Remove isStationary item fields from DTOs and Models

OrderItem, PurchaseItem, StockItem, PurchaseSummary

OrderItemService.cs

```
protected override IQueryable<OrderItem> ConfigureDataReader(IQueryable<OrderItem>
entities)
        {
            // return base.ConfigureDataReader(entities);
            return entities.Where(e => !e.IsStationery).Include(e =>
e.Supplier).Include(e => e.QuantityDescriptor).Include(i => i.NominalAccount);
        }
```

```
protected override IQueryable<OrderItem> ConfigureDataReader(IQueryable<OrderItem> entities)

{

   return entities

     .Include(e => e.Supplier)

     .Include(e => e.QuantityDescriptor)

     .Include(i => i.NominalAccount);

}
```

PurchaseItemController.cs

```
[HttpPost]
        [Route("{id}/getAllForRequsition")]
        public async Task<IActionResult> GetAllForRequsition(int id, CancellationToken
cancellationToken = default)
        {
            var res = (await this.CrudService.GetAll(cancellationToken)).Where(a
=> !a.IsStationery && !(a.IsInCanceledOrder ?? false) && a.PurchaseSummaryId == id);

            //This is not the most elegant solution, but at the moment I do not know
how to map the URL field
            foreach (var item in res)
            {
                var orderItem = this._context.Set<OrderItem>().FirstOrDefault(a =>
a.Id == item.OrderItemId);
                if (orderItem!=null)
                {
                    var fileDetails =
this._context.Set<FileDetails>().FirstOrDefault(a => a.Id == orderItem.FileDetailsId);
                    var map = _mapper.Map<FileDetailsDTO>(fileDetails);
                    if (map != null)
                    {
                        item.URL = map.Url;
                    }
                }

            }
```

```
            return Ok(res);
        }
```

```
[HttpPost]
[Route("{id}/getAllForRequsition")]
public async Task<IActionResult> GetAllForRequsition(int id, CancellationToken cancellationToken = default)
{
   var res = (await this.CrudService.GetAll(cancellationToken)).Where(a => !(a.IsInCanceledOrder ?? false) && a.PurchaseSummaryId == id);

   //This is not the most elegant solution, but at the moment I do not know how to map the URL field
   foreach (var item in res)
   {
      var orderItem = this._context.Set<OrderItem>().FirstOrDefault(a => a.Id == item.OrderItemId);
      if (orderItem!=null)
      {
         var fileDetails = this._context.Set<FileDetails>().FirstOrDefault(a => a.Id == orderItem.FileDetailsId);
         var map = _mapper.Map<FileDetailsDTO>(fileDetails);
         if (map != null)
         {
            item.URL = map.Url;
         }
      }
   }
   return Ok(res);
}
```

PurchaseSummaryService.cs

```
public PurchaseSummaryDTO Get(string name, int skipId)
        {
            var query = GetQueryable<PurchaseSummary>()
```

```
                .AsQueryable();

            if (skipId > 0)
            {
                query = query.Where(x => !x.IsStationery && x.Id != skipId);
            }

            if (!string.IsNullOrEmpty(name))
            {
                query = query.Where(x => !x.IsStationery && x.Requester.UserName ==
name);
            }

            return Mapper.Map<PurchaseSummaryDTO>(query.FirstOrDefault());
        }


public PurchaseSummaryDTO Get(string name, int skipId)
        {
            var query = GetQueryable<PurchaseSummary>()
                .AsQueryable();

            if (skipId > 0)
            {
                query = query.Where(x => x.Id != skipId);
            }

            if (!string.IsNullOrEmpty(name))
            {
                query = query.Where(x => x.Requester.UserName == name);
            }

            return Mapper.Map<PurchaseSummaryDTO>(query.FirstOrDefault());
        }
```

StationaryItemService.cs

```
protected override IQueryable<OrderItem> ConfigureDataReader(IQueryable<OrderItem>
entities)
        {
            entities = base.ConfigureDataReader(entities).Where(e => e.IsStationery)
                .Include(e => e.Supplier)
                .Include(e => e.QuantityDescriptor)
                .Include(a => a.FileDetails).Include(i => i.NominalAccount);
            return entities;
        }


protected override IQueryable<OrderItem> ConfigureDataReader(IQueryable<OrderItem>
entities)
        {
            entities = base.ConfigureDataReader(entities)
                .Include(e => e.Supplier)
                .Include(e => e.QuantityDescriptor)
                .Include(a => a.FileDetails).Include(i => i.NominalAccount);
            return entities;
        }
```

Remove line 264,

```
item.IsStationery = isStationeryItems;
```

StationeryRequisitionService.cs

```
protected override IQueryable<PurchaseSummary>
ConfigureDataReader(IQueryable<PurchaseSummary> entities)
        {
            // return base.ConfigureDataReader(entities).Where(x => x.IsStationery &&
x.Status.ToLower() != "not saved" && !x.IsNotSaved).Include(ps =>
ps.Requester).Include(ps => ps.PrimaryApprover).Include(ps =>
ps.FinalApprover).Include(ps => ps.Unit);
            return base.ConfigureDataReader(entities).Where(x => x.IsStationery /* &&
x.Status.ToLower() != "not saved" */ && !x.IsNotSaved).Include(ps =>
ps.Requester).Include(ps => ps.PrimaryApprover).Include(ps =>
ps.FinalApprover).Include(ps => ps.Unit).Include(p => p.Statuses);
        }
```

```
protected override IQueryable<PurchaseSummary>
ConfigureDataReader(IQueryable<PurchaseSummary> entities)
        {
            // return base.ConfigureDataReader(entities).Where(x => x.IsStationery &&
x.Status.ToLower() != "not saved" && !x.IsNotSaved).Include(ps =>
ps.Requester).Include(ps => ps.PrimaryApprover).Include(ps =>
ps.FinalApprover).Include(ps => ps.Unit);
             return base.ConfigureDataReader(entities).Where(x
=> !x.IsNotSaved).Include(ps => ps.Requester).Include(ps =>
ps.PrimaryApprover).Include(ps => ps.FinalApprover).Include(ps => ps.Unit).Include(p
=> p.Statuses);
        }
```

StockItemService.cs

Remove line 314,

```
stockItem.IsStationery = purchItem.IsStationery;
```

Remove line 361,

```
IsStationery = item.IsStationery,
```

Remove line 465,

```
stockItem.IsStationery = purchItem.IsStationery;
```

Remove line 583,

```
stockItem.IsStationery = purchItem.IsStationery;
```

Remove line 688,

```
IsStationery = item.IsStationery,
```

StationeryRequisitionItemController.cs

```
[HttpPost]
        [Route("{id}/getAllForRequsition")]
        public async Task<IActionResult> GetAllForRequsition(int id, CancellationToken
cancellationToken = default)
        {
            var res = (await this.CrudService.GetAll(cancellationToken)).Where(a =>
a.IsStationery && !(a.IsInCanceledOrder ?? false) && a.PurchaseSummaryId == id);
            //This is not the most elegant solution, but at the moment I do not know
how to map the URL field
            foreach (var item in res)
            {
                var orderItem = this._context.Set<OrderItem>().FirstOrDefault(a =>
a.Id == item.OrderItemId);
                var fileDetails = this._context.Set<FileDetails>().FirstOrDefault(a =>
a.Id == orderItem.FileDetailsId);
                var map = _mapper.Map<FileDetailsDTO>(fileDetails);
                if (map != null)
                {
                    item.URL = map.Url;
                }
            }
            return Ok(res);//Ok(_mapper.Map<PurchaseItemDTO>(t)); //Ok(await
this.CrudService.GetAll(cancellationToken));
        }
```

Change this line,

```
  var res = (await this.CrudService.GetAll(cancellationToken)).Where(a
=> !(a.IsInCanceledOrder ?? false) && a.PurchaseSummaryId == id);
```

mdia-requisitions.component.html

remove line 15,16

```
creationLink2="/app/requisitions/stationery-requisition/0"
creationButtonText2="Add Stationery Requisition"
```

Remove following code segment from mdia-requisitions.component.ts file.

Line 191-199, 278-286

```
  <FilterSettings>{
                    matchMode: CountableFilterMatchMode.Equals,
                    header: "Is Stationery",
                    inputType: InputType.Dropdown,
                    dropdownOptions: [{ label: "Stationery", value: true },
{ label: "Purchase", value: false }, { label: "Both", value: null }],
                    valueFieldName: "isStationery",
                    defaultValue: null,
                    filterType: FilterType.Boolean,
                },
```

Remove line 314-320

```
editRequisition(rowData: IPurchaseSummary) {
        if (rowData.isStationery) {
            this.router.navigate(["/app/requisitions/stationery-requisition/"
+ rowData.id]);
        } else {
            this.router.navigate(["/app/requisitions/purchase-requisition/" +
rowData.id]);
        }
    }
```

Remove lines 99-103

```
<GridColumn>{
                field: "isStationery", header: "Request Type",
displayingHandler: (cellVal, rowData) => {
                    return cellVal ? "Stationery" : "Purchase";
                }
            },
```

Remove, isStationery: boolean; field from following files,

purchaseItems.ts, orderItem.ts, purchaseSummary.ts , stock-item.ts

stock-items-component.ts

remove line 222

purchase-Items.component.ts

remove line 205

purchase-Summary. component.ts

remove line 95

Remove stationeryItem folder.

Restore imessage.d.ts and import.service.ts files

Remove project/ stationeryRequisition folder

Remove Project-routing.ts following lines

Remove 36,37,40

Stock-items-component.ts

Remove line 222

Stock-items-component.html

Remove lines from 133-140

Change 174-182