**Informe primera parte segundo trabajo de Bases de Datos II – 2021-1**

***Integrantes:***

*Aristizabal Giraldo Salomé - Rendón Giraldo Sebastián - Valencia Zapata Santiago Alexis*

**Primer punto: Blockchain empresa**

La solución de este punto se compone de los archivos comandosMultichain.txt, cada una de las líneas de código de este archivo deben correrse una a una en la consola en el orden en que aparecen en el archivo, adicionalmente cabe destacar que la línea de código multichaind empresa -daemon debe correrse en una consola individual, esta consola debe permanecer en ejecución para poder ejecutar el código correctamente. En este archivo se crea la blockchain llamada “empresa”, posteriormente se crean 2 streams “vendedores” y “ganancias”.

Para el Smart filter del stream ganancias se verifica inicialmente que la clave cedula sea única, si la longitud de la clave es diferente de 1, posteriormente se verifica que exista la clave, para esto se asegura que la longitud de la clave sea mayor a 1 y que no sea igual a null, seguidamente se verifica que el formato sea json usando .format, luego se verifica que los valores del stream ganancia sean únicos, que su longitud sea mayor a 1 y que no sean nulos ya que son datos obligatorios, finalmente si se ingresa un dato de tipo diferente a número toma el valor de NaN y genera un error.

Para el Smart filter del stream vendedores se verifica inicialmente que la clave cedula sea única, adicionalmente que su longitud sea mayor a 0 y que sea diferente de null ya que este campo es obligatorio, posteriormente se verifica que el formato de los datos ingresados sea Json, este es el único formato que se recibe. Una vez se asegure de que el formato es correcto se toman los demás datos del stream ganancia, la cantidad de datos ingresados sin contar la cedula debe ser igual a 2, ya que solo quedan los campos de nombre y teléfono, posteriormente se verifica que el nombre no sea null y su longitud sea mayor a 1 ya que este campo es obligatorio, y se hace lo mismo con teléfono.

Se obtienen las direcciones, y se procede a activar los Smart filter, cabe destacar que para que estos funcionen debe reemplazar en ambos comandos [dirección] por la dirección que obtuvo en el paso anterior, para que pueda ejecutar estos comandos correctamente, el resto del código son datos de prueba, no es necesario ejecutarlos.

**Segundo punto: Interfaz grafica en java**

Para el segundo punto se creo una aplicación en java, en la dirección src/app podrá encontrar 3 carpetas, controllers, views y models adicionalmente el archivo Main.java, dentro de este encontrara una variable commandManager, en esta debe modificar los valores de port el cual puede encontrar en la línea de comandos en la que ejecuto multichain empresa -daemon y password que puede encontrar en la dirección %appdata%/Multichain/empresa/multichain.conf. En la carpeta controllers puede ver los controladores para cada uno de los botones GananciaController, VendedorController y ConsultarController, si es necesario el uso inputs como en GananciaController y VendedorController estos se inicializaran con ayuda de @FXML para que posteriormente puedan conectarse con las vistas, adicionalmente estos dos archivos contienen métodos create que sirven para añadir datos a los streams de ganancias y vendedores, en formato JSON. Para los 3 controladores mencionados anteriormente se creó una función goToHome() que permite volver a la página principal (index) y la página principal IndexController consta de 3 funciones que se dirigen cada uno a cada una de las vistas de consultar, registrar vendedor y registrar ganancia.

En consultarController se define una variable tipo TableView<Person> este tipo de variable nos facilitara la visualización de la tabla de datos en el archivo Consultar.fxml, adicionalmente se implementa Initializable para sobreescribir la función initialize la cual se ejecutara tan pronto como la clase sea llamada, dentro de esta función definimos varias variables auxiliares que nos ayudaran a encontrar las variables necesitadas, por la naturaleza del initialize se debe ejecutar en un bloque try catch. Se definen 2 listas principales, un ArrayList de StreamKeys llamada vendedores en la cual se almacenaran todas las diferentes cedulas pertenecientes a todos los vendedores, y una lista denominada ganancias en la cual se almacena toda la información de todas las transacciones llevadas a cabo en el stream de ganancias, se usa un for para recorrer la lista de cedulas, dentro hay un for anidado que recorre la lista de transacciones del stream ganancias, cada uno de los datos de esta lista se convierte en un string y posteriormente en una lista llamada parts usando Split, en esta lista la cedula a la que pertene la ganancia esta en la tercera posición, pero además esta rodeada de valores que no representan interés para este caso por eso se realiza un substring de modo que en la variable key solo se almacene el numero completo que corresponde a la cedula del vendedor, posteriormente se realiza un if en el que se compara esta key con la del vendedor, si estas son iguales entonces aux permitirá obtener únicamente el número completo correspondiente a el valor de la ganancia, luego se utiliza la variable total para ir sumando todas las ganancias que pertenezcan a la misma cedula. Finalmente cuando se haya recorrido todo el for anidado se añadira a una lista observable data los valores encontrados de cedula y total.

Adicionalmente en la direccion src/app encontrara la carpeta views en la cual están todas las vistas, en Consultar.fxml se encuentra un TableView con dos columnas que están conectadas a las variables cedulaVendedor y totalGanancias definidas en Person, esto permite visualizar todas las recurrencias de vendedores con las ganancias totales que han acumulado. En index.fxml se encuentran 3 botones que llaman a cada una de las funciones de IndexController permitiéndolo navegar entre las demás vistas disponibles. En cada una de las demás vistas se encuentran botones lVolver al inicio que le permitirán volver al index.

En RegistrarGanancia.fxml se encuentra un choicebox que permite escoger entre las cedulas de los diferentes vendedores existentes, un textField en el que se debe ingresar el valor de la ganancia, y un botón que llama a la función createGanancia dándole como parámetros los definidos en el choicebox y el textField. Finalmente en RegistrarVendedor.fxml hay 3 textField que corresponden a las entradas de cedula, nombre y teléfono, un boton que llama a la función createVendedor dándole como parámetros los definidos en los 3 textField mencionados previamente.

En el directorio src/app se encuentra la carpeta models, la cual contiene la clase Person, esta se utiliza para almacenar los datos que se presentan en la tabla de consultar. Esta tiene 2 parametros, cedulaVendedor y totalGanancias, ambas son de tipo SimpleStringProperty ya que estos tienen unas funciones que serán útiles en la implementación, adicionalmente en esta clase se define el constructor, y las funciones get y set para ambos parametros.