En esta segunda fase del problema de la automatización de la presentación de examenes se incorpora un segundo servidor, que debe mantenerse comunicado con el otro servidor (ambos nodos).

La primera fase del problema, esta descrita dentro de la carpeta: **un_servidor** en el correspondiente informe.

En esta segunda parte, un servidor o nodo, debe saber si un aspirante se inscribió en el otro nodo e impedir que presente examen en un nodo en el que no se haya inscrito, si previamente ya se inscribió o presentó en el otro nodo. Con esto se impide que pueda presentar la prueba dos veces en los dos nodos.

En la primera fase del desarrollo de una solución para este problema, se implementó un cliente y un servidor y dicha comunicación era síncrona, sin embargo en esta segunda fase la comunicación entre los dos servidores se implementa de manera asíncrona, la razón de ello, es que se desconoce el orden y la cantidad de aspirantes que se inscribirán en un nodo o en otro, en un momento dado. Por lo que necesariamente dicha comunicación debe ser asíncrona (usando select).

La manera de implementar la solución propuesta fue desglosando el problema de la comunicación en problemas mas pequeños pero que estuvieran alineados a la solución del problema global que es lograr una comunicación para poder preservar una consistencia de la información que manejan los dos servidores en todo momento.

- Comunicación entre los servidores usando select.
- Utilización de datos comunes (memoria compartida) usando colas y escribiendo en ellas de manera simétrica y consistente, cada servidor (nodo).
- implementacion en cada servidor de la interpretacion de los datos que genera y que recibe (algoritmo) para preservar la consistencia.

Al resolver cada problema por separado y finalmente integrar todo, se puede llegar a una solución del problema global, programar toda la solución de una sola vez, es complicado y confuso.

El código esta compuesto por las siguientes carpetas:

psd la carpeta principal.
un_servidor: primera parte del proyecto con un cliente y un servidor
dos_servidores: segunda parte del proyecto, que integra a la primera, esta compuesta por dos clientes, dos servidores y dos bases de datos, en ella está integrada, la primera fase, la comunicación de los dos servidores, las colas y la lógica de comunicación (algoritmo) de comunicación entre los servidores.
test_dos_servidores: es la carpeta que contiene un prototipo, previa a la solución integrada (dos_servidores) con listas en vez de bd para implementar el subproblema de colas y comunicación de los servidores, esta carpeta solo se usó para realizar pruebas preliminares antes de implementar la solucion integrada en la carpeta **dos_servidores**.

Posibles mejoras:
Ya que se intentó aprovechar el codigó de la primera fase, tenemos que la comunicación entre cada servidor y sus clientes es síncrona y la comunicación entre los servidores es asíncrona, se puede implementar una solución con clientes que sean atendidos por threads y usen comunicación asíncrona también.

A la hora de presentar examen opción 2, se pide la cédula, y luego vuelve a pedir la cédula y el string aleatorio para validar credenciales, después permite presentar el examen en caso de que no se haya presentado o indicar que ya se presentó si ya se hizo, lo normal es que pida la cédula una sola vez, esto se hizo para no tener que reestructurar el código de la primera parte y poder aprovecharlo.

Si se quiere usar los programas desde distintas máquinas, es necesario colocar las Ips de los servidores y de los clientes y usar un sistema de nombrado o bien colocando las Ips en los /etc/hosts de cada máquina o bien realizando esto ultimo en el DNS interno o bien usando DNSmasq para que las máquinas se puedan localizar.

Como probar los programas:

Se deben abrir cuatro terminales:
Arriba estarán los dos servidores y abajo los clientes.

Terminal 2, ejecutar este despues de 1

./psd/dos_servidores/>python s**ervidor4321c1234.py**

Terminal 1, ejecutar este primero

./psd/dos_servidores/> python **servidor1234.py**

Terminal 3

./psd/dos_servidores/>python **cliente4321.py**

Terminal 4

./psd/dos_servidores/> python **cliente1234.py**

El terminal 1 debe ejecutarse primero ya que el servidor del terminal 2 tiene el cliente que se conecta al servidor del terminal 1, los terminales 3 y 4 (los clientes) se puede ejecutar en cualquier orden o solo el terminal 3 y despues el terminal 4, o viceversa, la única restricción es leventar el terminal 1 primero y despues el terminal 2
para que el cliente del terminal 2 se puede conectar al servidor del terminal 1 y poder establecer la comunicación.

Porque esos nombres en los archivos, servidor4321c1234.py, indica que es el servidor que escucha por el puerto 4321 y tiene un cliente que establece un canal de comunicación donde se encuentra el servidor que escucha por el puerto 1234.

servidor1234.py indica que es el servidor que escucha por el puerto 1234.

Aunque se usó cliente4321.py y cliente1234.py la única diferencia es el puerto al cual se conectan, ya que el código es idéntico para el caso de los clientes.

Cómo funciona la comunicación entre los servidores:

en la carpeta ./psd/dos_servidores_test se hizo un prototipo que en vez de realizar actualizaciones de campos en la base de datos (carpeta dos_servidores) usa listas que almacenan las cédulas y permiten comprobar casos de prueba, todos ellos están implementados en la carpeta (dos_servidores)

Cada cliente que se conecta a un servidor, puede hacer dos cosas, registrarse o presentar el examen en caso de que no lo haya presentado, entonces:

Si la opción es registro, (se usan nombres simbólicos de los Nodos para hacer más amigable la programación, nodo 1 es Caracas (servidor4321c1234.py) y nodo 2 es Maracay (servidor1234.py)
Lo primero que hace un cliente al conectarse es actualizar las colas de entrada, cada elemento de la cola de entrada es un elemento del otro servidor, es decir es alguien que se inscribió en el otro servidor y le notifica a este este hecho al otro servidor.

Si la cédula esta en el otro nodo, le indicará que debe presentar alla.
Si no, colocará esta cédula en la cola de salida para que el otro nodo sepa que se inscribió en este nodo, sino ocurre nada de eso es porque ya está inscrito en este nodo y no colocará nada en la cola de salida.

Con la opcion 2 de presentar examen es algo similar, si no está registrado en el otro nodo, y si está registrado en este nodo podrá comenzar a presentar el examen, de aquí en adelante se ejecuta el código de la primera parte del proyecto, se le preguntará nuevamente la cédula y el string aleatorio y procederá a realizar la prueba, se le enviará el resultado y se actualizará la BD, tambien se escribe en el directorio donde se ejecuta el servidor un archivo cuyo nombre es la cedula del aspirante y cuyo contenido es la cédula y un diccionario con las preguntas y respuestas para dejar una constancia mas detallada de que presento el examen.

El papel de las estructuras de datos y la comunicación de los servidores:

Cada servidor accede a una cola de entrada y a una cola de salida, cada vez que un aspirante se inscribe en un nodo, se le de la cola de entrada y se actualiza la BD y se escribe en la cola de salida para notificar al otro servidor.

Para comunicar a los dos servidores, el cliente1234, leera de su q_saliente y enviara un send para que el servidor1234 esciba en su q_entrante.
El servidor 1234 leera de su q_saliente y enviara send a cliente1234 para que este escriba en su q_entrante.

De esta manera los dos servidores se mantendrán comunicados en todo momento.

Algunos casos de prueba:
cliente se inscribe en nodo, intenta inscribirse en el otro nodo.
cliente se inscribe en nodo, presenta en nodo, e intenta presentar nuevamente en el mismo nodo o inscribirse o presentar en el otro nodo.
Varios clientes se inscriben en un nodo y cualquiera de ellos intenta inscribirse en el otro nodo o presentar en el otro nodo.
Varios clientes se inscriben alternativamente en ambos nodos y cualquiera de ellos intenta inscribirse o presentar en el otro nodo donde no se habia inscrito o habia presentado.
Se abren dos a mas clientes conectados a un servidor o a los dos servidores y se comprueba la concurrencia y la consistencia.