Controller

package mex.walmart.archivo.central.controllers.administration;  
  
import io.swagger.v3.oas.annotations.media.ArraySchema;  
import io.swagger.v3.oas.annotations.media.Content;  
import io.swagger.v3.oas.annotations.media.Schema;  
import io.swagger.v3.oas.annotations.responses.ApiResponse;  
import io.swagger.v3.oas.annotations.tags.Tag;  
import lombok.RequiredArgsConstructor;  
import lombok.extern.slf4j.Slf4j;  
import mex.walmart.archivo.central.constants.SwaggerConstants;  
import mex.walmart.archivo.central.exception.ArchivoException;  
import mex.walmart.archivo.central.model.request.ArchivoRequest;  
import mex.walmart.archivo.central.model.response.\*;  
import mex.walmart.archivo.central.model.response.Error;  
import mex.walmart.archivo.central.service.administration.ArchivoService;  
import mex.walmart.archivo.central.service.common.CommonService;  
import org.springframework.http.MediaType;  
import org.springframework.validation.annotation.Validated;  
import org.springframework.web.bind.annotation.\*;  
  
import javax.validation.Valid;  
import java.util.Collections;  
import java.util.List;  
  
@RestController  
@Tag(name = "Archivo")  
@RequestMapping("/administration/archivo")  
@RequiredArgsConstructor  
@Slf4j  
@Validated  
public class ArchivoController {  
 final CommonService commonService;  
 final ArchivoService archivoService;  
  
  
 */\*\*  
 \* This GET request allows us to get all types of WTypes  
 \*  
 \** ***@return*** *Response*<*List* < *ArchivoTypeResponse*>*> list of ArchivoTypeResponse object  
 \*/* @GetMapping(value = "/archivoTypes", produces = MediaType.*APPLICATION\_JSON\_VALUE*)  
 @ApiResponse(responseCode = "200", description = SwaggerConstants.*DESCRIPTION\_200* , content = @Content(array = @ArraySchema(arraySchema = @Schema(implementation = ArchivoTypeResponse.class))))  
 public Response<List<ArchivoTypeResponse>> getArchivoTypes() {  
 Response<List<ArchivoTypeResponse>> response = new Response<>();  
 final List<ArchivoTypeResponse> archivoTypeResponses = commonService.getAllArchivoTypes();  
 *log*.info("{} returned", archivoTypeResponses);  
 response.setData(archivoTypeResponses);  
 return response;  
 }  
  
  
 */\*\*  
 \* This endpoint is used to delete the Archivo lists  
 \*  
 \** ***@param*** *number  
 \** ***@return*** *count  
 \*/* @DeleteMapping(value = "/delete", produces = MediaType.*APPLICATION\_JSON\_VALUE*)  
 @ApiResponse(responseCode = "200", description = SwaggerConstants.*DESCRIPTION\_200* , content = @Content(array = @ArraySchema(arraySchema = @Schema(implementation = ArchivoResponse.class))))  
 public Response<List<Integer>> deleteArchivoList(@RequestParam(value = "number") List<String> number) {  
 Response<List<Integer>> response = new Response<>();  
 List<Integer> integers = archivoService.deleteArchivoList(number);  
 *log*.info("{} archivoList deleted", integers);  
 if(integers.size()>0){  
 return new Response<>(false, integers, Collections.*singletonList*(new Error(ArchivoException.Code.*DATA\_CANNOT\_BE\_DELETED\_FROM\_ARCHIVO*)));  
 } else {  
 return new Response<>(true, null, null);  
  
 }  
 }  
  
 */\*\*  
 \* This GET endpoint is used to fetch the Archivo lists  
 \*  
 \** ***@param*** *number  
 \** ***@param*** *description  
 \** ***@return*** *List*<*ArchivoRepsonse*>  
 *\*/* @GetMapping(value = "/search", produces = MediaType.*APPLICATION\_JSON\_VALUE*)  
 @ApiResponse(responseCode = "200", description = SwaggerConstants.*DESCRIPTION\_200* , content = @Content(array = @ArraySchema(arraySchema = @Schema(implementation = ArchivoResponse.class))))  
 public Response<List<ArchivoResponse>> getArchivoList(@RequestParam(value = "number", required = false) String number,  
 @RequestParam(value = "description", required = false) String description) {  
 Response<List<ArchivoResponse>> response = new Response<>();  
 final List<ArchivoResponse> archivoList = archivoService.getArchivoList(number, description);  
 *log*.info("{} archivoList returned", archivoList.size());  
 response.setData(archivoList);  
 return response;  
 }  
  
  
 */\*\*  
 \* Inserts or Updates data from request into 'Warehouse' table  
 \*  
 \** ***@Request*** *ArchivoRequest  
 \*/* @PostMapping(value = "/createOrUpdate", consumes = MediaType.*APPLICATION\_JSON\_VALUE*)  
 @ApiResponse(responseCode = "200", description = SwaggerConstants.*DESCRIPTION\_200*,  
 content = @Content(schema = @Schema()))  
 public Response<Void> newArchivoData(@RequestBody @Valid ArchivoRequest request) {  
 Response<Void> response = new Response<>();  
 archivoService.saveorUpdateArchivosData(request);  
 response.setSuccess(true);  
 *log*.info("Successfully saved data into Warehouses table");  
 return response;  
 }  
  
 */\*\*  
 \* This GET request allows us to get all materials from Materials  
 \*  
 \** ***@return*** *Response*<*List* < *MaterialsServiceResponse*>*> list of MaterialsServiceResponse object  
 \*/* @GetMapping(value = "/getMaterials", produces = MediaType.*APPLICATION\_JSON\_VALUE*)  
 @ApiResponse(responseCode = "200", description = SwaggerConstants.*DESCRIPTION\_200* , content = @Content(array = @ArraySchema(arraySchema = @Schema(implementation = ArchivoTypeResponse.class))))  
 public Response<List<MaterialsServiceResponse>> getMaterials() {  
 Response<List<MaterialsServiceResponse>> response = new Response<>();  
 final List<MaterialsServiceResponse> materials = commonService.getMaterials();  
 *log*.info("{} returned", materials);  
 response.setData(materials);  
 return response;  
 }  
}

Service

package mex.walmart.archivo.central.service.administration;  
  
import lombok.AccessLevel;  
import lombok.RequiredArgsConstructor;  
import lombok.experimental.FieldDefaults;  
import lombok.extern.slf4j.Slf4j;  
import mex.walmart.archivo.central.exception.ArchivoException;  
import mex.walmart.archivo.central.model.entity.Locations;  
import mex.walmart.archivo.central.model.entity.Shelfs;  
import mex.walmart.archivo.central.model.entity.Warehouses;  
import mex.walmart.archivo.central.model.entity.projections.ArchivoProjection;  
import mex.walmart.archivo.central.model.mapper.ArchivoMapper;  
import mex.walmart.archivo.central.model.request.ArchivoRequest;  
import mex.walmart.archivo.central.model.response.ArchivoResponse;  
import mex.walmart.archivo.central.repositorys.LocationsRepository;  
import mex.walmart.archivo.central.repositorys.ShelfsRepository;  
import mex.walmart.archivo.central.repositorys.WarehousesRepository;  
import mex.walmart.archivo.central.repositorys.dao.ArchivoDao;  
import org.apache.commons.lang3.StringUtils;  
import org.springframework.stereotype.Service;  
import org.springframework.util.CollectionUtils;  
import org.springframework.util.ObjectUtils;  
  
import javax.transaction.Transactional;  
import java.util.ArrayList;  
import java.util.Collections;  
import java.util.List;  
import java.util.Optional;  
import java.util.stream.Collectors;  
  
@Service  
@Slf4j  
@FieldDefaults(level = AccessLevel.*PRIVATE*)  
@RequiredArgsConstructor  
public class ArchivoService {  
 final ArchivoDao archivoDao;  
 final ArchivoMapper archivoMapper;  
 final ShelfsRepository shelfsRepository;  
 final WarehousesRepository warehousesRepository;  
 final LocationsRepository locationsRepository;  
  
 */\*\*  
 \* Gets all warehouses from Warehouse Table based on id and description  
 \** ***@param*** *number number  
 \** ***@param*** *description description  
 \** ***@return*** *List*<*ArchivoResponse*>  
 *\*/* public List<ArchivoResponse> getArchivoList(String number, String description) {  
 List<ArchivoProjection> objects = archivoDao.searchArchivoList(number, description);  
 if (CollectionUtils.*isEmpty*(objects))  
 throw new ArchivoException(ArchivoException.Code.*DATA\_NOT\_FOUND*);  
 return archivoMapper.mapArchivoDaoToArchivoResponse(objects);  
 }  
  
 */\*\*  
 \* Delete warehouses based on warehouse Id. If warehouse has shelfs, then do not delete warehouse.  
 \** ***@param*** *number id  
 \** ***@return*** *List*<*Integer*>  
 *\*/* //*TODO* @Transactional  
 public List<Integer> deleteArchivoList(List<String> number) {  
 List<Integer> errorList = new ArrayList<>();  
 List<Integer> numbers = number.parallelStream().map(Integer::*parseInt*).collect(Collectors.*toList*());  
  
 List<Shelfs> shelfsList = shelfsRepository.getShelfsCount(numbers);  
 List<Integer> integerShelf = shelfsList.stream().map(shelfs -> shelfs.getEmbeddedShelfsId().getWarehouses().getWarehouseId()).distinct().collect(Collectors.*toList*());  
 if(!integerShelf.isEmpty()){  
 errorList.addAll(integerShelf);  
 numbers.removeAll(integerShelf);  
 }  
  
 List<Locations> locationsList = locationsRepository.getLocationsCount(numbers);  
 List<Integer> integerLocation = locationsList.stream().map(locations -> locations.getWarehouses().getWarehouseId()).distinct().collect(Collectors.*toList*());  
 if(!integerLocation.isEmpty()){  
 errorList.addAll(integerLocation);  
 numbers.removeAll(integerLocation);  
 }  
  
 warehousesRepository.deleteAllById(numbers);  
 return errorList;  
 }  
  
 */\*\*  
 \* Create or update warehouse  
 \** ***@param*** *request  
 \*/* @Transactional  
 public void saveorUpdateArchivosData(ArchivoRequest request) {  
  
 if (ObjectUtils.*isEmpty*(request.getArchivoNumber()) || request.getArchivoNumber() == 0) {  
 // Create  
 Warehouses warehouses = archivoMapper.mapRequestToWarehouse(request);  
 warehousesRepository.save(warehouses);  
 } else {  
 // Update  
 Optional<Warehouses> byId = Optional.*ofNullable*(warehousesRepository.findById(request.getArchivoNumber())  
 .orElseThrow(() -> new ArchivoException(ArchivoException.Code.*DATA\_NOT\_FOUND*)));  
 if(byId.isPresent())  
 warehousesRepository.save(archivoMapper.mapRequestToWarehouse(request));  
 }  
  
 }  
  
}

DAO

package mex.walmart.archivo.central.repositorys.dao;  
  
import io.micrometer.core.instrument.util.StringUtils;  
import mex.walmart.archivo.central.model.entity.projections.ArchivoProjection;  
import mex.walmart.archivo.central.utility.Santize;  
import org.springframework.stereotype.Component;  
  
import javax.persistence.EntityManager;  
import javax.persistence.PersistenceContext;  
import java.util.List;  
  
@Component  
public class ArchivoDaoImpl implements ArchivoDao {  
  
 @PersistenceContext  
 private EntityManager em;  
  
 @Override  
 public List<ArchivoProjection> searchArchivoList(String number, String description) {  
 String requestParam = StringUtils.*isNotEmpty*(number) ? "number" : StringUtils.*isNotEmpty*(description)?"description" : "";  
 StringBuilder stringBuilder;  
 List<ArchivoProjection> resultList = null;  
 switch (requestParam) {  
 case "number":  
 stringBuilder = new StringBuilder("Select new mex.walmart.archivo.central.model.entity.projections.ArchivoProjection( " +  
 "W.warehouseId as archivoNumber, RTRIM(W.warehouseName) as archivoDescription ," +  
 "RTRIM(T.description) as type,RTRIM(M.materialDescription) as material, " +  
 "RTRIM(W.warehouseAddress) as warehouseAddress,W.warehouseShelfs as warehouseShelfs,W.warehouseCapacity as warehouseCapacity, W.typeDetail as typeDetail ) " +  
 "from Warehouses W,WTypes T,Materials M " +  
 "Where W.warehouseId = :number" +  
 " AND W.types.wtypeId=T.wtypeId AND W.materials.materialId=M.materialId Order By W.warehouseId");  
 resultList = em.createQuery(stringBuilder.toString(),ArchivoProjection.class).setParameter("number",Integer.*parseInt*(number)).getResultList();  
 break;  
  
 case "description":  
 String queryParam = new StringBuilder(description).insert(0, "%").append("%").toString();  
 stringBuilder = new StringBuilder("Select new mex.walmart.archivo.central.model.entity.projections.ArchivoProjection( " +  
 "W.warehouseId as archivoNumber, RTRIM(W.warehouseName) as archivoDescription ," +  
 "RTRIM(T.description) as type,RTRIM(M.materialDescription) as material, " +  
 "RTRIM(W.warehouseAddress) as warehouseAddress,W.warehouseShelfs as warehouseShelfs,W.warehouseCapacity as warehouseCapacity, W.typeDetail as typeDetail) " +  
 "from Warehouses W,WTypes T,Materials M " +  
 "Where W.warehouseName like :description" +  
 " AND W.types.wtypeId=T.wtypeId AND W.materials.materialId=M.materialId Order By W.warehouseId");  
 resultList = em.createQuery(stringBuilder.toString(),ArchivoProjection.class).setParameter("description", queryParam).getResultList();  
 break;  
  
 default:  
 stringBuilder = new StringBuilder("Select new mex.walmart.archivo.central.model.entity.projections.ArchivoProjection( " +  
 "W.warehouseId as archivoNumber, RTRIM(W.warehouseName) as archivoDescription ," +  
 "RTRIM(T.description) as type,RTRIM(M.materialDescription) as material, " +  
 "RTRIM(W.warehouseAddress) as warehouseAddress,W.warehouseShelfs as warehouseShelfs,W.warehouseCapacity as warehouseCapacity, W.typeDetail as typeDetail) " +  
 "from Warehouses W,WTypes T,Materials M " +  
 "WHERE W.types.wtypeId=T.wtypeId AND W.materials.materialId=M.materialId Order By W.warehouseId");  
 resultList = em.createQuery(stringBuilder.toString(),ArchivoProjection.class).getResultList();  
 break;  
 }  
 List<ArchivoProjection> dbResult = new Santize<List<ArchivoProjection>>(resultList).getData();  
 return dbResult;  
 }  
  
}

Projection

package mex.walmart.archivo.central.model.entity.projections;  
  
import lombok.AllArgsConstructor;  
import lombok.Getter;  
import lombok.Setter;  
  
import java.math.BigDecimal;  
import java.time.LocalDate;  
  
@AllArgsConstructor  
@Getter  
@Setter  
public class ArchivoProjection {  
 Integer archivoNumber;  
 String archivoDescription;  
 String type;  
 String material;  
 String warehouseAddress;  
 Short warehouseShelfs;  
 Integer warehouseCapacity;  
 String typeDetail;  
}

Mapper

package mex.walmart.archivo.central.model.mapper;  
  
import mex.walmart.archivo.central.model.entity.Warehouses;  
import mex.walmart.archivo.central.model.entity.projections.ArchivoProjection;  
import mex.walmart.archivo.central.model.request.ArchivoRequest;  
import mex.walmart.archivo.central.model.response.ArchivoResponse;  
import org.apache.commons.lang3.StringUtils;  
import org.mapstruct.Mapper;  
import org.mapstruct.Mapping;  
import org.springframework.stereotype.Component;  
  
import java.util.List;  
  
@Component  
@Mapper(componentModel = "spring",imports = StringUtils.class)  
public interface ArchivoMapper {  
  
 @Mapping(target="archivoNumber",expression = "java(StringUtils.leftPad(archivo.getArchivoNumber().toString(),7,\"0\"))")  
 @Mapping(target="warehouseShelfs",expression = "java(archivo.getWarehouseShelfs() != null ? archivo.getWarehouseShelfs() : new Integer((short) 0))")  
 @Mapping(target="warehouseCapacity",expression = "java(archivo.getWarehouseCapacity() == null ? 0 : archivo.getWarehouseCapacity())")  
 ArchivoResponse mapArchivoDaoToArchivoResponse(ArchivoProjection archivo);  
 List<ArchivoResponse> mapArchivoDaoToArchivoResponse(List<ArchivoProjection> archivoList);  
  
 @Mapping(source = "archivoNumber",target = "warehouseId")  
 @Mapping(source = "archivoDescription",target = "warehouseName")  
 @Mapping(source = "warehouseAddress",target = "warehouseAddress")  
 @Mapping(source = "warehouseShelfs",target = "warehouseShelfs")  
 @Mapping(source = "material",target = "materials.materialId")  
 @Mapping(source = "type",target = "types.wtypeId")  
 @Mapping(source = "warehouseCapacity",target = "warehouseCapacity")  
 @Mapping(source = "warehouseDetail",target = "typeDetail")  
 Warehouses mapRequestToWarehouse(ArchivoRequest request);  
  
}

Response

package mex.walmart.archivo.central.model.response;  
  
import com.fasterxml.jackson.annotation.JsonInclude;  
import lombok.AllArgsConstructor;  
import lombok.Getter;  
import lombok.NoArgsConstructor;  
import lombok.Setter;  
  
@Getter  
@Setter  
@AllArgsConstructor  
@NoArgsConstructor  
public class ArchivoResponse {  
 String archivoNumber;  
 String archivoDescription;  
 String type;  
 String material;  
 String warehouseAddress;  
 Integer warehouseShelfs;  
 Integer warehouseCapacity;  
 String typeDetail;  
}

Controller

package mex.walmart.archivo.central.controllers.administration.catalogs;  
  
import io.swagger.v3.oas.annotations.media.ArraySchema;  
import io.swagger.v3.oas.annotations.media.Content;  
import io.swagger.v3.oas.annotations.media.Schema;  
import io.swagger.v3.oas.annotations.responses.ApiResponse;  
import io.swagger.v3.oas.annotations.tags.Tag;  
import lombok.RequiredArgsConstructor;  
import lombok.extern.slf4j.Slf4j;  
import mex.walmart.archivo.central.constants.SwaggerConstants;  
import mex.walmart.archivo.central.model.entity.Division;  
import mex.walmart.archivo.central.model.enums.CrudOperationEnum;  
import mex.walmart.archivo.central.model.request.DocumentDivisionDeleteRequest;  
import mex.walmart.archivo.central.model.request.DocumentDivisionRequest;  
import mex.walmart.archivo.central.model.response.\*;  
import mex.walmart.archivo.central.service.administration.catalogs.DocumentosDivisionService;  
import mex.walmart.archivo.central.service.common.CommonService;  
import org.springframework.http.MediaType;  
import org.springframework.validation.annotation.Validated;  
import org.springframework.web.bind.annotation.\*;  
  
import javax.validation.Valid;  
import java.util.Comparator;  
import java.util.List;  
  
@RestController  
@Tag(name = "Documento Division")  
@RequestMapping("/administration/catalog/documentodivision")  
@RequiredArgsConstructor  
@Slf4j  
@Validated  
public class DocumentosDivisionController {  
 final CommonService commonService;  
 final DocumentosDivisionService documentosDivisionService;  
  
  
 */\*\*  
 \* This GET request allows us to get all divisions  
 \*  
 \** ***@return*** *Response*<*List* < *DivisionResponse*>*> list of DivisionResponse object  
 \*/* @GetMapping(value = "/getAllDivisions", produces = MediaType.*APPLICATION\_JSON\_VALUE*)  
 @ApiResponse(responseCode = "200", description = SwaggerConstants.*DESCRIPTION\_200* , content = @Content(array = @ArraySchema(arraySchema = @Schema(implementation = DivisionResponse.class))))  
 public Response<List<DivisionResponse>> getAllDivisions() {  
 Response<List<DivisionResponse>> response = new Response<>();  
 final List<DivisionResponse> divisionResponses = commonService.getAllDivisions(Comparator.*comparing*(Division::getDivisionDescription));  
 *log*.info("{} returned", divisionResponses);  
 response.setData(divisionResponses);  
 return response;  
 }  
  
 */\*\*  
 \* This GET request allows us to get all documents  
 \*  
 \** ***@return*** *Response*<*List* < *DocumentResponse*>*> list of DocumentResponse object  
 \*/* @GetMapping(value = "/getAllDocuments", produces = MediaType.*APPLICATION\_JSON\_VALUE*)  
 @ApiResponse(responseCode = "200", description = SwaggerConstants.*DESCRIPTION\_200* , content = @Content(array = @ArraySchema(arraySchema = @Schema(implementation = DivisionResponse.class))))  
 public Response<List<DocumentResponse>> getAllDocuments() {  
 Response<List<DocumentResponse>> response = new Response<>();  
 List<DocumentResponse> documents = commonService.getDocuments();  
 *log*.info("{} returned", documents);  
 response.setData(documents);  
 return response;  
 }  
  
 */\*\*  
 \* This GET request allows us to get all force  
 \*  
 \** ***@return*** *Response*<*List* < *ForceServiceResponse*>*> list of ForceServiceResponse object  
 \*/* @GetMapping(value = "/getAllForce", produces = MediaType.*APPLICATION\_JSON\_VALUE*)  
 @ApiResponse(responseCode = "200", description = SwaggerConstants.*DESCRIPTION\_200* , content = @Content(array = @ArraySchema(arraySchema = @Schema(implementation = DivisionResponse.class))))  
 public Response<List<ForceServiceResponse>> getAllForce() {  
 Response<List<ForceServiceResponse>> response = new Response<>();  
 List<ForceServiceResponse> forceServiceResponses = commonService.getAllForce();  
 *log*.info("{} returned", forceServiceResponses);  
 response.setData(forceServiceResponses);  
 return response;  
 }  
  
 */\*\*  
 \* This GET request allows us to get all documents based on division  
 \*  
 \** ***@return*** *Response*<*List* < *DocumentosDivisionResponse*>*> list of ForceServiceResponse object  
 \*/* @GetMapping(value = "/getDocumentosDivision", produces = MediaType.*APPLICATION\_JSON\_VALUE*)  
 @ApiResponse(responseCode = "200", description = SwaggerConstants.*DESCRIPTION\_200* , content = @Content(array = @ArraySchema(arraySchema = @Schema(implementation = DivisionResponse.class))))  
 public Response<List<DocumentosDivisionResponse>> getDocumentosDivision(@RequestParam(value = "division", required = false) Integer divisionId,@RequestParam(value = "company", required = false) Integer companyId) {  
 Response<List<DocumentosDivisionResponse>> response = new Response<>();  
 List<DocumentosDivisionResponse> documentosDivisionResponses = documentosDivisionService.getDocumentosDivisionList(divisionId,companyId);  
 *log*.info("{} returned", documentosDivisionResponses);  
 response.setData(documentosDivisionResponses);  
 return response;  
 }  
  
 */\*\*  
 \* Inserts data from request into 'Docdiv' table  
 \*  
 \** ***@Request*** *DocumentDivisionRequest  
 \*/* @PostMapping(value = "/create", consumes = MediaType.*APPLICATION\_JSON\_VALUE*)  
 @ApiResponse(responseCode = "200", description = SwaggerConstants.*DESCRIPTION\_200*,  
 content = @Content(schema = @Schema()))  
 public Response<Void> createDocumentDivision(@RequestBody @Valid DocumentDivisionRequest request) {  
 Response<Void> response = new Response<>();  
 documentosDivisionService.saveorUpdateDocumentData(request, CrudOperationEnum.*SAVE*);  
 response.setSuccess(true);  
 *log*.info("Successfully saved data into Docdiv table");  
 return response;  
 }  
  
 */\*\*  
 \* Updates data from request into 'Docdiv' table  
 \*  
 \** ***@Request*** *DocumentDivisionRequest  
 \*/* @PostMapping(value = "/update", consumes = MediaType.*APPLICATION\_JSON\_VALUE*)  
 @ApiResponse(responseCode = "200", description = SwaggerConstants.*DESCRIPTION\_200*,  
 content = @Content(schema = @Schema()))  
 public Response<Void> updateDocumentDivision(@RequestBody @Valid DocumentDivisionRequest request) {  
 Response<Void> response = new Response<>();  
 documentosDivisionService.saveorUpdateDocumentData(request, CrudOperationEnum.*UPDATE*);  
 response.setSuccess(true);  
 *log*.info("Successfully updated data into Docdiv table");  
 return response;  
 }  
  
 */\*\*  
 \* This endpoint is used to delete the document lists  
 \*  
 \*/* @DeleteMapping(value = "/delete", produces = MediaType.*APPLICATION\_JSON\_VALUE*)  
 @ApiResponse(responseCode = "200", description = SwaggerConstants.*DESCRIPTION\_200* , content = @Content(array = @ArraySchema(arraySchema = @Schema(implementation = DocumentosDivisionResponse.class))))  
 public Response deleteDocumentList(@RequestBody @Valid List<DocumentDivisionDeleteRequest> request) {  
 Response response = new Response<>();  
 documentosDivisionService.deleteDocumentosList(request);  
 *log*.info("documentlist deleted");  
 return new Response(true, null,null);  
 }  
  
}

package mex.walmart.archivo.central.service.administration.catalogs;  
  
import lombok.AccessLevel;  
import lombok.RequiredArgsConstructor;  
import lombok.experimental.FieldDefaults;  
import lombok.extern.slf4j.Slf4j;  
import mex.walmart.archivo.central.exception.ArchivoException;  
import mex.walmart.archivo.central.model.entity.DocDiv;  
import mex.walmart.archivo.central.model.entity.projections.DocumentosDivisionProjection;  
import mex.walmart.archivo.central.model.enums.CrudOperationEnum;  
import mex.walmart.archivo.central.model.mapper.DocumentosDivisionMapper;  
import mex.walmart.archivo.central.model.request.DocumentDivisionDeleteRequest;  
import mex.walmart.archivo.central.model.request.DocumentDivisionRequest;  
import mex.walmart.archivo.central.model.response.DocumentosDivisionResponse;  
import mex.walmart.archivo.central.repositorys.DocDivRepository;  
import mex.walmart.archivo.central.repositorys.dao.DocumentosDivisionDao;  
import org.springframework.stereotype.Service;  
import org.springframework.util.CollectionUtils;  
import org.springframework.validation.annotation.Validated;  
  
import javax.transaction.Transactional;  
import java.util.List;  
import java.util.Optional;  
  
@Service  
@Slf4j  
@FieldDefaults(level = AccessLevel.*PRIVATE*)  
@Validated  
@RequiredArgsConstructor  
public class DocumentosDivisionService {  
 final DocumentosDivisionDao documentosDivisionDao;  
 final DocumentosDivisionMapper documentosDivisionMapper;  
 final DocDivRepository docDivRepository;  
  
  
 */\*\*  
 \* Gets all documents from Documents and Division Table based on division  
 \** ***@param*** *divisionId divisionId  
 \** ***@param*** *companyId companyId  
 \** ***@return*** *List*<*DocumentosDivisionResponse*>  
 *\*/* public List<DocumentosDivisionResponse> getDocumentosDivisionList(Integer divisionId,Integer companyId) {  
 List<DocumentosDivisionProjection> objects = documentosDivisionDao.searchDocumentosDivisionList(divisionId,companyId);  
 if (CollectionUtils.*isEmpty*(objects))  
 throw new ArchivoException(ArchivoException.Code.*DATA\_NOT\_FOUND*);  
 return documentosDivisionMapper.mapDocumentoDivisionDaoToDocumentosDivisionResponse(objects);  
 }  
  
 */\*\*  
 \* Delete documents based on docid, divid and companyid  
 \** ***@param*** *requests requests  
 \*/* public void deleteDocumentosList(List<DocumentDivisionDeleteRequest> requests) {  
 requests.forEach(request ->  
 docDivRepository.deleteByDivisionIdAndDocumentId(request.getDocumentId(),  
 request.getDivisionId(),  
 request.getCompanyId()));  
 }  
  
 */\*\*  
 \* Creates or updates document lists.  
 \** ***@param*** *request request  
 \** ***@param*** *value value  
 \*/* @Transactional  
 public void saveorUpdateDocumentData(DocumentDivisionRequest request, CrudOperationEnum value) {  
 if (value.equals(CrudOperationEnum.*SAVE*)) {  
 // Create  
 docDivRepository.save(documentosDivisionMapper.mapRequestToDocDiv(request));  
 } else {  
 // Update  
 Optional<DocDiv> byId = Optional.*ofNullable*(docDivRepository.findByEmbededDocumentoDivisionId(request.getDivisionId(),request.getDocumentId(),request.getCompanyId())  
 .orElseThrow(() -> new ArchivoException(ArchivoException.Code.*DATA\_NOT\_FOUND*)));  
 if(byId.isPresent())  
 docDivRepository.save(documentosDivisionMapper.mapRequestToDocDiv(request));  
 }  
  
 }  
}

Repo

package mex.walmart.archivo.central.repositorys.dao;  
  
import io.micrometer.core.instrument.util.StringUtils;  
import mex.walmart.archivo.central.model.entity.projections.ArchivoProjection;  
import mex.walmart.archivo.central.model.entity.projections.DocumentosDivisionProjection;  
import mex.walmart.archivo.central.utility.Santize;  
import org.springframework.stereotype.Component;  
  
import javax.persistence.EntityManager;  
import javax.persistence.PersistenceContext;  
import java.util.List;  
  
@Component  
public class DocumentosDivisionDaoImpl implements DocumentosDivisionDao {  
  
 @PersistenceContext  
 private EntityManager em;  
  
 @Override  
 public List<DocumentosDivisionProjection> searchDocumentosDivisionList(Integer divisionId,Integer companyId) {  
 String requestParam = divisionId != null ? "divisionId" : "";  
 StringBuilder stringBuilder;  
 List<DocumentosDivisionProjection> resultList = null;  
 switch (requestParam) {  
 case "divisionId":  
 stringBuilder = new StringBuilder("Select new mex.walmart.archivo.central.model.entity.projections.DocumentosDivisionProjection( " +  
 " D.embededDocumentoDivisionId.documentId as documentId,D.embededDocumentoDivisionId.divisionId as divisionId," +  
 "D.embededDocumentoDivisionId.companyId as companyId,RTRIM(Do.documentDescription) as documentName,RTRIM(Di.divisionDescription) as division,F.forceDays as forceDays )" +  
 "from DocDiv D, Division Di,Force F,Documents Do Where D.embededDocumentoDivisionId.divisionId= Di.embededDivisionId.divisionId " +  
 "AND Di.embededDivisionId.divisionId=:divisionId AND D.force.forceId=F.forceId " +  
 "And D.embededDocumentoDivisionId.companyId=Di.embededDivisionId.company.companyId " +  
 "AND Di.embededDivisionId.company.companyId=:companyId " +  
 "And D.embededDocumentoDivisionId.documentId=Do.documentId order By D.embededDocumentoDivisionId.divisionId,D.embededDocumentoDivisionId.companyId,D.embededDocumentoDivisionId.documentId" );  
 resultList = em.createQuery(stringBuilder.toString(),DocumentosDivisionProjection.class).setParameter("divisionId",divisionId).setParameter("companyId",companyId).getResultList();  
 break;  
  
 default:  
 stringBuilder = new StringBuilder("Select new mex.walmart.archivo.central.model.entity.projections.DocumentosDivisionProjection( " +  
 " D.embededDocumentoDivisionId.documentId as documentId,D.embededDocumentoDivisionId.divisionId as divisionId," +  
 "D.embededDocumentoDivisionId.companyId as companyId,RTRIM(Do.documentDescription) as documentName,RTRIM(Di.divisionDescription) as division,F.forceDays as forceDays )" +  
 "from DocDiv D, Division Di,Force F,Documents Do Where D.embededDocumentoDivisionId.divisionId=Di.embededDivisionId.divisionId " +  
 "AND D.force.forceId=F.forceId " + "And D.embededDocumentoDivisionId.companyId=Di.embededDivisionId.company.companyId " +  
 "And D.embededDocumentoDivisionId.documentId=Do.documentId order By D.embededDocumentoDivisionId.divisionId,D.embededDocumentoDivisionId.companyId,D.embededDocumentoDivisionId.documentId" );  
 resultList = em.createQuery(stringBuilder.toString(),DocumentosDivisionProjection.class).getResultList();  
 break;  
 }  
 List<DocumentosDivisionProjection> dbResult = new Santize<List<DocumentosDivisionProjection>>(resultList).getData();  
 return dbResult;  
 }  
  
}

Projection

package mex.walmart.archivo.central.model.entity.projections;  
  
import lombok.AllArgsConstructor;  
import lombok.Getter;  
import lombok.Setter;  
  
@AllArgsConstructor  
@Getter  
@Setter  
public class DocumentosDivisionProjection {  
 Integer documentId;  
 Integer divisionId;  
 Integer companyId;  
 String documentName;  
 String division;  
 Integer forceDays;  
}

Mapper

package mex.walmart.archivo.central.model.mapper;  
  
import mex.walmart.archivo.central.model.entity.DocDiv;  
import mex.walmart.archivo.central.model.entity.Warehouses;  
import mex.walmart.archivo.central.model.entity.projections.ArchivoProjection;  
import mex.walmart.archivo.central.model.entity.projections.DocumentosDivisionProjection;  
import mex.walmart.archivo.central.model.request.ArchivoRequest;  
import mex.walmart.archivo.central.model.request.DocumentDivisionRequest;  
import mex.walmart.archivo.central.model.response.ArchivoResponse;  
import mex.walmart.archivo.central.model.response.DocumentosDivisionResponse;  
import org.apache.commons.lang3.StringUtils;  
import org.mapstruct.Mapper;  
import org.mapstruct.Mapping;  
import org.springframework.stereotype.Component;  
  
import java.util.List;  
  
@Component  
@Mapper(componentModel = "spring",imports = StringUtils.class)  
public interface DocumentosDivisionMapper {  
  
 @Mapping(target="documentNumber",expression = "java(StringUtils.leftPad(divisionProjection.getDocumentId().toString(),7,\"0\"))")  
 DocumentosDivisionResponse mapDocumentoDivisionDaoToDocumentosDivisionResponse(DocumentosDivisionProjection divisionProjection);  
 List<DocumentosDivisionResponse> mapDocumentoDivisionDaoToDocumentosDivisionResponse(List<DocumentosDivisionProjection> divisionProjections);  
  
 @Mapping(source="documentId", target="embededDocumentoDivisionId.documentId")  
 @Mapping(source="divisionId", target="embededDocumentoDivisionId.divisionId")  
 @Mapping(source="companyId", target="embededDocumentoDivisionId.companyId")  
 @Mapping(source="forceId", target="force.forceId")  
 @Mapping(source="forceDays", target="force.forceDays")  
 DocDiv mapRequestToDocDiv(DocumentDivisionRequest request);  
  
}

Response

package mex.walmart.archivo.central.model.response;  
  
import lombok.AllArgsConstructor;  
import lombok.Getter;  
import lombok.NoArgsConstructor;  
import lombok.Setter;  
  
@Getter  
@Setter  
@AllArgsConstructor  
@NoArgsConstructor  
public class DocumentosDivisionResponse {  
 String documentNumber;  
 String documentName;  
 String division;  
 Integer divisionId;  
 Integer forceDays;  
}

DAO Object

package mex.walmart.archivo.central.repositorys.dao;  
  
import org.springframework.stereotype.Repository;  
  
import javax.persistence.EntityManager;  
import javax.persistence.PersistenceContext;  
import java.util.List;  
  
@Repository  
public class CambioDeEstatusDaoImpl implements CambioDeEstatusDao {  
  
 @PersistenceContext  
 private EntityManager em;  
  
 @Override  
 public List<Object[]> getStatusList() {  
  
 StringBuffer sfHql = new StringBuffer("select distinct\n" +  
 "CONCAT( b.box\_yearb , '-',REPLICATE('0',7 - LEN(b.box\_fuc)) ,b.box\_fuc) as cashRegister,cia.cia\_code," +  
 "d.div\_cvediv,chr.chr\_desc,doc.doc\_desc,s.shf\_datedoc,dp.dep\_desc,\n" +  
 "st.sta\_desc,w.whs\_name " + //,concat(l.shl\_idshl,'-',l.loc\_position,'-',l.loc\_level) as location \n" +  
 "from sheafs s \n" +  
 "inner join boxes b \n" +  
 "on b.box\_fuc=s.box\_fuc and b.box\_yearb=s.box\_yearb and b.sta\_idsta=s.sta\_idsta \n" +  
 "Left Join Subsystems Sb \n" +  
 "on s.Sub\_IdSub=Sb.Sub\_IdSub \n" +  
 " Left Join Company Cia \n" +  
 " on S.Cia\_IdCia=Cia.Cia\_IdCia \n" +  
 " Left Join Characters Chr \n" +  
 "on S.Chr\_IdChr=Chr.Chr\_IdChr \n" +  
 "Left Join Documents Doc \n" +  
 " on S.Doc\_IdDoc=Doc.Doc\_IdDoc \n" +  
 "Left Join Deparment Dp \n" +  
 "on Dp.Dir\_IdDir=S.Dir\_IdDir and \n" +  
 "Dp.Dep\_IdDep=S.Dep\_IdDep \n" +  
 "join Division d \n" +  
 "on S.Div\_IdDiv=D.Div\_IdDiv and S.Cia\_IdCia=D.Cia\_IdCia \n" +  
 "join locations l on l.box\_yearb=s.box\_yearb \n" +  
 "join warehouses w on l.whs\_idwhs=w.whs\_idwhs \n" +  
 "join status st on s.sta\_idsta=st.sta\_idsta \n" +  
 "where b.sta\_idsta=8");  
  
 return em.createNativeQuery(sfHql.toString()).getResultList();  
 }  
}

Service

package mex.walmart.archivo.central.service.process;  
  
import lombok.AccessLevel;  
import lombok.RequiredArgsConstructor;  
import lombok.experimental.FieldDefaults;  
import lombok.extern.slf4j.Slf4j;  
import mex.walmart.archivo.central.exception.ArchivoException;  
import mex.walmart.archivo.central.model.entity.Sheafs;  
import mex.walmart.archivo.central.model.mapper.CambioDeEstatusMapper;  
import mex.walmart.archivo.central.model.request.CambioDeEstatusRequest;  
import mex.walmart.archivo.central.model.response.CambioDeEstatusResponse;  
import mex.walmart.archivo.central.repositorys.BoxesRepository;  
import mex.walmart.archivo.central.repositorys.LocationsRepository;  
import mex.walmart.archivo.central.repositorys.SheafsRepository;  
import mex.walmart.archivo.central.repositorys.dao.CambioDeEstatusDao;  
import org.springframework.stereotype.Service;  
import org.springframework.util.CollectionUtils;  
  
import javax.transaction.Transactional;  
import java.util.List;  
import java.util.Optional;  
  
@Service  
@Slf4j  
@FieldDefaults(level = AccessLevel.*PRIVATE*)  
@RequiredArgsConstructor  
public class CambioDeEstatusService {  
 final CambioDeEstatusMapper cambioDeEstatusMapper;  
 final CambioDeEstatusDao cambioDeEstatusDao;  
 final SheafsRepository sheafsRepository;  
 final BoxesRepository boxesRepository;  
 final LocationsRepository locationsRepository;  
  
 public List<CambioDeEstatusResponse> getStatusList() {  
 var objects = cambioDeEstatusDao.getStatusList();  
 if (CollectionUtils.*isEmpty*(objects))  
 throw new ArchivoException(ArchivoException.Code.*DATA\_NOT\_FOUND*);  
 return cambioDeEstatusMapper.mapCambioDeEstatusDaoToCambioResponse(objects);  
 }  
  
  
 */\*\*  
 \* This endpoint is used to update the status id to 21 in Sheafs, Boxes and Locations table.  
 \** ***@param*** *request request  
 \*/* @Transactional  
 public void actualize(CambioDeEstatusRequest request) {  
  
 request.getCashRegister().forEach(s -> {  
 var byBoxFucAndBoxYear = sheafsRepository.findByBoxFucAndBoxYear(s.getBoxFUC(), s.getBoxYear());  
 Optional.*of*(byBoxFucAndBoxYear).orElseThrow(()->new ArchivoException(ArchivoException.Code.*DATA\_NOT\_FOUND*));  
 sheafsRepository.updateStatusIdForEstatus(s.getBoxYear(),s.getBoxFUC());  
  
 var totalBoxSheafs = boxesRepository.getTotalBoxSheafs(s.getBoxYear(), s.getBoxFUC());  
 Optional.*of*(totalBoxSheafs).orElseThrow(()->new ArchivoException(ArchivoException.Code.*DATA\_NOT\_FOUND*));  
 boxesRepository.updateStatusIdForEstatus(s.getBoxYear(),s.getBoxFUC());  
  
 var count = locationsRepository.getCount(s.getBoxYear(), s.getBoxFUC());  
 Optional.*of*(count).orElseThrow(()->new ArchivoException(ArchivoException.Code.*DATA\_NOT\_FOUND*));  
 locationsRepository.updateStatusIdForEstatus(s.getBoxYear(),s.getBoxFUC());  
 });  
 }  
}

Mapper

package mex.walmart.archivo.central.model.mapper;  
  
import mex.walmart.archivo.central.model.response.CambioDeEstatusResponse;  
import org.mapstruct.Mapper;  
import org.springframework.stereotype.Component;  
  
import java.util.ArrayList;  
import java.util.List;  
  
@Component  
@Mapper(componentModel = "spring")  
public interface CambioDeEstatusMapper {  
  
 default String replaceNullWithEmptyString(Object obj){  
 return obj!=null?obj.toString():"";  
 }  
 default List<CambioDeEstatusResponse> mapCambioDeEstatusDaoToCambioResponse(List<Object[]> objects) {  
 List<CambioDeEstatusResponse> responses = new ArrayList<>();  
 objects.forEach(object -> {  
 CambioDeEstatusResponse response = new CambioDeEstatusResponse();  
 response.setCashRegister(replaceNullWithEmptyString(object[0]).trim());  
 response.setCompany(replaceNullWithEmptyString(object[1]).trim());  
 response.setDivision(replaceNullWithEmptyString(object[2]).trim());  
 response.setCharacter(replaceNullWithEmptyString(object[3]).trim());  
 response.setDocument(replaceNullWithEmptyString(object[4]).trim());  
 response.setLearnedDate(replaceNullWithEmptyString(object[5]).trim());  
 response.setDepartment(replaceNullWithEmptyString(object[6]).trim());  
 response.setStatus(replaceNullWithEmptyString(object[7]).trim());  
 response.setArchiveDesc(replaceNullWithEmptyString(object[8]).trim());  
 // response.setLocation(replaceNullWithEmptyString(object[9]).trim());  
 responses.add(response);  
 });  
 return responses;  
 }  
}

Testcase

package mex.walmart.archivo.central.service;  
  
  
import mex.walmart.archivo.central.exception.ArchivoException;  
import mex.walmart.archivo.central.model.entity.\*;  
import mex.walmart.archivo.central.model.entity.projections.ArchivoProjection;  
import mex.walmart.archivo.central.model.mapper.ArchivoMapper;  
import mex.walmart.archivo.central.model.request.ArchivoRequest;  
import mex.walmart.archivo.central.model.response.ArchivoResponse;  
import mex.walmart.archivo.central.repositorys.LocationsRepository;  
import mex.walmart.archivo.central.repositorys.ShelfsRepository;  
import mex.walmart.archivo.central.repositorys.WarehousesRepository;  
import mex.walmart.archivo.central.repositorys.dao.ArchivoDao;  
import mex.walmart.archivo.central.service.administration.ArchivoService;  
import org.apache.commons.lang3.StringUtils;  
import org.junit.Ignore;  
import org.junit.Test;  
import org.junit.jupiter.api.Assertions;  
import org.junit.runner.RunWith;  
import org.mockito.InjectMocks;  
import org.mockito.Mock;  
import org.mockito.Mockito;  
import org.springframework.test.context.junit4.SpringRunner;  
  
import java.util.Arrays;  
import java.util.Collections;  
import java.util.List;  
import java.util.Optional;  
import java.util.function.Supplier;  
import java.util.stream.Collectors;  
  
@RunWith(SpringRunner.class)  
public class ArchivoServiceTest {  
  
 @InjectMocks  
 ArchivoService archivoService;  
  
 @Mock  
 ArchivoDao archivoDao;  
  
 @Mock  
 ArchivoMapper archivoMapper;  
  
 @Mock  
 WarehousesRepository warehousesRepository;  
  
 @Mock  
 ShelfsRepository shelfsRepository;  
  
 @Mock  
 LocationsRepository locationsRepository;  
  
 Supplier<List<Object[]>> listOfObjectSupplier = () -> {  
 Object[] objectslist = new Object[3];  
 objectslist[0] = "some value 0";  
 objectslist[1] = "some value 1";  
 objectslist[2] = "some value 2";  
 return Collections.*singletonList*(objectslist);  
  
 };  
 Supplier<List<ArchivoResponse>> archivoResponses = () -> Collections.*singletonList*(new ArchivoResponse("0000001", "desc", "2", "2", "address", 2, 2,"1"));  
 Supplier<ArchivoRequest> archivoRequestSupplier = () -> new ArchivoRequest(1, "desc", "address", 2, 2, 10, 10, 0);  
 Supplier<List<Integer>> listOfIntegers=()-> Arrays.*asList*(1,2,3,4,5);  
 Supplier<Optional<Warehouses>> warehousessSupplier=()->Optional.*of*( new Warehouses(1,"desc",new WTypes(1,"desc"),"address",new Materials(1,"desc",Boolean.*TRUE*), (short) 10,10,"0"));  
 Supplier<List<ArchivoProjection>> listSupplier = () -> Collections.*singletonList*(new ArchivoProjection(1,"desc","ttt","mmm","addr",(short)0,0,"1"));  
 Supplier<List<Shelfs>> listSupplierShelfs = () -> Collections.*singletonList*(new Shelfs(new EmbeddedShelfsId(1,new Warehouses(1,"desc",new WTypes(1,"desc"),"address",new Materials(1,"desc",Boolean.*TRUE*), (short) 10,10,"0")),1,1,1));  
 //Supplier<List<Locations>> listSupplierLocations = () -> Collections.singletonList(new Locations())  
  
 @Test  
 public void testGetArchivosDesc() {  
  
 Mockito.*when*(archivoDao.searchArchivoList(Mockito.*anyString*(),Mockito.*anyString*())).thenReturn(listSupplier.get());  
 Mockito.*when*(archivoMapper.mapArchivoDaoToArchivoResponse(Mockito.*anyList*())).thenReturn(archivoResponses.get());  
 List<ArchivoResponse> response = archivoService.getArchivoList(StringUtils.*EMPTY*, "desc");  
 Mockito.*verify*(archivoDao, Mockito.*atLeastOnce*()).searchArchivoList(Mockito.*anyString*(),Mockito.*anyString*());  
 Mockito.*verify*(archivoMapper, Mockito.*atLeastOnce*()).mapArchivoDaoToArchivoResponse(Mockito.*anyList*());  
 Assertions.*assertNotNull*(response);  
 Assertions.*assertTrue*(response.size() > 0);  
 }  
  
 @Test  
 public void testGetArchivosNumber() {  
  
 Mockito.*when*(archivoDao.searchArchivoList(Mockito.*anyString*(),Mockito.*anyString*())).thenReturn(listSupplier.get());  
 Mockito.*when*(archivoMapper.mapArchivoDaoToArchivoResponse(Mockito.*anyList*())).thenReturn(archivoResponses.get());  
 List<ArchivoResponse> response = archivoService.getArchivoList("1",StringUtils.*EMPTY*);  
 Mockito.*verify*(archivoDao, Mockito.*atLeastOnce*()).searchArchivoList(Mockito.*anyString*(),Mockito.*anyString*());  
 Mockito.*verify*(archivoMapper, Mockito.*atLeastOnce*()).mapArchivoDaoToArchivoResponse(Mockito.*anyList*());  
 Assertions.*assertNotNull*(response);  
 Assertions.*assertTrue*(response.size() > 0);  
 }  
  
 @Test  
 public void testGetArchivosDef() {  
  
 Mockito.*when*(archivoDao.searchArchivoList(Mockito.*anyString*(),Mockito.*anyString*())).thenReturn(listSupplier.get());  
 Mockito.*when*(archivoMapper.mapArchivoDaoToArchivoResponse(Mockito.*anyList*())).thenReturn(archivoResponses.get());  
 List<ArchivoResponse> response = archivoService.getArchivoList(StringUtils.*EMPTY*, StringUtils.*EMPTY*);  
 Mockito.*verify*(archivoDao, Mockito.*atLeastOnce*()).searchArchivoList(Mockito.*anyString*(),Mockito.*anyString*());  
 Mockito.*verify*(archivoMapper, Mockito.*atLeastOnce*()).mapArchivoDaoToArchivoResponse(Mockito.*anyList*());  
 Assertions.*assertNotNull*(response);  
 Assertions.*assertTrue*(response.size() > 0);  
 }  
  
 @Test(expected = ArchivoException.class)  
 public void testgetArchivoDetailsNegative(){  
 Mockito.*doThrow*(new ArchivoException(ArchivoException.Code.*DATA\_NOT\_FOUND*)).when(archivoDao).searchArchivoList(Mockito.*anyString*(),Mockito.*anyString*());  
 List<ArchivoResponse> response = archivoService.getArchivoList(StringUtils.*EMPTY*,"desc");  
 Assertions.*assertNotNull*(response);  
 }  
  
  
 @Test  
 @Ignore  
 public void testDeleteArchivos(){  
  
 Mockito.*when*(shelfsRepository.getShelfsCount(Mockito.*anyList*())).thenReturn(listSupplierShelfs.get());  
 // Mockito.when(locationsRepository.getLocationsCount(Mockito.anyList())).thenReturn(listSupplierShelfs.get());  
 Mockito.*doNothing*().when(warehousesRepository).deleteAllById(Mockito.*anyList*());  
 archivoService.deleteArchivoList(Arrays.*asList*("1"));  
 Mockito.*verify*(warehousesRepository,Mockito.*atLeastOnce*()).deleteAllById(Mockito.*anyList*());  
 }  
  
 @Test(expected = ArchivoException.class)  
 public void testDeleteArchivoError(){  
 Mockito.*doThrow*(new ArchivoException(ArchivoException.Code.*DATA\_CANNOT\_BE\_DELETED\_FROM\_ARCHIVO*)).when(shelfsRepository).getShelfsCount(Mockito.*anyList*());  
 archivoService.deleteArchivoList(Arrays.*asList*("1"));  
 }  
  
 @Test  
 public void testInsertArchivos(){  
 Mockito.*when*(warehousesRepository.findById(Mockito.*any*())).thenReturn(warehousessSupplier.get());  
 Mockito.*when*(archivoMapper.mapRequestToWarehouse(Mockito.*any*())).thenReturn(warehousessSupplier.get().get());  
 Mockito.*when*(warehousesRepository.save(Mockito.*any*())).thenReturn(warehousessSupplier.get().get());  
 archivoService.saveorUpdateArchivosData(archivoRequestSupplier.get());  
 Mockito.*verify*(archivoMapper,Mockito.*atLeastOnce*()).mapRequestToWarehouse(Mockito.*any*());  
 Mockito.*verify*(warehousesRepository,Mockito.*atLeastOnce*()).save(Mockito.*any*());  
 }  
  
 @Test(expected = ArchivoException.class)  
 public void testUpdateArchivoDataNotFoundError(){  
 Mockito.*doThrow*(new ArchivoException(ArchivoException.Code.*DATA\_NOT\_FOUND*)).when(warehousesRepository).findById(Mockito.*anyInt*());  
 archivoService.saveorUpdateArchivosData(archivoRequestSupplier.get());  
 }  
  
}

package mex.walmart.archivo.central.controllers;  
  
import mex.walmart.archivo.central.controllers.administration.ArchivoController;  
import mex.walmart.archivo.central.controllers.administration.catalogs.DocumentosController;  
import mex.walmart.archivo.central.exception.ArchivoException;  
import mex.walmart.archivo.central.model.request.ArchivoRequest;  
import mex.walmart.archivo.central.model.request.ColumnsDetailRequest;  
import mex.walmart.archivo.central.model.request.DocumentosDetailRequest;  
import mex.walmart.archivo.central.model.response.\*;  
import mex.walmart.archivo.central.model.response.Error;  
import mex.walmart.archivo.central.service.administration.ArchivoService;  
import mex.walmart.archivo.central.service.administration.catalogs.DocumentosService;  
import mex.walmart.archivo.central.service.common.CommonService;  
import org.apache.commons.lang3.arch.Processor;  
import org.junit.Test;  
import org.junit.jupiter.api.Assertions;  
import org.junit.runner.RunWith;  
import org.mockito.InjectMocks;  
import org.mockito.Mock;  
import org.mockito.Mockito;  
import org.springframework.test.context.junit4.SpringRunner;  
  
import java.time.LocalDate;  
import java.util.Collections;  
import java.util.List;  
import java.util.function.Supplier;  
  
@RunWith(SpringRunner.class)  
public class ArchivoControllerTest {  
  
 @InjectMocks  
 ArchivoController archivoController;  
  
 @Mock  
 CommonService commonService;  
  
 @Mock  
 ArchivoService archivoService;  
  
 Supplier<List<ArchivoTypeResponse>> archivoTypes = () -> Collections.*singletonList*(new ArchivoTypeResponse(1, "desc"));  
 Supplier<List<ArchivoResponse>> archivoResponses = () -> Collections.*singletonList*(new ArchivoResponse("0000001", "desc", "2", "2", "address", 2, 2,"1"));  
 Supplier<ArchivoRequest> archivoRequestSupplier = () -> new ArchivoRequest(1, "desc", "address", 2, 2, 10, 10, 0);  
  
 @Test  
 public void testGetArchivoTypes() {  
 Mockito.*when*(commonService.getAllArchivoTypes()).thenReturn(archivoTypes.get());  
 var response = archivoController.getArchivoTypes();  
 Assertions.*assertNotNull*(response);  
 Assertions.*assertTrue*(response.isSuccess());  
 }  
  
  
 @Test  
 public void testGetSearchList() {  
 Mockito.*when*(archivoService.getArchivoList(Mockito.*anyString*(), Mockito.*anyString*())).thenReturn(archivoResponses.get());  
 var response = archivoController.getArchivoList(Mockito.*anyString*(), Mockito.*anyString*());  
 Assertions.*assertNotNull*(response);  
 Assertions.*assertTrue*(response.isSuccess());  
 }  
  
 @Test  
 public void testDeleteArchivos() {  
 Mockito.*when*(archivoService.deleteArchivoList(Mockito.*anyList*())).thenReturn(Collections.*singletonList*(1));  
 var response = archivoController.deleteArchivoList(Mockito.*anyList*());  
 Mockito.*verify*(archivoService, Mockito.*atLeastOnce*()).deleteArchivoList(Mockito.*anyList*());  
 Assertions.*assertNotNull*(response);  
 Assertions.*assertFalse*(response.isSuccess());  
 }  
  
 @Test  
 public void testPostArchivos() {  
 Mockito.*doNothing*().when(archivoService).saveorUpdateArchivosData(Mockito.*any*());  
 var response = archivoController.newArchivoData(archivoRequestSupplier.get());  
 Mockito.*verify*(archivoService, Mockito.*atLeastOnce*()).saveorUpdateArchivosData(Mockito.*any*());  
 Assertions.*assertNotNull*(response);  
 Assertions.*assertTrue*(response.isSuccess());  
 }  
}