Control de Cambios App ServiPunto

# Creación de Inventario en Adición de Producto

|  |  |
| --- | --- |
| Clase | AddProduct.java |
| Método | **private boolean** insertRecordInDB() |
| Despues de | updateproductdb(); |
| Código | */\*\*  \* Harold: Permite crear el producto en Inventario automáticamente  \*/* List<DTO> temp = getDataFromViews(); ProductDTO dtoTemp = (ProductDTO)temp.get(0); dtoTemp.getBarcode(); SelectedProddutsDTO selectedProddutsDTO = **new** SelectedProddutsDTO(); selectedProddutsDTO.setBarcode(dtoTemp.getBarcode()); selectedProddutsDTO.setQuantity(**"0"**); selectedProddutsDTO.setUnits(dtoTemp.getUom()); selectedProddutsDTO.setPrice(dtoTemp.getPurchasePrice()); selectedProddutsDTO.setSellPrice(dtoTemp.getSellingPrice()); ProductFinalActivity productFinalActivity = **new** ProductFinalActivity(); productFinalActivity.insertInventoryRecord(selectedProddutsDTO,**"000"**); *//END* |

# Cambiar Precio Producto en Módulo Venta

|  |  |
| --- | --- |
| Clase | SalesActivity.java |
| Método | **private** List<DTO> getSalesDetailsList(String payments) |
| Despues de | SelectedProddutsDTO dto = (SelectedProddutsDTO) selectedProdList.get(i); |
| Código | */\* Harold: Valida si el precio cambio y lo cambia en la BD  \*/* validateIfSellPriceChange(dto); *//END*  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Creación del Método \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  */\* Harold: Método que valida si el precio cambio y lo reescribe  \*/* **private void** validateIfSellPriceChange(SelectedProddutsDTO product){  **double** utility = 0.0,utilityTemp = 0.0;  **try**{  utilityTemp = CommonMethods.*getDoubleFormate*(product.getSellPrice())-CommonMethods.*getDoubleFormate*(product.getPrice());  utility = CommonMethods.*getDoubleFormate*(product.getUtilityValue());   }  **catch** (Exception e){  e.printStackTrace();  }  **if**(utilityTemp!=utility){  ProductDTO dtoo = ProductDAO.*getInstance*().getRecordsByBarcode(  DBHandler.*getDBObj*(Constants.***READABLE***), product.getBarcode());  dtoo.setSellingPrice(product.getSellPrice());  dtoo.setSyncStatus(0);  ProductDAO.*getInstance*().updateProductSynk(  DBHandler.*getDBObj*(Constants.***WRITABLE***), dtoo);  }  } |

# Traer todos los Proveedores de la Lista en la Creación del Producto

|  |  |
| --- | --- |
| Clase | AddProductActivity.java |
| Método | **private** List<String> getOutProviders() { |
| Despues de | supplierTable = **new** Hashtable<String, String>(); |
| Código | *//Add Harold to return all supplier when supplier is new* **for** (DTO dto : **supplierList**) {  SupplierDTO suppDto = (SupplierDTO) dto;  list.add(suppDto.getName() + **" - "** + suppDto.getCedula());  **supplierTable**.put(suppDto.getName() + **" - "** + suppDto.getCedula(),suppDto.getCedula()); } Collections.*sort*(list);  list.add(0,**productDto**.getSupplierName() + **" - "** + **productDto**.getSupplierId()); *//END* |

# Cambiar Nombre del Producto y Proveedor Cuando Viene de la Maestra

|  |  |
| --- | --- |
| Clase | AddProductActivity.java |
| Método | **private void** setDataToViewsFromCentroalServer() { |
| Despues de |  |
| Código | *//Harold: Cambiamos a true* **spnSupplier**.setEnabled(**true**);  *//Se comenta la siguiente línea //spnSupplier.setBackgroundResource(R.drawable.etxt\_disable);*  *//Harold: Cambiamos A True* **etxtprodName**.setEnabled(**true**); *//Comentamos las siguiente línea //etxtprodName.setBackgroundResource(R.drawable.etxt\_disable);* |

# Realizar Precargue de la información

|  |  |
| --- | --- |
| Clase | ProductsActivity.java |
| Método | **private void** inItUI() { |
| Despues de | imgCloseicon.setOnClickListener(**this**); |
| Código | *//add HAROLD* **imgLogoLoadProduct** = (ImageView) findViewById(R.id.***img\_logo***); **imgLogoLoadProduct**.setOnClickListener(**this**); |
|  |  |
| **Clase** | ProductsActivity.java |
| Método | **private void** inItUI() { |
| Despues de | **case** R.id.***btn\_filter***:  *// showFilters();* **break**; |
| Código | *//Harold. Case que da la opción de Precargue* **case** R.id.***img\_logo***:  **final** AlertDialog.Builder builder = **new** AlertDialog.Builder(**this**);  builder.setTitle(**"Realizar un precargue de información"**);  builder.setMessage(**"1. Asegurese de que tenga toda la información del Servidor\n"** +  **"2. Si no está seguro Sincronice la información, borre el registro de sincronización en Proveedores,\n"** +  **"cierre el establecimiento y vuelva a Ingresar Usuario Y Contraseña\n\n"** +  **"¿Desea realizar el Precargue?"**);  builder.setCancelable(**true**);  builder.setNegativeButton(**"Cancelar"**, **new** DialogInterface.OnClickListener() {  **public void** onClick(DialogInterface dialog, **int** which) {  dialog.cancel();  }   });  builder.setPositiveButton(**"Aceptar"**, **new** DialogInterface.OnClickListener() {  **public void** onClick(DialogInterface dialog, **int** which) {  loadInformation();   }  });  AlertDialog alert11 = builder.create();  alert11.show();  **break**; |

|  |  |
| --- | --- |
| Clase | ProductsActivity.java |
| Método | **Nuevo método** |
| Despues de | **NA** |
| Código | *//Add Harold, Método que carga la información* **private void** loadInformation() {   List<DTO> productsToInsert = loadProducts(**appContext**);  List<DTO> suppliersToInsert = loadSuppliers(**appContext**);  List<DTO> inventoryToInsert = loadInventory(**appContext**);   **if** (productsToInsert == **null** || suppliersToInsert == **null** || inventoryToInsert == **null**)  ;  **else** {   ProductDAO.*getInstance*().insert(  DBHandler.*getDBObj*(Constants.***WRITABLE***), productsToInsert);  SupplierDAO.*getInstance*().insert(  DBHandler.*getDBObj*(Constants.***WRITABLE***), suppliersToInsert);  InventoryDAO.*getInstance*().insert(  DBHandler.*getDBObj*(Constants.***WRITABLE***), inventoryToInsert);  refreshList();  } }  *//Add Harold, Carga productos, proveedores e Inventario* **private** List<DTO> loadProducts(Context activity) {  **try** {  String tempProduct[];  AssetManager am = activity.getAssets();  InputStream is = am.open(**"Products"**);  Scanner s = **new** Scanner(is).useDelimiter(**"\\n"**);  List<DTO> productListTemp = **new** ArrayList<DTO>();  ProductDTO productTemp = **new** ProductDTO();  **int** prodCount;  **while** (s.hasNext()) {  tempProduct = s.next().split(**";"**);  prodCount = ProductDAO.*getInstance*().isProductExists(  DBHandler.*getDBObj*(Constants.***READABLE***), tempProduct[0]);  **if** (prodCount != 0) ;  **else** {  productTemp = **new** ProductDTO();  productTemp.setProductId(tempProduct[0]);  productTemp.setBarcode(tempProduct[0]);  productTemp.setName(tempProduct[1]);  productTemp.setQuantity(**"0"**);  productTemp.setPurchasePrice(tempProduct[2]);  productTemp.setSellingPrice(tempProduct[3]);  productTemp.setVat(**"0"**);  productTemp.setSupplierId(tempProduct[4]);  productTemp.setGroupId(**""**);  productTemp.setLineId(**""**);  productTemp.setUom(**""**);  productTemp.setCreateDate(Dates.*getSysDate*(Dates.***YYYY\_MM\_DD\_HH\_MM***));  productTemp.setModifiedDate(Dates.*getSysDate*(Dates.***YYYY\_MM\_DD\_HH\_MM***));  productTemp.setProductFlag(**"1"**);  productTemp.setActiveStatus(Constants.***TRUE***);  productTemp.setSyncStatus(Constants.***TRUE***);  productTemp.setSubgroup(**""**);  productTemp.setMin\_count\_inventory(**"1"**);  productTemp.setExpiry\_date(**""**);  productTemp.setDiscount(**""**);  productListTemp.add(productTemp);   }  }   is.close();  s.close();   **return** productListTemp;  } **catch** (Exception e) {  e.printStackTrace();  **return null**;  }   }  **private** List<DTO> loadSuppliers(Context activity) {  **try** {  String tempSupplier[];  AssetManager am = activity.getAssets();  InputStream is = am.open(**"Suppliers"**);  Scanner s = **new** Scanner(is).useDelimiter(**"\\n"**);  List<DTO> supplierListTemp = **new** ArrayList<DTO>();  SupplierDTO supDTO = **new** SupplierDTO();  List<String> supplierIDsList = SupplierDAO.*getInstance*()  .getSupplierIDs(  DBHandler.*getDBObj*(Constants.***READABLE***));    **while** (s.hasNext()) {  tempSupplier = s.next().split(**";"**);    **if** (supplierIDsList.contains(tempSupplier[0])) ;  **else** {  supDTO = **new** SupplierDTO();  supDTO.setSupplierId(tempSupplier[0]);  supDTO.setCedula(tempSupplier[0]);  supDTO.setName(tempSupplier[1]);  supDTO.setAddress(**""**);  supDTO.setTelephone(**""**);  supDTO.setVisitFrequency(**""**);  supDTO.setVisitDays(**""**);  supDTO.setContactName(**""**);  supDTO.setContactTelephone(**""**);  supDTO.setModifiedDate(Dates.*getSysDate*(Dates.***YYYY\_MM\_DD\_HH\_MM***));  supDTO.setSyncStatus(1);  supplierListTemp.add(supDTO);  }  }    is.close();  s.close();  **return** supplierListTemp;  } **catch** (Exception e) {  e.printStackTrace();  **return null**;  } }   **private** List<DTO> loadInventory(Context activity){  **try**{  String tempInventory[];  AssetManager am = activity.getAssets();  InputStream is = am.open(**"Products"**);  Scanner s = **new** Scanner(is).useDelimiter(**"\\n"**);  List<DTO> listTempInventory = **new** ArrayList<DTO>();  InventoryDTO inventoryTempDTO ;  List<String> inventoryIDs = InventoryDAO.*getInstance*()  .getInventoryIDs(  DBHandler.*getDBObj*(Constants.***READABLE***));  **int** count = 0;  Long id = Dates.*getSysDateinMilliSeconds*();   **while** (s.hasNext()){  tempInventory = s.next().split(**";"**);    **if** (inventoryIDs.contains(tempInventory[0]));  **else**{  inventoryTempDTO = **new** InventoryDTO();  inventoryTempDTO.setInventoryId((id+count)+**""**);  inventoryTempDTO.setProductId(tempInventory[0]);  inventoryTempDTO.setQuantity(**"0"**);  inventoryTempDTO.setSyncStatus(1);  inventoryTempDTO.setUom(**""**);  listTempInventory.add(inventoryTempDTO);  count++;  }  }    is.close();  s.close();  **return** listTempInventory;  }  **catch** (Exception e){  e.printStackTrace();  **return null**;  } } |

|  |  |
| --- | --- |
| Clase | InventoryDao.java |
| Método | **Nuevo método** |
| Despues de | **NA** |
| Código | **public** List<String> getInventoryIDs(SQLiteDatabase dbObject) {  List<String> instList = **new** ArrayList<String>();  Cursor cursor = **null**;   **try** {  cursor = dbObject.rawQuery(**"SELECT product\_id FROM tbl\_inventory"**, **null**);  **if** (cursor.moveToFirst()) {  **do** {   instList.add(cursor.getString(0));   } **while** (cursor.moveToNext());  }  } **catch** (Exception e) {  Log.*e*(**"InventeryDAO -- getProductsIDs"**, e.getMessage());  } **finally** {  **if** (cursor != **null** && !cursor.isClosed())  cursor.close();   dbObject.close();  }   **return** instList; } |

|  |  |
| --- | --- |
| Clase | SalesActivity.java |
| Método | **private void** updateMenuAndDishInventory() { |
| Despues de | inDTO.setProductId(selectedDTO.getIdProduct()); |
| Código | *//Harold* **if** (inDTO.getQuantity().equals(**"0"**))  updateNewProduct(selectedDTO); |

|  |  |
| --- | --- |
| Clase | SalesActivity.java |
| Método | **Nuevo Método** |
| Despues de | NA |
| Código | *//Add Harold* **private void** updateNewProduct(SelectedProddutsDTO selectedDTO){   ProductDTO productDTOTemp = ProductDAO.*getInstance*().getRecordsByBarcode(DBHandler.*getDBObj*(Constants.***READABLE***),  selectedDTO.getIdProduct());  productDTOTemp.setSyncStatus(0);  SupplierDTO supplierDTOTemp = SupplierDAO.*getInstance*().getRecordsBySupplierID(DBHandler.*getDBObj*(Constants.***READABLE***),  productDTOTemp.getSupplierId());  supplierDTOTemp.setSyncStatus(0);  ProductDAO.*getInstance*().update(DBHandler.*getDBObj*(Constants.***WRITABLE***), productDTOTemp);  SupplierDAO.*getInstance*().update(DBHandler.*getDBObj*(Constants.***WRITABLE***), supplierDTOTemp); } |

# Borrar Registro de Sincronización

|  |  |
| --- | --- |
| Clase | SuppliersActivity.java |
| Método | **private void** inItUI() { |
| Despues de | lvSupplier.setClickable(**false**); |
| Código | *//Harold* imgLogoDeleteSyncRecord.setOnClickListener(**this**);  *//Harold* **imgLogoDeleteSyncRecord** = (ImageView) findViewById(R.id.***img\_logo***); **imgLogoDeleteSyncRecord**.setOnClickListener(**this**); |

|  |  |
| --- | --- |
| Clase | SuppliersActivity.java |
| Método | **private void** inItUI() { |
| Despues de | **case** R.id.***btn\_Addsupplier***:  //…  **break**; |
| Código | **case** R.id.***img\_logo***:   **final** AlertDialog.Builder builder = **new** AlertDialog.Builder(**this**);   builder.setTitle(**"Usted Borrará el Registro de Sincronización"**);  builder.setMessage(**"Esta opción le ayudará a recuperar la información del Servidor\n"** +  **"Utilicela en caso que haya Formateado o Cambiado la Tablet\n"** +  **"Una vez borrado el registro, por favor cierre la tienda en el Modulo Abrir o Cerrar,\n"** +  **"Salgase del aplicativo e Ingrese Usuario y Contraseña"** +  **"Para que la información se descargue del servidor\n\n"** +  **"¿Desea continuar?"**);  builder.setCancelable(**true**);  builder.setNegativeButton(**"Cancelar"**, **new** DialogInterface.OnClickListener() {  **public void** onClick(DialogInterface dialog, **int** which) {  dialog.cancel();  }  });  builder.setPositiveButton(**"Aceptar"**, **new** DialogInterface.OnClickListener() {  **public void** onClick(DialogInterface dialog, **int** which) {  **new** DeleteCatlogsyncRecord().execute();  }  });  AlertDialog alert11 = builder.create();  alert11.show();   **break**; |

|  |  |
| --- | --- |
| Clase | SuppliersActivity.java |
| Método | **Nuevo Método** |
| Despues de |  |
| Código | **private class** DeleteCatlogsyncRecord **extends** AsyncTask<Void, Void, Void> {  ServiceHandler **servicehandler** = **new** ServiceHandler(SuppliersActivity.**this**);   @Override  **protected void** onPreExecute() {  **super**.onPreExecute();  }   @Override  **protected** Void doInBackground(Void... params) {  **if** (CommonMethods.*getInternetSpeed*(SuppliersActivity.**this**) >= ServiApplication.*local\_data\_speed*) {  SharedPreferences sharedpreferences;  sharedpreferences = getSharedPreferences(  ServiApplication.*MyPREFERENCES*, Context.***MODE\_PRIVATE***);  String val\_productDetails = **servicehandler**.makeServiceCall(  ServiApplication.*URL* + **"/catalog/clear/"** + sharedpreferences.getString(**"store\_code"**, **""**),  ServiceHandler.***POST***, **null**);  **if** (**new** JSONStatus().status(val\_productDetails) == 0) {  CommonMethods.*showCustomToast*(SuppliersActivity.**this**, getResources().getString(R.string.***store\_remove***));  }  } **else** {  ServiApplication.*connectionTimeOutState* = **false**;  }  **return null**;  }   @Override  **protected void** onPostExecute(Void result) {  **super**.onPostExecute(result);  CommonMethods.*progressDialog*.dismiss();   } } |

# Crear Producto desde Venta

|  |  |
| --- | --- |
| Clase | SalesActivity.java |
| Método | etxtBarcode.setOnKeyListener(**new** OnKeyListener() { |
| Despues de | etxtBarcode.setSingleLine(**true**); |
| Código | **if** (validateBarCode()) {  …  …  …  } **else** {  **if**(*etxtBarcode*.getText().length()>5) {  getDataFromCentralserver(*etxtBarcode*.getText()  .toString().trim());  }  **else** {  CommonMethods.*displayAlert*(**true**, getResources().getString(R.string.***alert***),  getResources().getString(R.string.***noproduct\_exist***),  getResources().getString(R.string.***ok***), **""**, SalesActivity.**this**, **null**, **false**);  } } |

|  |  |
| --- | --- |
| Clase | SalesActivity.java |
| Método | Nuevo Método |
| Despues de | NA |
| Código | *//Add Harold* **private void** updateNewProduct(SelectedProddutsDTO selectedDTO) {  ProductDTO productDTOTemp = ProductDAO.*getInstance*().getRecordsByBarcode(DBHandler.*getDBObj*(Constants.***READABLE***),  selectedDTO.getIdProduct());  productDTOTemp.setSyncStatus(0);  SupplierDTO supplierDTOTemp = SupplierDAO.*getInstance*().getRecordsBySupplierID(DBHandler.*getDBObj*(Constants.***READABLE***),  productDTOTemp.getSupplierId());  supplierDTOTemp.setSyncStatus(0);  ProductDAO.*getInstance*().update(DBHandler.*getDBObj*(Constants.***WRITABLE***), productDTOTemp);  SupplierDAO.*getInstance*().update(DBHandler.*getDBObj*(Constants.***WRITABLE***), supplierDTOTemp); }  *////////add by Andres Diaz////////* **private void** getDataFromCentralserver(String trim) {  **if** (NetworkConnectivity.*netWorkAvailability*(SalesActivity.**this**)) {  **new** GetProductMasterData().execute(trim);  } **else** {  CommonMethods.*showCustomToast*(SalesActivity.**this**,  getResources().getString(R.string.***no\_wifi\_adhoc***));  createEmptyProduct();   } }  **private class** GetProductMasterData **extends** AsyncTask<String, Void, Void> {  ServiceHandler **servicehandler** = **new** ServiceHandler(  SalesActivity.**this**);  String **val\_clientList**;   @Override  **protected void** onPreExecute() {  **super**.onPreExecute();  CommonMethods.*showProgressDialog*(getString(R.string.***please\_wait***),  SalesActivity.**this**);  }   @Override  **protected** Void doInBackground(String... params) {  **val\_clientList** = **servicehandler**.makeServiceCall(  ServiApplication.*URL* + **"/product-master/"** + params[0],  ServiceHandler.***GET***);  **return null**;  }   @Override  **protected void** onPostExecute(Void result) {  **super**.onPostExecute(result);  **if** (**new** JSONStatus().status(**val\_clientList**) == 0) {  **if** (ServiApplication.*connectionTimeOutState*) {  ProductDTO prodto = **new** ParsingHandler()  .mAsterProductInfo(**val\_clientList**);  **if** (ServiApplication.*connectionTimeOutState*) {  **getCentralflage** = **true**;  *productDto* = prodto;  setDataToViewsFromCentroalServer();  }  }  } **else** {  createEmptyProduct();  }  CommonMethods.*dismissProgressDialog*();  } }  **private void** setDataToViewsFromCentroalServer() {  **try** {  **if** (*productDto*.getSupplierId().length() > 0) {  *preciocompra* = *productDto*.getPurchasePrice();  *precioventa* = *productDto*.getSellingPrice();  *nombre* = *productDto*.getName();  }    newSimpleProductDialog dialog = **new** newSimpleProductDialog(SalesActivity.**this**);  dialog.show();  } **catch** (Exception e) {  e.printStackTrace();  } } **private void** createEmptyProduct() {  *productDto*.setPurchasePrice(**"0.0"**);  *productDto*.setSellingPrice(**"0.0"**);  *productDto*.setName(**""**);  *productDto*.setBarcode(**etxtBarcode**.getText().toString().trim());  *productDto*.setSupplierName(**"NO DISPONIBLE"**);  *productDto*.setGroupId(**"0"**);  *productDto*.setGroupName(**"0"**);  *productDto*.setLineId(**"0"**);  *productDto*.setLineName(**"0"**);  *productDto*.setSubgroup(**"NO DISPONIBLE"**);  *productDto*.setUom(**""**);  *productDto*.setSupplierId(**"14236"**);  *productDto*.setVat(**"0"**);  *productDto*.setQuantity(**"0"**);  *productDto*.setSyncStatus(0);  *productDto*.setMin\_count\_inventory(**"1"**);  setDataToViewsFromCentroalServer(); }  **public void** validateAndSave() {   StringBuffer stringBuffer = **new** StringBuffer();   **if** (!validationFields(stringBuffer)){  CommonMethods.*displayAlert*(**true**,  getResources().getString(R.string.***alert***),  stringBuffer.toString(),  getResources().getString(R.string.***ok***), **""**,  SalesActivity.**this**, **null**, **false**);}  insertRecordInDB(); }  **private boolean** validationFields(StringBuffer stringBuffer) {   **name** = newSimpleProductDialog.*nombreProducto*;  **barcode** = *productDto*.getBarcode();  **purprice** = newSimpleProductDialog.*preciocompra*;  **sellprice** = newSimpleProductDialog.*precioventa*;  **vat** = **"0"**;  **supplier** = *productDto*.getSupplierName();  **group** = *productDto*.getGroupName();  **line** = *productDto*.getLineName();  **uom** = *productDto*.getUom()==**"0"**?**""**:**""**;  **subgroup** =*productDto*.getSupplierName();  **min\_count\_in\_inventory** = **"1"**;  **isValid** = **true**;   **if** (**name**.length() == 0) {  **isValid** = **false**;  stringBuffer.append(getResources().getString(R.string.***enter\_name***));  } **else if** (**"0"**.equals(**name**) || **"."**.equals(**name**)) {  **isValid** = **false**;  stringBuffer  .append(getResources().getString(R.string.***invalid\_name***));  }  **if** (**barcode**.length() == 0) {  **isValid** = **false**;  stringBuffer.append(**"\n"** + getResources().getString(R.string.***enter\_barcode***));  } **else if** (**"0"**.equals(**barcode**) || **"."**.equals(**barcode**)) {  **isValid** = **false**;  stringBuffer.append(**"\n"** + getResources().getString(R.string.***invalid\_barcode***));  }    **if** (**min\_count\_in\_inventory**.length() == 0) {  **isValid** = **false**;  stringBuffer.append(**"\n"** + getResources().getString(R.string.***invalid\_qty***));  }   **if** (**purprice**.length() == 0) {  **isValid** = **false**;  stringBuffer.append(**"\n"** + getResources().getString(R.string.***enter\_purprice***));  } **else if** (**"0"**.equals(**purprice**) || **"."**.equals(**purprice**)) {  **isValid** = **false**;  stringBuffer.append(**"\n"** + getResources().getString(R.string.***invalid\_purchase***));  }   **if** (**sellprice**.length() == 0) {  **isValid** = **false**;  stringBuffer.append(**"\n"** + getResources().getString(R.string.***enter\_selling\_price***));  } **else if** (**"0"**.equals(**sellprice**) || **"."**.equals(**sellprice**)) {  **isValid** = **false**;  stringBuffer.append(**"\n"** + getResources().getString(R.string.***invalid\_selling***));  }   **if** (CommonMethods.*getDoubleFormate*(**purprice**) > CommonMethods  .*getDoubleFormate*(**sellprice**)) {  **isValid** = **false**;  stringBuffer.append(**"\n"** + getResources().getString(R.string.***error\_selling\_price***));  }     **if** (**vat**.length() == 0) {  **isValid** = **false**;  stringBuffer.append(getResources().getString(  R.string.***enter\_validvat***));   } **else if** (**"."**.equals(**vat**)) {  **isValid** = **false**;  stringBuffer.append(**"\n"** + getResources().getString(R.string.***invalid\_vat***));  } **else if** (**vat**.length() != 0 && !**"."**.equals(**vat**)) {  **if** (CommonMethods.*getDoubleFormate*(**vat**) > 100) {  **isValid** = **false**;  stringBuffer.append(**"\n"** + getResources().getString(R.string.***vat\_msg***));  }  }  **if** (**supplier**.equals(getResources().getString(R.string.***select***))) {  **isValid** = **false**;  stringBuffer.append(**"\n"** + getResources().getString(R.string.***select\_supplier***));  }  **return isValid**;  }   **private boolean** insertRecordInDB() {  *// barcode* **if** (ProductDAO.*getInstance*().insert(  DBHandler.*getDBObj*(Constants.***WRITABLE***), getDataFromViews())) {   updateSupplierFromServer();   *//Add by Harold And Ivan* List<DTO> temp = getDataFromViews();  ProductDTO dtoHarold = (ProductDTO)temp.get(0);  dtoHarold.getBarcode();  SelectedProddutsDTO selectedProddutsDTO = **new** SelectedProddutsDTO();  selectedProddutsDTO.setBarcode(dtoHarold.getBarcode());  selectedProddutsDTO.setQuantity(**"0"**);   selectedProddutsDTO.setUnits(dtoHarold.getUom());  selectedProddutsDTO.setPrice(dtoHarold.getPurchasePrice());  selectedProddutsDTO.setSellPrice(dtoHarold.getSellingPrice());  ProductFinalActivity productFinalActivity = **new** ProductFinalActivity();  productFinalActivity.insertInventoryRecord(selectedProddutsDTO,**"000"**);   **return true**;    }  **return false**; }  *//AddHarold* **private void** updateSupplierFromServer() {  **try** {   List<DTO> ldto = **new** ArrayList<DTO>();   SupplierDTO lsdto = SupplierDAO.*getInstance*()  .getRecordsBySupplierID(  DBHandler.*getDBObj*(Constants.***READABLE***),  *productDto*.getSupplierId());  **if** (*productDto*.getSupplierId().equals(lsdto.getCedula())) {  } **else** {  SupplierDTO sdto = **new** SupplierDTO();  sdto.setCedula(*productDto*.getSupplierId());  sdto.setName(*productDto*.getSupplierName());  sdto.setVisitFrequency(**""** + 0);  sdto.setBalanceAmount(**"0"**);  sdto.setCreatedDate(Dates.*getSysDate*(Dates.***YYYY\_MM\_DD\_HH\_MM***));  ldto.add(sdto);  SupplierDAO.*getInstance*().insert(  DBHandler.*getDBObj*(Constants.***WRITABLE***), ldto);  ServiApplication.*setSupplier* = SupplierDAO.*getInstance*()  .getRecordsWithValues(  DBHandler.*getDBObj*(Constants.***READABLE***),  **"cedula"**, *productDto*.getSupplierId());  **for** (DTO dto : ServiApplication.*setSupplier*) {  SupplierDTO stdo = (SupplierDTO) dto;  }  Intent intent = **new** Intent(SalesActivity.**this**,  SupplierUpdateService.**class**);  startService(intent);  ldto.clear();  }   *//if (getCentralflage) {* GroupDTO gdto = **new** GroupDTO();  gdto.setGroupId(*productDto*.getGroupId());  gdto.setName(*productDto*.getGroupName());  ldto.add(gdto);  GroupDAO.*getInstance*().insert(  DBHandler.*getDBObj*(Constants.***WRITABLE***), ldto);  ldto.clear();   LineDTO linedto = **new** LineDTO();  linedto.setLineId(*productDto*.getLineId());  linedto.setName(*productDto*.getLineName());  linedto.setGroupId(*productDto*.getGroupId());  ldto.add(linedto);  LineDAO.*getInstance*().insert(  DBHandler.*getDBObj*(Constants.***WRITABLE***), ldto);  ldto.clear();   ProductDTO dtoo = ProductDAO.*getInstance*().getRecordsByBarcode(  DBHandler.*getDBObj*(Constants.***WRITABLE***), **barcode**);  dtoo.setSupplierId(*productDto*.getSupplierId());  ProductDAO.*getInstance*().updateProductSynk(  DBHandler.*getDBObj*(Constants.***WRITABLE***), dtoo);   *//}* } **catch** (Exception e){  } }  @SuppressWarnings(**"static-access"**) **private** List<DTO> getDataFromViews() {  **fecha\_final** = **new** Dates().*serverdateformate*(Dates  .*getSysDate*(Dates.***YYYY\_MM\_DD\_HH\_MM***));  List<DTO> productList = **new** ArrayList<DTO>();   **if** (**uom**.equals(getResources().getString(R.string.***select***)))  **uom** = **""**;  *// uom = uomTable.get(uom);* ProductDTO prodDTO = **new** ProductDTO();  prodDTO.setBarcode(**barcode**);  prodDTO.setName(**name**);  prodDTO.setQuantity(**" "**);  prodDTO.setPurchasePrice(**purprice**);  prodDTO.setSellingPrice(**sellprice**);  prodDTO.setVat(**vat**);  *//prodDTO.setSupplierId(supplierTable.get(supplier));* prodDTO.setQuantity(**"0"**);  prodDTO.setGroupId(**group**);  prodDTO.setLineId(**line**);  prodDTO.setUom(**uom**);  prodDTO.setCreateDate(Dates.*getSysDate*(Dates.***YYYY\_MM\_DD\_HH\_MM***));  prodDTO.setActiveStatus(Constants.***FALSE***);  prodDTO.setSyncStatus(Constants.***FALSE***);  prodDTO.setProductFlag(**"0"**);  prodDTO.setMin\_count\_inventory(**min\_count\_in\_inventory**);  prodDTO.setExpiry\_date(**expiry\_date**);  prodDTO.setDiscount(**discount**);  prodDTO.setFecha\_inicial(**fecha\_inicial**);  prodDTO.setFecha\_final(**fecha\_final**);  *//sup\_id = supplierTable.get(supplier);* productList.add(prodDTO);  **return** productList; } |

Anexo Layout, en el código.

Gracias