GENERALITAT VALENCIANA
Conselleria d'Educació, Cultura i Esport

ceedcv CENTRE ESPECÍFIC D'EDUCACIÓ A DISTÀNCIA DE LA COMUNITAT VALENCIANA

2ª Evaluación

DESCRIPCIÓN DE LOS PROYECTOS

CONTENIDO

Licencia

1. Listado de proyectos

1.1. Proyecto 1: Sistema de Control de Calidad en Fábrica

1.2. Proyecto 2: Sistema de Control de Iluminación en Fábrica

1.3. Proyecto 3: Sistema de Gestión de Inventario

1.4. Proyecto 4: Gestión de Mantenimiento Correctivo en Maquinaria Pesada

1.5. Proyecto 5: Control de Acceso a Áreas Críticas con Usuario y Contraseña

1.6. Proyecto 6: Monitoreo de Transporte con Control de Parámetros de Seguridad

1.7. Proyecto 7: Gestión de Tiempos y Productividad en Estaciones de Trabajo

1.8. Proyecto 8: Simulador de Algoritmos en tiempo real

1.9. Proyecto 9: Controlador de vuelo

1.10. Proyecto 10: Planificación de viajes

1.11. Proyecto 11: Cartera de inversiones

1.12. Proyecto 12: Gestor de Reservas de un Hotel

1.13. Proyecto 13: Simulador de Panadería

1.14. Proyecto 14: Proyecto genérico

¿Qué tengo que hacer?

- Programar la aplicación.
- Primero se implementa en local.
- La programación debe ser modular. Busca información al respecto, lo que se pide es:
- El programa main debe ser muy simple y con poco código
- Todo lo que puedas hacer con funciones, hazlo con funciones
- Crea un menú, en que el usuario tiene varias opciones.
- Guarda la información necesaria de la aplicación.
- Se debe notificar cada uno de los cambios.
- Se deben tener en cuenta las posibles excepciones que se puedan producir.
- Permite al usuario moverse entre los diferentes menús de la aplicación.
- Se valorará que se añadan funcionalidades extra a las citadas en el documento.

2. Requisitos de los proyectos

- La creación de clientes se realizará mediante la herencia de la clase `threading.Thread`.
- Los servidores serán los únicos encargados de realizar cálculos, guardar información, etc. No está permitido el uso de variables globales para almacenar datos.
- El servidor debe ser capaz de atender a varios usuarios de forma simultánea, gestionando la concurrencia mediante semáforos o `locks`.
- Será posible configurar la cantidad de servidores necesarios, utilizando programación multihilo.
- Se debe implementarse un menú inicial para el servidor que permita cambiar las opciones iniciales, seleccionar la cantidad deseada de servidores, iniciar y detener el servidor.
- Es obligatorio manejar todas las posibles excepciones para evitar fallos en la ejecución.
- El código del servidor y del cliente debe estar organizado en archivos Python separados (un archivo para el servidor y otro para el cliente).
- La comunicación entre el cliente y el servidor se realizará a través de **sockets.**
- Debe implementarse un control de errores para manejar situaciones como la introducción de opciones inválidas por parte del usuario. En estos casos, se debe informar al usuario del error y darle otra oportunidad para corregirlo.
- La comunicación entre el cliente y el servidor debe utilizar **encriptación híbrida.** Esto implica el uso de claves RSA para el intercambio de claves y encriptación simétrica para proteger los mensajes.
- Se valorará utilizar librerías para realizar notificaciones push a iOS y Android.
- Documentación del proyecto, será necesario explicar todas las suposiciones que se realizan para este proyecto, la explicación de las simulaciones llevadas a cabo, de las distintas respuestas obtenidas y los flujos tanto de información como de funciones implementadas.

3. Descripción proyectos

3.1. Proyecto 1: Sistema de Control de Calidad en Fábrica

- Desarrolla un servidor central que reciba datos en tiempo real de varios clientes, cada uno representando una máquina en una línea de producción.
- El servidor debe ejecutar los cálculos mediante programación asíncrona o multihilo.
- Cada cliente envía lecturas sobre el estado y calidad de los productos (temperatura, peso y humedad).
- La comunicación con el servidor debe ser encriptada.
- Cálculo del índice de calidad del producto:

La calidad del producto (Q) se calcula como:

$$Q = 1 - \frac{w_1 \cdot D_1 + w_2 \cdot D_2 + w_3 \cdot D_3}{3}$$

Donde:

- Q: La calidad del producto, evaluada entre **0** (calidad inaceptable) y **1** (calidad perfecta).
- w_1, w_2, w_3: **Pesos relativos** asignados a los parámetros P_1, P_2, P_3, indicando la **importancia** de cada uno en la evaluación de calidad.
  - Los valores de w deben cumplir que:

$$w_1 + w_2 + w_3 = 1$$

- D_i: El **desvío normalizado** del parámetro P_i, que mide cuánto se aleja el valor medido (P_i) de su valor ideal (R_{ideal}):

$$D_i = \max\left(0, \frac{|P_i - R_{\text{ideal}}|}{R_{\text{tolerancia}}}\right)$$

  Donde:

  - P_i: Valor medido del parámetro i (por ejemplo, temperatura, peso, humedad, etc.).
  - R_{ideal}: Valor ideal del parámetro i.
  - $R_{\text{tolerancia}}$: Rango de tolerancia aceptable para el parámetro i, calculado como:

$$R_{\text{tolerancia}} = R_{\text{max}} - R_{\text{min}}$$

EJEMPLO

$$D_i = \frac{|P_i - R_{\text{ideal}}|}{R_{\text{tolerancia}}}$$

1. **Temperatura**:

$$D_1 = \frac{|22 - 20|}{5} = \frac{2}{5} = 0.4$$

2. **Peso**:

$$D_2 = \frac{|52 - 50|}{10} = \frac{2}{10} = 0.2$$

3. **Humedad**:

$$D_3 = \frac{|60 - 50|}{15} = \frac{10}{15} = 0.67$$

1. **Temperatura**:

$$w_1 \cdot D_1 = 0.4 \cdot 0.4 = 0.16$$

2. **Peso**:

$$w_2 \cdot D_2 = 0.3 \cdot 0.2 = 0.06$$

3. **Humedad**:

$$w_3 \cdot D_3 = 0.3 \cdot 0.67 = 0.201$$

La calidad se calcula como:

$$Q = 1 - \frac{w_1 \cdot D_1 + w_2 \cdot D_2 + w_3 \cdot D_3}{3}$$

Sustituyendo:

$$Q = 1 - \frac{0.16 + 0.06 + 0.201}{3}$$

$$Q = 1 - \frac{0.421}{3} = 1 - 0.140 = 0.86$$

- Si alguna máquina detecta valores de calidad fuera de los límites, el servidor envía alertas a los operarios.
- Será posible cambiar los datos ideales, las tolerancias, los pesos relativos y los límites del índice de calidad del producto.

- En cualquier momento será posible detener el servidor, el sistema mostrará los datos más relevantes como las alarmas y datos estadísticos de la jornada como la tasa de defectos.

3.2. Proyecto 2: Sistema de Control de Iluminación en Fábrica

- Desarrolla un servidor central que gestione el encendido y apagado de luces en diferentes zonas de una fábrica.
- Cada cliente representa una zona de iluminación y envía datos encriptados como ocupación, hora y grado de apertura de las ventanas cada cierto tiempo.
- El servidor analiza los datos, puedes utilizar librerías como astral para obtener las horas de amanecer y atardecer, realiza los cálculos necesarios mediante una fórmula que tendrás que diseñar para ajustar el sistema de iluminación y envía el porcentaje de luz que debe aportar el sistema, de 0% a 100%.
- Los clientes pueden solicitar aumentar o disminuir el valor de la iluminación en un 20%.
- Debe existir la posibilidad de cambiar datos iniciales como la localización y constantes de la fórmula creada para este sistema en el servidor.
- En cualquier momento es posible solicitar al servidor un informe con todos los cambios y a la hora que se han producido.
- Graficar los datos de iluminación a lo largo del día con la librería matplotlib.

3.3. Proyecto 3: Sistema de Gestión de Inventario

- Desarrolla un servidor que administre un inventario central, existen diferentes puntos de control en la cadena de suministro (almacén primario, almacén intermedio y tienda). Cada punto de control ejecutará los cálculos en un hilo distinto del servidor.
- Los clientes envían solicitudes encriptadas para extraer unidades de la tienda, se puede simular con una función aleatoria para comprar o no y esperas. Supondremos que cada 30 segundos pasa un día en la tienda.
- A medida que los clientes compran y disminuya el stock, alcanzaremos un nivel crítico, en ese momento se realizará una solicitud al punto de control anterior en la cadena, por ejemplo si falta producto en la tienda, se enviará una señal al almacén intermedio y este enviará X veces la cantidad de ese producto que se vende al día. Estos productos se extraerán del inventario del almacén intermedio y al día siguiente (después de 30 segundos) se sumarán al inventario de la tienda. Puedes elegir el valor de X.
- Se deberá mantener un stock de seguridad en los dos almacenes.

- Si el stock cae por debajo de un nivel crítico, el servidor envía alertas para reabastecimiento. El nivel equivale a los días que faltan para llegar al stock de seguridad si no llegan nuevos productos. Se calculará restando la cantidad actual al stock de seguridad y dividiendo por los productos que se venden al día de esa referencia.
- El servidor actualiza el inventario en tiempo real y envía confirmación de las transacciones.
- En el servidor se pueden crear o eliminar referencias de productos.
- Se puede solicitar un resumen del inventario y las transacciones registradas al servidor. También será posible cambiar datos iniciales como la cantidad inicial de productos en los puntos de control, la cantidad de stock de seguridad y el valor de X.
- Utilizar librerías para graficar la cantidad de materiales inventariados con un gráfico de tarta.

3.4. Proyecto 4: Gestión de Mantenimiento Correctivo en Maquinaria Pesada

- **Datos de las Máquinas (Clientes)**
    - Cada cliente (máquina) envía, cada 5 segundos:
        - **Horas de operación acumuladas desde la última intervención y total.**
        - **Consumo energético promedio (en kWh). Un valor aleatorio entre 5 y 7,5 kWh por cada 5 segundos.**
        - **Errores detectados:** Gravedad del fallo (encriptado). Los errores se generarán aleatoriamente cada 40 segundos con un 25% de probabilidad de que ocurran.
        - **Estado general:** Operativa (ON), Detenida (OFF) mientras se realiza el mantenimiento, o En Fallo (ERROR).
- El servidor
- **Gestión de Tickets (Servidor)**
    - El servidor generará:
        - **Tickets Correctivos:** Cuando detecte fallos críticos (estado `ERROR` o consumo fuera de rango).
        - **Tickets Preventivos:** Cada 320 segundos de operación o cuando el consumo energético supere 400kWh. Solo cuando se ejecute la intervención preventiva, estos contadores vuelven a empezar de 0.

- Cada ticket incluye:
    - ID de la máquina, prioridad (baja, media, alta), tipo (correctivo/preventivo), tiempo estimado de intervención y técnico asignado.

- **Priorización de Tickets**

    - La prioridad se calcula según:
        - **Gravedad del fallo:** Crítico > Moderado > Leve. Si se trata de un ticket preventivo G=0.
        - **Tiempo desde la última intervención:** Máquinas sin mantenimiento reciente tienen mayor prioridad.

Fórmula de Prioridad

$$P = G \cdot W_g + T \cdot W_t$$

Donde:

- P: **Prioridad** del ticket (valor calculado, mayor significa más urgente).
- G: **Gravedad del fallo**, evaluada como un valor numérico según la severidad (ejemplo: 1 = leve, 2 = moderado, 3 = crítico).
- T: **Tiempo desde la última intervención**, medido en horas (o segundos si se trabaja en tiempo simulado).
- W_g: **Peso relativo de la gravedad**, define la importancia de la gravedad en la fórmula (ejemplo: 0.7).
- W_t: **Peso relativo del tiempo acumulado**, define la importancia del tiempo desde la última intervención (ejemplo: 0.3).

Criterios para Ajustar los Pesos

- Si el mantenimiento correctivo es más prioritario que el preventivo, asigna $W_g > W_t$ (por ejemplo, $W_g = 0.7, W_t = 0.3$).
- Para sistemas más preventivos, incrementa W_t respecto a W_g.

- **Gestión de Técnicos**

    - Dos técnicos disponibles:

    - **Intervenciones:** Cada intervención dura entre 20 y 40 segundos (simulado).
    - **Horarios:** Almuerzo (10 segundos) y comida (20 segundos). La jornada de la fábrica serán 160 segundos.
    - **Asignación:** Se crea una cola basada en la prioridad de los tickets.
- **Alertas y Notificaciones**
    - Las máquinas y los técnicos reciben notificaciones en tiempo real cuando un técnico es asignado.
- **Configuración Dinámica**
    - Desde el servidor se podrán ajustar:
        - Límites críticos (ej. consumo máximo, horas entre mantenimientos).
        - Frecuencia de envío de datos de las máquinas.
- **Informes Finales**
    - **Informe de Máquinas:** Número de intervenciones, fallos detectados y tiempo sin estar operativa.
- **Criterios de Finalización**
    - El servidor se puede detener cuando se solicite, guardando los datos históricos en un archivo.

3.5. Proyecto 5: Control de Acceso a Áreas Críticas con Usuario y Contraseña

1. **Gestión de Accesos (Servidor)**
    - **Recepción de Credenciales:** El servidor recibe credenciales (usuario y contraseña) encriptadas.
    - **Validación:** Comprueba las credenciales contra una base de datos local que incluye:
        - ID de usuario.

        - Contraseña hash.
        - Áreas autorizadas.
        - Horarios permitidos.
        - Si el usuario ha accedido y no ha salido.
    - **Respuesta:** El servidor responde con:
        - `ACCESO PERMITIDO` si las credenciales y permisos son válidos.
        - `ACCESO DENEGADO` si las credenciales son incorrectas o el área no está autorizada.

2. **Reintentos Fallidos y Bloqueo General**

    - Cada usuario tiene un máximo de **3 intentos fallidos consecutivos**. Si excede este límite:
        - **Acciones inmediatas:**
            - Bloqueo de todas las áreas críticas.
            - Generación de una alerta automática al sistema de seguridad. Se ordenarán las alertas por nivel de seguridad del área.
        - **Desbloqueo:** Solo se puede realizar con una **contraseña maestra**, registrada en un archivo de configuración seguro.

3. **Gestión de Áreas**

    - Cada área tiene configuraciones específicas:
        - Capacidad máxima de usuarios simultáneos.
        - Reglas de horario (por ejemplo, acceso permitido solo de 8:00 a 18:00). Se recomienda utilizar un sistema simulado para realizar pruebas, que genere usuarios cada cierto segundos que accedan a las áreas simulando los posibles accesos permitidos y denegados. Un día se puede calcular como una cantidad de segundos y obtener el resto de tiempos a partir de esa equivalencia.
    - Si se supera el límite de usuarios simultáneos, el sistema denegará nuevos accesos hasta que alguien salga.

4. **Registro de Actividades**

    - Cada acceso permitido se registra con:
        - ID de usuario.
        - Área.

    - Hora de ingreso y salida (calculada cuando el usuario abandona el área).
- Los accesos denegados se registran con:
    - Motivo del rechazo (credenciales incorrectas, área no autorizada, horario restringido, etc.).

5. **Configuración Dinámica**

- El servidor debe permitir:
    - Agregar, eliminar o modificar usuarios y permisos.
    - Configurar horarios y capacidad máxima por área.
    - Cambiar la contraseña maestra y las políticas de seguridad (por ejemplo, límite de reintentos).

6. **Informes**

- Al final del día, se generan dos informes:
    - **Informe de Accesos:** Listado de usuarios, áreas, tiempos de ingreso y salida.
    - **Informe de Rechazos:** Motivo, hora y usuario asociado.

3.6. Proyecto 6: Monitoreo de Transporte con Control de Parámetros de Seguridad

1. **Recepción de Demandas**

- Los centros de distribución envían solicitudes en tiempo real al servidor indicando:
    - Producto(s) requerido(s).
    - Cantidad necesaria.
    - Urgencia de la entrega (valor de prioridad asignado por el centro).
    - Plazo máximo para completar el envío.
- Cada solicitud queda registrada en una **cola de necesidades** asociada al centro de distribución.
- Se simularán las solicitudes aleatorias por los centros cada 30 segundos, con una urgencia y datos aleatorios, respecto a los productos disponibles, puedes crear una tabla con productos.

2. **Asignación de Camiones**

    - Los camiones son monitoreados por el servidor y reportan:
        - Centro de distribución de destino o actual y tiempo restante de ruta.
        - Capacidad de carga disponible.
        - Estado (disponible o en ruta).
    - El servidor selecciona un camión, minimizando el tiempo de llegada al centro de distribución, puede elegir uno ocupado, si se encuentra más cerca, teniendo en cuenta que el tiempo de descarga es de 10 segundos.
    - Se creará una cola con las necesidades.

3. **Cálculo de Prioridades**

    - Las solicitudes en la cola de necesidades se ordenan según una fórmula que evalúa:
        - **Urgencia (U)**: Grado de prioridad indicado por el centro (1 a 3).
        - **Plazo restante (Tr)**: Tiempo hasta el vencimiento del plazo, en segundos.
    - Fórmula de prioridad para cada solicitud: $P=(U \cdot Wu)+(Tr+11 \cdot Wt)$ Donde Wu y Wt son pesos ajustables que determinan la importancia relativa de cada factor. Por ejemplo Wu=0,7 y Wt= 0,3.

4. **Gestión de Rutas**

    - El servidor mantiene una tabla con las distancias entre los diferentes centros de distribución.
    - La selección de camión y ruta se realiza para minimizar el tiempo de llegada.
    - Ejemplo de tabla de distancias en segundos:

Centro	C1	C2	C3
C1	0	120	80
C2	120	0	60
C3	80	60	0

5. **Estados de Camiones**

    - Los camiones pasan por diferentes estados:
        - **Disponible:** Sin asignación de viaje.

    - **En ruta:** Transportando mercancías a un centro.
    - **Descargando:** Descargando en un centro.

6. **Planificación Dinámica**

    - Si una nueva solicitud con alta prioridad ingresa al sistema, el servidor puede:
        - Solo modificar la cola de necesidades sin asignación.

7. **Registro y Estadísticas**

    - Al final de la jornada, el servidor genera un informe con:
        - Lista de solicitudes atendidas, con tiempos de envío y recepción.
    - Estadísticas de eficiencia:
        - Promedio de tiempo de respuesta.

8. **Configuración y Adaptabilidad**

    - Se puede configurar dinámicamente:
        - Pesos de la fórmula de prioridad (Wu y Wt).
        - Distancias entre centros.

3.7. Proyecto 7: Gestión de Tiempos y Productividad en Estaciones de Trabajo

Descripción del proyecto

Diseñar un servidor central que administre y coordine las tareas generadas por **8 clientes** en estaciones de trabajo, utilizando un algoritmo **Round Robin** con un Quantum de 10 minutos. El sistema garantizará el procesamiento eficiente de las tareas, evaluará los tiempos de espera y productividad, y generará informes al final de la jornada.

Requisitos del Sistema

1. **Creación de Tareas**

    - Cada cliente representa una **estación de trabajo** que, durante la jornada, genera **9 tareas** con tiempos de ejecución aleatorios entre **10 y 30 minutos**. Las tareas se deben crear durante toda la jornada laboral, por ejemplo, un tiempo aleatorio entre 2h y 2,5h.
    - El servidor gestiona **una única máquina** para procesar las tareas.
    - Las tareas se envían de manera **encriptada** al servidor.

2. **Planificación de Tareas**

    - El servidor organiza las tareas en una **cola global** según el algoritmo **Round Robin**.
    - Cada tarea recibe un **Quantum** de 10 minutos:
        - Si no se completa en ese tiempo, vuelve al final de la cola con su tiempo restante actualizado.
        - Si se completa, se elimina de la cola.
    - Las tareas se procesan en el orden recibido, pero se permite un **máximo de 2 tareas prioritarias** por cliente, que interrumpen el flujo normal y se procesan mediante otra cola de tareas prioritarias, ordenada mediante **Shortest Job First**, que se ejecutará cada dos tareas normales.

3. **Nuevos Parámetros**

    - **Prioridades de Tareas:** Cada cliente puede marcar hasta **2 tareas prioritarias**. Estas tendrán preferencia en el procesamiento y se intercalan cada 2 tareas normales.
    - **Tiempo límite:** El servidor simula una jornada de trabajo de 24 horas. Las tareas no completadas en este tiempo se consideran "abandonadas" y se registran.

4. **Cálculos del Servidor**

    - **Tiempo de espera (Tespera):** Tiempo transcurrido desde que una tarea entra en la cola hasta que se completa.
    - **Tiempo de espera promedio (Tmedio):** Promedio del tiempo de espera de todas las tareas completadas.

5. **Informe Detallado**

    - Al final de la jornada, el sistema genera un informe con:
        - Número total de tareas completadas, abandonadas y prioritarias.
        - **Tiempo promedio de espera** por cliente.

6. **Flexibilidad del Sistema**

    - El servidor permite configurar:
        - **Quantum:** Modificar su duración para simular diferentes escenarios.
        - **Prioridades:** Ajustar el número máximo de tareas prioritarias por cliente.
        - **Simulación:** Multiplicar todos los tiempos por un factor constante igual en todos, para simular todo el sistema en pocos minutos, en vez de 24 horas.
    - Posibilidad de aumentar el número de máquinas para procesar tareas en paralelo, convirtiendo el modelo en un sistema **multinúcleo**. Cada máquina tendrá su propia subcola Round Robin.

7. **Reinicio del Sistema**

    - Al finalizar la jornada, se podrá reiniciar para una nueva sesión, vaciando la cola y restableciendo las estadísticas.

3.8. Proyecto 8: Simulador de Algoritmos en tiempo real

- Desarrolla un servidor central que actúe como gestor de simulación para los algoritmos vistos en clase (FSFC, SJF, SRTN, Round Robin).
- Cada cliente creará tareas a tiempo real con un tiempo aleatorio de llegada y duración.
- Las comunicaciones se realizarán de forma cifrada.
- El servidor procesa las solicitudes de forma concurrente utilizando cada uno de los algoritmos y devuelve los pasos detallados de ejecución, permitiendo a los estudiantes ver cómo cambia el conjunto de datos en cada etapa del algoritmo.
- El servidor envía también estadísticas en tiempo real, como el estado de las colas y los intercambios realizados, para visualizar el rendimiento de cada algoritmo.
- Al finalizar la simulación de un algoritmo, el sistema genera un informe que incluye el tiempo total de ejecución, el tiempo de espera y el tiempo de espera medio de cada algoritmo.
- Permite reiniciar el sistema para probar con nuevos conjuntos de datos, facilitando la comparación entre ellos.
- Es interesante que se puedan modificar datos iniciales, desde el menú inicial, cómo la duración mínima y máxima de las tareas, la cantidad de tareas que envía cada cliente, la cantidad de clientes, el intervalo con el que se producen las tareas en los clientes, etc.

3.9. Proyecto 9: Controlador de vuelo

- **Descripción del Servidor Central**: Desarrolla un servidor central que actúe como controlador de vuelo, priorizando tareas esenciales para la estabilidad y seguridad de un vuelo simulado.
- **Clientes Representando Sensores y Sistemas de Control**: Cada cliente simula un sistema del avión, como el sensor de combustible, sistemas de control como el estabilizador y sistema de temperatura interior del avión. Los clientes envían datos encriptados al servidor en intervalos regulares. La estabilidad vendrá dada por tres sensores distintos, lo que significa que tendremos 3 clientes distintos con datos sobre el estabilizador, se evaluarán las tres medidas solo cuando lleguen las tres medidas, si no llegan las 3 se activará una alarma.
- Cada cliente enviará datos aleatorios, según un máximo y mínimo. Los datos se enviarán siempre de forma cifrada.
- **Algoritmo de Priorización**: Implementa el algoritmo de prioridades en el servidor que evalúe y gestione cada tarea según su importancia. Las tareas se simularán mediante una espera personalizada para cada una de ellas. Características del algoritmo:
  - Las tareas críticas de estabilidad (sistema de estabilización) reciben la prioridad más alta y se ejecutan en el momento que aparecen.
    - El sistema de estabilidad crea una tarea cuando se supera una desviación total según la siguiente fórmula:

Fórmula de Desviación Total para Tarea de Estabilización

$$D = \sum_{i=1}^{3} |V_i - V_{\text{ideal}_i}|$$

Condición de Alarma:

$$A = \begin{cases} 1 & \text{si } D > T_{\text{umbral}} \\ 0 & \text{en caso contrario} \end{cases}$$

Variables:

- D: Desviación total de los tres sensores.
- V_1, V_2, V_3: Valores actuales de los sensores 1, 2 y 3 (estabilizadores).
- $V_{\text{ideal}_1}, V_{\text{ideal}_2}, V_{\text{ideal}_3}$: Valores ideales de los sensores 1, 2 y 3.
- T_{umbral}: Umbral de desviación total para activar la alarma.

- Las tareas de monitoreo (como el nivel de combustible) reciben una prioridad media.
- Las tareas de conveniencia (temperatura interna del avión) tienen menor prioridad.

Fórmula de Priorización

$$P = W + \frac{T_e}{K}$$

- P: Prioridad de la tarea
- W: Peso de la tarea (Media: 2, Baja: 1)
- T_e: Tiempo de espera acumulado para la tarea
- K: Constante de ajuste para el impacto del tiempo de espera

Explicación:

1. **Peso W**:
   - Tareas **medias** ($W = 2$): Se les da un peso intermedio.
   - Tareas **bajas** ($W = 1$): Se les da un peso mínimo.
2. **Tiempo de Espera (T_e)**:
   - El tiempo de espera acumulado T_e se ajusta mediante la constante K, que modula su impacto en la prioridad.

- **Respuestas del Servidor y Acciones Prioritarias**: El servidor debe enviar respuestas a cada cliente, indicando el estado de la tarea, esperando, ejecutando o finalizada.
  - Si el sistema recibe una solicitud de ajuste de estabilidad mientras procesa una tarea de media o baja prioridad, interrumpe la tarea de menor prioridad y asigna los recursos a la tarea crítica.
- **Sistema de Alarmas Personalizadas**: Debes establecer unos valores en los cuales el sistema entra en alarma. El servidor debe generar alarmas en situaciones específicas, como:
  - Sistema de estabilización cuando no llega alguna de las 3 medidas necesarias.
  - Combustible bajo.
- **Informe de Vuelo**: Al final de cada sesión de vuelo, el servidor genera un informe detallado, incluyendo todas las tareas procesadas, alarmas activadas y las prioridades de cada acción.
- **Reinicio del Sistema**: El sistema permite reiniciar para simular un nuevo vuelo con diferentes condiciones, cambiando los máximos y los mínimos y los intervalos de las alarmas, y así comparar cómo afecta la priorización en diferentes escenarios.

3.10. Proyecto 10: Planificación de viajes

- **Descripción del Servidor Central**: Diseña un servidor central que actúe como gestor de planificación de viajes, optimizando rutas y tiempos de viaje de diferentes clientes (usuarios) con base en algoritmos de optimización y planificación.
- **Clientes como Usuarios de Viaje**: Los clientes pueden crear y modificar viajes creados por ellos o por otros clientes. Los viajes incluyen parámetros como origen, destino, horario deseado y preferencias de transporte (por ejemplo, tren, autobús, avión). Las comunicaciones entre clientes y el servidor deben estar cifradas.
- **Procesamiento y Respuesta del Servidor**:
  - El servidor recibe las solicitudes de cada cliente y analiza según una tabla que tiene todos los viajes con los diferentes tipos de transporte, el coste y los horarios. Un viaje puede comprender varios tramos de viajes de la tabla.
  - El servidor responde con los detalles del itinerario optimizado para cada cliente, incluyendo rutas, tiempos estimados de viaje, costo total y posibles alternativas.
  - El cliente elige uno o varios itinerarios y empieza la simulación de los viajes en otros hilos distintos, únicamente para estos viajes.
- **Estadísticas en Tiempo Real durante la simulación de los viajes**: Durante la simulación, los clientes pueden cambiar sus solicitudes y el servidor proporciona estadísticas en tiempo real, como:
  - Tiempo de espera para cada cliente.
  - El costo promedio de cada viaje. Existe un coste extra por cambios durante el viaje.
- **Simulación:** Los tiempos de la simulación serán los correspondientes a la tabla, debes utilizar viajes en torno a los 30 a 180 segundos.
- **Informe Final de Simulación**: Al finalizar cada simulación de planificación de viaje, el servidor genera un informe que incluye:
  - La ruta elegida, el tiempo total de viaje, y el costo total.
  - Comparaciones de las distintas opciones de ruta según el algoritmo utilizado (por ejemplo, tiempo más corto vs. costo más bajo).
- **Configuración Inicial en el Menú**:
  - Permite modificar datos iniciales, la tabla de tiempos, tipos de transporte y coste.
- **Funcionalidad de Reinicio y Nuevos Datos**: Permite reiniciar el sistema para simular nuevos escenarios de planificación con diferentes rutas, horarios y costos de transporte, facilitando la comparación de la efectividad de cada algoritmo en distintas situaciones.

3.11. Proyecto 11: Cartera de inversiones

- **Servidor Central como Gestor de Inversiones**: Crea un servidor central que funcione como un gestor de inversiones, capaz de recibir y almacenar las órdenes de compra y venta de diferentes clientes (usuarios). El servidor se encarga de mantener actualizados los datos de la cartera y simular las ganancias o pérdidas de cada usuario.
- **Clientes como Inversores**: Cada cliente representa un inversor que envía solicitudes cifradas al servidor para realizar operaciones de compra, venta o consulta de valores específicos (por ejemplo, acciones, bonos o fondos de inversión).
- **Simulación de Datos del Mercado**: El servidor genera cambios en el valor de los activos a intervalos regulares para simular las fluctuaciones del mercado. Estas variaciones se envían a los clientes en tiempo real para que puedan decidir si comprar, vender o mantener sus inversiones. Simulación de fluctuaciones:
  1. Las acciones estarán relacionadas con un mercado y un sector.
  2. Los mercados y los sectores tendrán una fluctuación aleatoria positiva o negativa, esto afectará directamente a las acciones según la siguiente fórmula:

Fórmula final ajustada:

$$V_{t+1} = V_t \times (1 + \mu_{\text{mercado}} + \mu_{\text{sector}} + \epsilon)$$

Donde:

- V_t es el valor del activo en el tiempo t.
- μ_{mercado} es la tendencia general del mercado, con valores entre **0 y 0.001** (muy pequeños cambios en el mercado).
- μ_{sector} es la tendencia del sector específico de la acción, también con valores entre **0 y 0.001** (cambios pequeños y moderados).
- ϵ es la fluctuación aleatoria, calculada como:

$$\epsilon = \text{random\_factor} \times \text{random\_sign}$$

Donde:

- **random_factor** es un valor aleatorio entre **0** y **0.01** (pequeñas fluctuaciones).
- **random_sign** es un valor que puede ser **1** o **-1** (indica si la fluctuación es hacia arriba o hacia abajo).

- **Interacciones y Respuestas del Servidor**:
  - El servidor recibe las solicitudes de compra o venta de cada cliente, actualiza su cartera y responde con un resumen de la operación (precio, cantidad comprada o vendida, saldo restante). Se utilizará un hilo para cada cliente.
  - Permite consultar el estado de la cartera en cualquier momento, proporcionando un desglose del valor actual de los activos, el saldo disponible y las ganancias o pérdidas acumuladas.
- **Informe de Inversiones al Final de la Simulación**: Al final de cada simulación, el servidor genera un informe que muestra:
  - El valor final de la cartera para cada cliente, incluyendo las ganancias o pérdidas totales.
  - Un resumen de las transacciones realizadas y el rendimiento de cada activo.
  - Comparaciones de rendimiento para ver cuáles clientes obtuvieron mayores beneficios o pérdidas.
- **Configuración del Menú Inicial**:
  - Permite modificar datos iniciales, como el número de clientes (inversores), los tipos de activos disponibles, el saldo inicial de cada cliente y las tarifas por transacción.
  - Incluye la opción de ajustar el intervalo de fluctuación de los valores de mercado y el rango de variación de los precios para simular diferentes condiciones de mercado.
- **Funcionalidad de Reinicio**: Permite reiniciar el sistema para iniciar una nueva simulación con distintos valores iniciales, saldos y condiciones de mercado, facilitando la comparación entre diferentes estrategias de inversión.

3.12. Proyecto 12: Gestor de Reservas de un Hotel

Servidor Central:

- **Gestión Dinámica de Habitaciones:** Administra una **tabla compartida** con:
  - **Tipos de habitación** (normal, con balcón, suite).
  - **Capacidades** específicas para cada tipo de habitación.
  - **Estado actual** (disponible, ocupada, bloqueada por mantenimiento, reservada).
- **Precios Dinámicos:** Ajusta tarifas según la **temporada alta o baja y demanda en tiempo real.**

Fórmula de control de los precios de las habitaciones

$$P_f = P_b \cdot (1 + S + D)$$

Variables:

1. P_b: **Precio base**
   - Tarifa estándar para la habitación, definida previamente.
2. S: **Impacto de la temporada**
   - $S = 0.3$: Temporada alta (incremento del 30%).
   - $S = -0.2$: Temporada baja (reducción del 20%).
   - $S = 0$: Temporada media (sin ajuste).
3. D: **Impacto de la demanda**
   - $D = 0.2$: Alta demanda (incremento del 20%).
   - $D = -0.1$: Baja demanda (reducción del 10%).
   - $D = 0$: Demanda estable (sin ajuste).

    - **Alta Demanda:** El 20% de las habitaciones están libres..
    - **Baja Demanda:** El 80% de las habitaciones están libres.
- **Historial de Reservas:** Registra todas las reservas y cancelaciones para consulta y análisis posterior.
- **Bloqueos Administrativos:** Permite bloquear habitaciones por mantenimiento o cualquier incidencia.

Clientes:

- Simulan usuarios que:
    - Envían solicitudes para **reservar habitaciones**, indicando las fechas de entrada y salida, así como **preferencias** (tipo de habitación).
    - Pueden **consultar su estado de reserva** o **cancelarla** desde un panel de usuario.
    - Reciben respuestas automáticas:

- Confirmación de reserva con **número de habitación asignada**.
- Inclusión en la **lista de espera** si no hay disponibilidad, indicando su posición en la cola.

- Coste de la estancia.

Algoritmo de Priorización:

El sistema utiliza una **cola de prioridades** para procesar las solicitudes. La prioridad se calcula con base en:

1. **Tipo de Cliente**:
   - **VIP/Frecuente:** Alta prioridad.
   - Cliente ocasional: Prioridad media.
   - Nuevos clientes: Baja prioridad.
2. **Duración de la estancia:** Reservas más largas pueden recibir mayor prioridad.
3. **Anticipación de la reserva:** Las solicitudes realizadas con mayor anticipación tienen ventaja.

Fórmula del algoritmo

$$P = (C \cdot 10) + (D \cdot 1) + (A \cdot 0.4)$$

Variables:

1. C: **Tipo de cliente**
   - 3: Cliente VIP o frecuente.
   - 2: Cliente ocasional.
   - 1: Nuevo cliente.
2. D: **Duración de la estancia (en noches)**
   - Número de noches de la reserva.
3. A: **Anticipación (en días)**
   - Días entre la solicitud y la fecha de entrada.

Lista de Espera:

- Las solicitudes no procesadas inmediatamente son añadidas a una **lista de espera priorizada**.
- El servidor monitorea cambios en la disponibilidad y notifica automáticamente a los clientes en espera.
- Los usuarios pueden aceptar o rechazar las nuevas opciones.

Funcionalidades Adicionales:

1. **Modificaciones de Reservas:** Los clientes pueden cambiar fechas o tipos de habitación si hay disponibilidad.
2. **Políticas de Cancelación:** Gestión de cancelaciones con penalizaciones automáticas según las condiciones del hotel.

Informes y Estadísticas:

- Al final del día, el sistema genera un **informe de actividad**, incluyendo:
    - Total de reservas procesadas.
    - Cancelaciones y bloqueos administrativos.
- Estadísticas avanzadas como:
    - Porcentaje de ocupación por semanas.
    - Tipos de habitaciones más solicitadas.

Tolerancia a Fallos y Recuperación:

- **Reinicio del Sistema:** Permite reiniciar sin perder datos críticos.

Mejoras Técnicas:

- **Simulación Multihilo:** Cada cliente se gestiona en un hilo independiente para procesar múltiples solicitudes simultáneamente.
- **Bloqueo de Recursos Compartidos.**

3.13. Proyecto 13: Simulador de Panadería

Descripción General

Desarrolla un sistema que simule la operación de una panadería, gestionando pedidos y optimizando el uso de los hornos. El sistema debe priorizar tareas según tiempos de horneado y atender a los clientes de forma eficiente.

1. Requisitos del Sistema

- **Servidor Central**:
    - Gestiona el estado de los hornos (*ocupado, disponible, en mantenimiento*).
    - Organiza los pedidos utilizando algoritmos de planificación seleccionables:
        - **Shortest Remaining Time Next (SRTN).**
        - **Shortest Job First (SJF).**
    - Realiza mantenimientos preventivos tras un número predefinido de ciclos de uso.
- **Clientes**:
    - Simulan compradores que envían pedidos con tipo y cantidad de productos.
    - Reciben confirmaciones del pedido, tiempo estimado de preparación y notificaciones cuando los productos están listos.
    - Pueden consultar el estado de sus pedidos en tiempo real.

2. Funcionamiento

1. **Hornos y Productos**:
    - Cada horno tiene un límite de ciclos antes de necesitar mantenimiento.
    - Los productos (pan, pasteles, croissants) tienen tiempos de horneado fijos por unidad.
2. **Pedidos**:
    - Cada pedido incluye el tipo de producto y la cantidad solicitada.
    - El tiempo total del pedido se calcula como:

$$T_{\text{pedido}} = \sum_{i=1}^{n} q_i \cdot T_{p_i}$$

- q_i: Cantidad del producto i.
- T_{p_i}: Tiempo por unidad del producto i.

3. **Mantenimiento de Hornos**:

Fórmula del Ciclo Acumulado

$$C_{\text{actual}} = \sum_{i=1}^{n} (q_i \cdot P_{p_i})$$

- C_{actual}: Ciclo acumulado actual del horno.
- q_i: Cantidad horneada del producto i.
- P_{p_i}: Peso asignado al producto i según su impacto en el desgaste (valor predefinido).
- n: Total de tipos de productos horneados.

Lógica de Mantenimiento

- El mantenimiento debe realizarse cuando:

$$C_{\text{actual}} \geq C_{\text{máx}}$$

- $C_{\text{máx}}$: Límite de desgaste antes del mantenimiento (configurable).

Ejemplo de Pesos (P_{p_i}) para Productos

Producto	Peso (P_{p_i})
Pan	1.0
Pasteles	1.5
Croissants	2.0

3. Funcionalidades del Sistema

1. **Pedidos**:
    - Recepción de pedidos desde múltiples clientes.

    - Confirmación con tiempo estimado y posición en la cola.
2. **Notificaciones**:
    - Aviso cuando el pedido esté listo para recoger.
3. **Consultas**:
    - Estado de pedidos en tiempo real.
4. **Configuración Inicial**:
    - Cantidad de hornos.
    - Tipos de productos y tiempos de horneado.
    - Límite de ciclos de los hornos antes de mantenimiento.
5. **Estadísticas Finales**:
    - Tiempo promedio de espera.
    - Uso de hornos y productos horneados.

3.14. Proyecto 14: Proyecto genérico

Realizar una aplicación como las anteriores:

- Con un menú para poder cambiar la cantidad de clientes y servidores.
- Conexiones cifradas entre clientes y servidores.
- Los clientes enviarán datos cada cierto tiempo y estos datos se tendrán que gestionar en el servidor.
- Podéis seleccionar cualquier proceso industrial y aplicar fórmulas a los datos obtenidos para realizar un control, utilizar algoritmos de ordenación u otros similares, etc.
- Los servidores pueden enviar alertas con la información, conectarse a alguna API para obtener información a partir de la que han calculado, etc.
- También es posible implementar una cola con prioridades para gestionar los cálculos, o alternativamente, procesarlos de forma concurrente.
- Tenéis libertad para plantear un proyecto dentro de estos parámetros, **TODOS LOS PROYECTOS DE ESTE TIPO TENDRÁN QUE SER ACEPTADOS PREVIAMENTE**, enviadme un correo.