglo de palabras

Fue una complicación en la estructura cargar un arreglo de palabras, consultamos con documentación de C97 y ChatGPT, pero las respuestas que nos daban no eran muy seguras, es por eso por lo que para solucionarlo:

## Pull request

Como utilizamos GIT y GitHub al 100% para el manejo en simultáneo y colaborar entre el equipo, tuvimos que aprender cómo solucionar pull request y sus conflictos.

¿Cómo lo solucionamos? A prueba y error, creando conflictos y solucionándolos a mano, hasta que entendimos su funcionamiento y el flujo de trabajo se benefició.

Elaboramos un pequeño tutorial para el resto del equipo y pudimos aprender entre todos.

La conclusión es que GIT hace que todo sea más fácil.

A screenshot of a computer

Description automatically generated

Ilustración 1: Cómo se ve nuestro repo

# Aspectos organizativos

Durante nuestra experiencia de uso de Git y GitHub en el desarrollo del sistema de gestión de hoteles, aprendimos lecciones valiosas sobre cómo manejar conflictos y trabajar con pull requests de manera efectiva. Estas habilidades mejoraron nuestra capacidad de colaborar de mejor método y garantizar la calidad del código en el proyecto.

En primer lugar, adquirimos el hábito de actualizar regularmente nuestro repositorio local con los cambios realizados en el repositorio remoto. Esto nos permitió mantenernos al día con los avances de los demás miembros del equipo y reducir la posibilidad de conflictos. Al mantenernos sincronizados, pudimos evitar problemas futuros y trabajar de manera más cohesionada.

Cuando se presentaban conflictos, utilizamos herramientas como las características de resolución de conflictos incorporadas. Es analizar cuidadosamente los cambios en conflicto y a tomar decisiones basadas en la lógica y los objetivos proyecto. Al resolverlos, mantuvimos una comunicación abierta y colaborativa para garantizar que todas las partes involucradas estuvieran al tanto de las decisiones tomadas. Esta colaboración mejoró nuestra cohesión como equipo y nos permitió resolver los conflictos de manera eficiente.

Para gestionar las tareas y organizar nuestro flujo de trabajo, empleamos Trello, una herramienta colaborativa basada en tableros y listas. Creamos listas que representaban las distintas etapas de desarrollo del proyecto, como "Por hacer", "En progreso" y "Completado". Cada tarea se asignó a un miembro del equipo y se movió de una lista a otra a medida que avanzaba. Esto nos permitió mantenernos organizados y tener una visión clara del progreso del proyecto.

A screenshot of a computer

Description automatically generated with medium confidence

Ilustración 2: Cómo se ve nuestro Trello

Además, utilizamos Discord como plataforma de comunicación en tiempo real. A través de dicha plataforma, nos mantuvimos conectados durante todo el desarrollo del proyecto, compartiendo actualizaciones, resolviendo dudas y discutiendo ideas. La comunicación constante nos permitió tomar decisiones rápidas y abordar cualquier problema o pregunta de manera oportuna. También facilitó la colaboración y la resolución de problemas de manera eficiente, lo que mejoró la cohesión y el rendimiento general del equipo.

# Convenciones y optimizaciones internas

Para poder llevar a cabo de la mejor manera posible, hicimos una suerte de convenciones que nos invocan las funciones de ciertas maneras. Para todo realizamos lo siguiente:

* Géneros: 0 para masculino, 1 para femenino, 2 no específico.
* Clientes: 0 está en el hotel, 1 para no estar en el hotel.

A screenshot of a computer program

Description automatically generated with medium confidenceTambién optimizamos las funciones.

El caso en donde se carga la función primero pasa por parámetro el DNI o cualquier campo a comprobar si está repetido.

Luego, dependiendo el retorno de la función de búsqueda de repetición, (usualmente si es diferente a 1 o -1, si la función no encontró ninguna coincidencia), se le pasa el parámetro del dato a través de los parámetros actuales. Con esto nos ahorramos preguntar de nuevo el DNI dentro de la función.

Pero en el caso de que la función se utilice de otra manera que no sea en el case, por convención, nosotros, los desarrolladores, le damos de parámetro actual 0 para que en un condicional dentro se pueda agregar dicho dato.

## Otras optimizaciones

También elegimos optimizar funciones que hacían lo mismo con otros parámetros, aprovechando los *int* de C, realizamos optimizaciones que nos permiten achicar líneas de código de la mejor manera posible:

En vez de realizar esto (página siguiente):



Realizamos una que haga los casos:

