

Universidad Nacional Autónoma de México
Facultad de Ingeniería

Semestre 2024-1

Sistemas operativos
Grupo 6

Proyecto 2

“Una situación cotidiana paralelizable”

Docente:
Ing. Gunnar Eyal Wolf Iszaevich

Alumnos:
 Ceniceros Mariaca Carlos
 Ramirez Martinez Luis Ángel

La situación para modelar es la siguiente:

En un hotel, hay una serie de operaciones que ocurren simultáneamente y que requieren una gestión eficiente para garantizar una experiencia satisfactoria para los huéspedes. Por lo que la situación a modelar es el sistema de reservas de un hotel.

¿Dónde pueden verse las consecuencias nocivas de la concurrencia?
¿Qué eventos pueden ocurrir que queramos controlar?

Cuando el sistema consulta las habitaciones ocupadas y disponibles, en caso de no manejarse bien podríamos generar reservas duplicadas.

¿Hay eventos concurrentes para los cuales el ordenamiento relativo no resulta importante?

- Registro de nuevas reservas: Esta parte se puede manejar de manera secuencial
- Consultas relacionadas a la disponibilidad: El orden no es crítico por lo que se puede tomar de manera no tan importante mientras la información proporcionada sea actualizada y correcta
- Actualización del estado de la habitación: El orden no es critico por lo que se puede tomar como algo no tan importante mientras no se produzcan conflictos procurando un mecanismo de consultas eficaz

Mecanismo de sincronización empleado:

El código utiliza mecanismos de sincronización en forma de un mutex (mutual exclusion) para garantizar la exclusión mutua y la consistencia en las operaciones compartidas. A continuación, se describen los mecanismos de sincronización:

Mutex (mutex): El mutex es una instancia de `threading.Lock()` que se utiliza para proporcionar exclusión mutua. Los mutex son bloqueos que se pueden adquirir y liberar por hilos para garantizar que solo un hilo pueda acceder a una sección crítica de código a la vez. En este programa, el mutex se utiliza para proteger las operaciones críticas que involucran el diccionario `habitaciones_disponibles`.

with mutex:: La estructura with se utiliza para adquirir y liberar el mutex de manera segura. Cuando un hilo entra en una sección crítica dentro del bloque with mutex, adquiere el mutex y se asegura de que otros hilos esperen su liberación antes de ejecutar la misma sección crítica.

Estos mecanismos de sincronización garantizan que las operaciones de check-in y check-out en el diccionario habitaciones_disponibles se realicen de manera segura en un entorno de múltiples hilos. El mutex permite que solo un hilo acceda a estas operaciones críticas a la vez, evitando conflictos y garantizando la consistencia de los datos compartidos.

La logica de operación es la siguiente:

Identificación del estado compartido (variables o estructuras globales):

En este código, el estado compartido es representado por el diccionario '**habitaciones_disponibles**', que almacena la disponibilidad de las habitaciones del hotel. Cada habitación se identifica por un número (clave) y su estado (valor). Las variables globales '**lista_habitaciones**' (un widget de lista en la interfaz de usuario) y **ocupadas_text** (un widget de texto en la interfaz de usuario) también son compartidas y utilizadas para mostrar la disponibilidad de las habitaciones y las habitaciones ocupadas.

Descripción algorítmica del avance de cada hilo/proceso:

- El programa principal (**'crear_interfaz'**) inicia la interfaz de usuario y permite a los usuarios realizar manualmente operaciones de check-in y check-out.
- El botón "Simular reserva" en la interfaz inicia un hilo (**'simular_reservas'**) que realiza simulaciones automáticas de check-in y check-out de manera aleatoria. El hilo funciona en un bucle infinito y, en cada iteración:
 - Genera un cliente y una habitación aleatoria.
 - Intenta realizar un check-in si la habitación está disponible o un check-out si la habitación está ocupada.
 - Actualiza la interfaz de usuario para reflejar los cambios en la disponibilidad de habitaciones y las habitaciones ocupadas.

- Espera un segundo antes de la siguiente simulación.

Descripción de la interacción entre ellos (mediante los mecanismos de sincronización o de alguna otra manera):

- La interacción entre los hilos se produce en el contexto del diccionario '***habitaciones_disponibles***', que es una estructura de datos compartida.
- El mecanismo de sincronización utilizado es un '***mutex***' que protege las operaciones críticas en el diccionario '***habitaciones_disponibles***'. Los hilos adquieren el mutex antes de realizar operaciones de check-in o check-out en el diccionario y lo liberan después de completar estas operaciones. Esto garantiza que solo un hilo pueda modificar el estado de una habitación a la vez y evita conflictos de concurrencia.
- La interacción entre el hilo de simulación y el hilo principal de la interfaz de usuario se logra mediante el uso del mecanismo de ***mutex***. El hilo de simulación accede al diccionario '***habitaciones_disponibles***' de manera segura gracias al ***mutex***.
- La actualización de la interfaz de usuario, como la lista de habitaciones disponibles y el texto de las habitaciones ocupadas, se realiza para reflejar los cambios en el estado compartido. La sincronización de esta actualización es manejada por el entorno de la interfaz gráfica de usuario (GUI) subyacente, que generalmente es capaz de manejar eventos y actualizaciones en entornos multi-hilo.

Descripción del entorno de desarrollo, suficiente para reproducir una ejecución exitosa

El código está escrito en Python, un lenguaje de programación de alto nivel. En el código se utiliza Python 3, ya que la sintaxis y las bibliotecas específicas que se emplean son compatibles con Python 3.

Además de las bibliotecas estándar de Python, el código utiliza la biblioteca tkinter para crear la interfaz gráfica de usuario (GUI) y threading para la gestión de hilos.

En cuanto al sistema operativo o distribución, Python es un lenguaje multiplataforma, lo que significa que puede ejecutarse en varios sistemas operativos, incluyendo Windows, macOS y diversas distribuciones de Linux. El código se puede desarrollar y probar en una amplia gama de sistemas operativos.

Pruebas del funcionamiento exitoso:

Para iniciar, al ejecutar el código, la interfaz luce así:

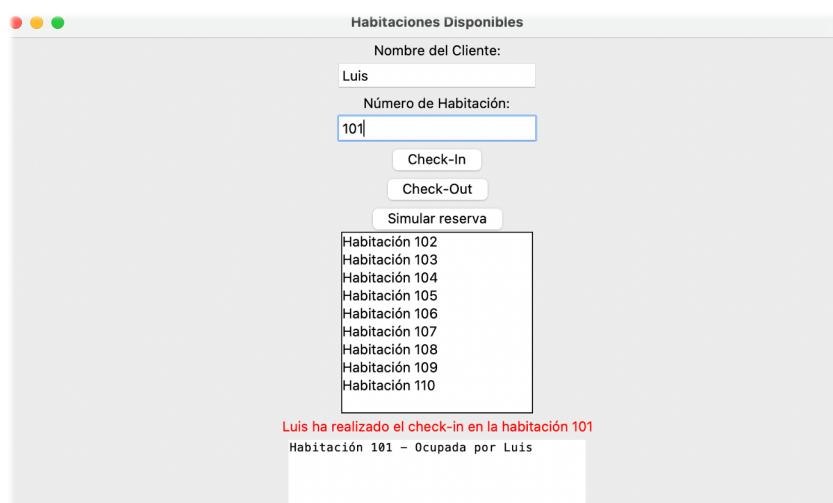


Para ver el funcionamiento del programa, como se menciono antes, existe una forma “manual” de hacerlo y ver como funciona en donde tu vas agregando el nombre del cliente y la habitacion.

En el cuadro donde se despliegan el numero de habitaciones, indica cuales son las habitaciones disponibles, entonces colocas el nombre del cliente, numero de habitacion que quieres y presionas el boton “Check-in” y se muestra

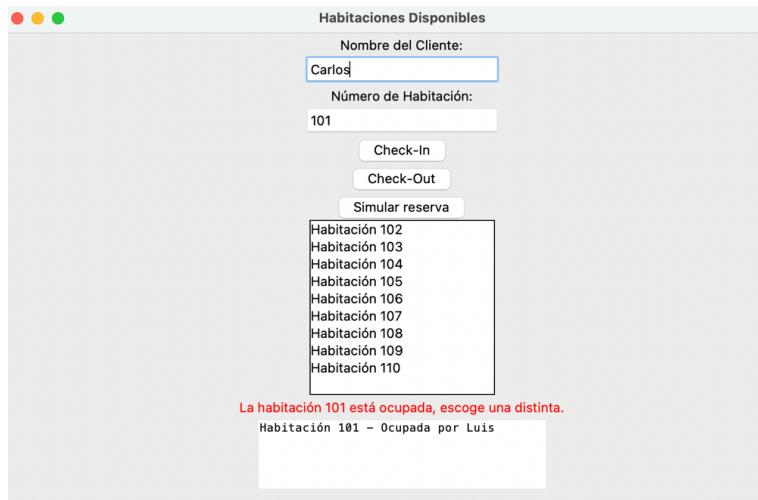
lo

siguiente:



Así funciona para cada habitación, incluso en el código se pueden agregar más habitaciones.

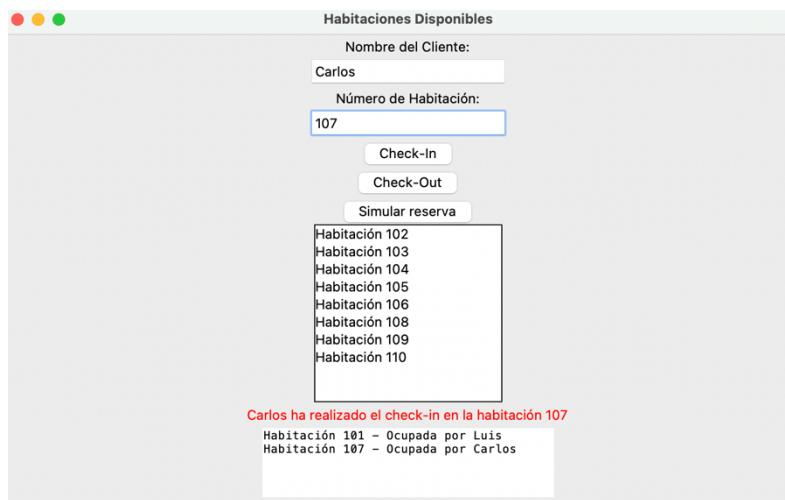
Puedes agregar otro nombre, pero si intentas colocar la misma habitación da la siguiente notificación (en rojo):



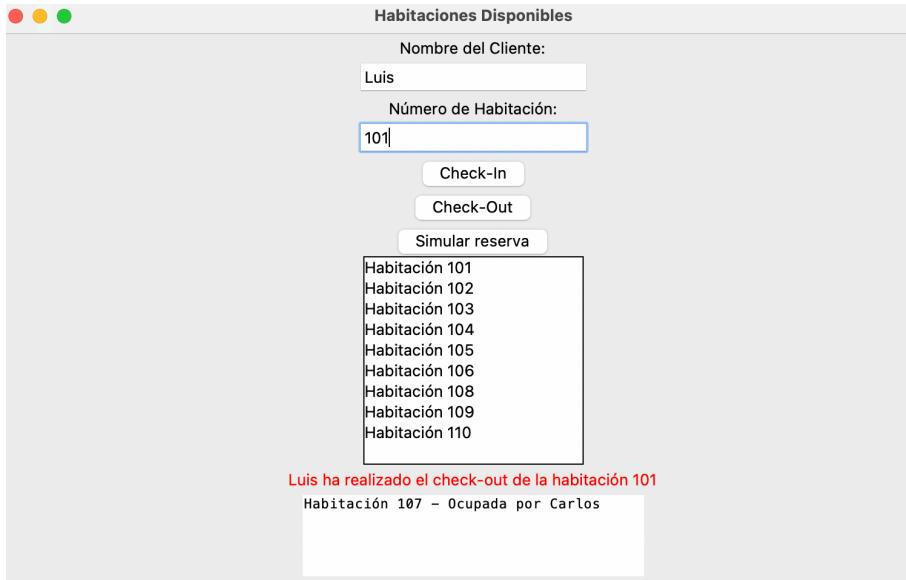
La habitación 101 está ocupada, escoge una distinta.

Podemos apreciar que, la habitación 101 desaparece de la lista de habitaciones disponibles, además en el cuadro de texto de abajo indica las habitaciones ocupadas, en donde se muestra el número de habitación y el nombre de la persona que lo está ocupando.

A continuación, muestro que pasa si asigno un nuevo cliente a otra habitación, con el fin de demostrar que sirve para cualquier habitación disponible:



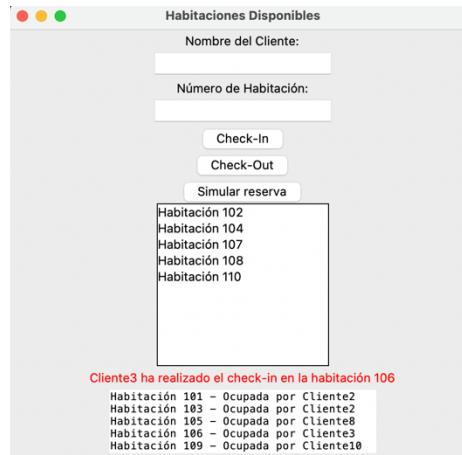
Ahora, si volvemos a indicar el cliente junto con la habitacion que tiene asignada y damos click al boton “Check-out”, la habitacion pasa a estar disponible y desaparece de la lista de ocupados, y por consecuente, la habitacion pasa a la lista de disponibles.



Podemos observar en la imagen, que ademas de hacer lo ya antes mencionado, tambien muestra la notificacion (en rojo), de que el cliente, en este caso Luis, ha hecho el “Check-out”.

Por otro lado, podemos hacer que las reservas se hagan, por asi decirlo, solas, infinitamente hasta que se decida parar cerrando el programa. Para esto solo tenemos que apretar el boton que dice “Simular reserva”

A continuacion, pantallazos de esta simulación:



Habitaciones Disponibles

Nombre del Cliente:

Número de Habitación:

Check-In

Check-Out

Simular reserva

Habitación 102
Habitación 104
Habitación 106
Habitación 107
Habitación 108
Habitación 110

Cliente3 ha realizado el check-out de la habitación 106

Habitación 101 - Ocupada por Cliente2
Habitación 103 - Ocupada por Cliente2
Habitación 105 - Ocupada por Cliente8
Habitación 109 - Ocupada por Cliente10

Habitaciones Disponibles

Nombre del Cliente:

Número de Habitación:

Check-In

Check-Out

Simular reserva

Habitación 101
Habitación 102
Habitación 104
Habitación 106
Habitación 107
Habitación 108

Cliente8 ha realizado el check-in en la habitación 110

Habitación 103 - Ocupada por Cliente2
Habitación 105 - Ocupada por Cliente8
Habitación 109 - Ocupada por Cliente10
Habitación 110 - Ocupada por Cliente8

Habitaciones Disponibles

Nombre del Cliente:

Número de Habitación:

Check-In

Check-Out

Simular reserva

Habitación 101
Habitación 102
Habitación 104
Habitación 106
Habitación 107
Habitación 108
Habitación 110

Cliente8 ha realizado el check-out de la habitación 110

Habitación 103 - Ocupada por Cliente2
Habitación 105 - Ocupada por Cliente8
Habitación 109 - Ocupada por Cliente10