THOMAS MORE

Exporteren van Grids naar Excel

Realisatiedocument

Sebastiaan Daniëls
Student Bachelor in de Toegepaste Informatica – Applicatieontwikkeling

2024 - 2025

Voorwoord

In mijn laatste jaar van de bachelor Toegepaste informatica - Specialisatie Application Development - heb ik de kans gekregen om dertien weken stage te lopen bij Pension Architects in Zoersel. Dit realisatiedocument is een verslag van deze stage.

Allereerst zou ik graag alle medewerkers van Pension Architects willen bedanken. Zij hebben mij vanaf de eerste dag met open armen ontvangen en mij met plezier begeleid doorheen de stage. Ik kon steeds bij iedereen terecht met praktische vragen, en ik kreeg enorm veel hulp en begrip wanneer ik vastzat.

In het bijzonder zou ik graag ook nog Jenthe Mariën en Kevin Aerts willen bedanken. Zij waren mijn stagementoren tijdens deze stage. En hebben mij geholpen om dit project tot een goed einde te brengen. Zij waren steeds beschikbaar voor mijn vragen of voor een aangenaam gesprek.

Daarnaast zou ik ook mijnheer Patrick Dielens willen bedanken als stagebegeleider. Hij was steeds beschikbaar voor vragen te beantwoorden of voor algemene praktische hulp tijdens de stage.

SAMENVATTING

Dit realisatiedocument is een zelfgeschreven[1] verslag van mijn stage bij Pension Architects in Zoersel. Dit is een bedrijf dat zich specialiseert in het beheren van aanvullende pensioenen. Ze werken nauw samen met grote bedrijven en overheidsinstanties om het makkelijk te maken om de aanvullende pensioenen van hun werknemers te beheren, uit te betalen, etc...

Hiervoor hebben de klanten toegang tot een *admin paneel* waarop ze deze zaken kunnen bekijken en beheren. Het beheren van de pensioenplannen van meer dan een miljoen werknemers vergt enorm veel data. Deze data wordt op de webtoepassing vaak getoond in een grid-formaat. (Grote tabellen waar data, knoppen, en enorm veel andere dingen in kunnen staan).

Mijn stageopdracht was om de mogelijkheid toe te voegen om deze grote tabellen (grids) asynchroon te kunnen exporteren naar een Excel bestand, zodat de klanten hier verdere berekeningen op kunnen doen met hun eigen interne tools. Hiervoor moest ik rekening houden met data-integriteit, security, alsook efficiëntie. (Alle persoonsgegevens op afbeeldingen zijn test gegevens en zijn niet gelinkt aan een echt persoon)

[1] Met zelfgeschreven wordt bedoeld dat ik doorheen heel dit document geen enkele vorm van Generatieve (of andere) vormen van Artificiële Intelligentie heb gebruikt. Noch voor het schrijven van teksten, noch voor het bijwerken van teksten.

LIJST MET GEBRUIKTE ILLUSTRATIES

AFKORTINGENLIJST

Afkorting	Uitleg
adminpanel	De webtoepassing van Pension Architects waarop klanten alles kunnen beheren
PA	Pension Architects
job-scheduler	Een service wiens taak het is om asynchroon taken uit te voeren die te zwaar zijn voor andere services
Sprint	Een twee-wekelijkse tijdspanne waaring een vooraf bepaalde set van taken hoort afgewerkt te zijn.
IDE	Integrated Development Environment

Inhoudsopgave

1. Inleiding

In dit eindwerk vindt u informatie over mijn uitvoering van de stage rond het exporteren en overzetten van Grids vanuit een front-end toepassing naar een asynchrone back-end toepassing.
Deze nieuwe implementatie zorgt ervoor dat klanten van Pension Architects gemakkelijk en snel grote grids (tabellen op de website) naar een excel-bestand kunnen exporteren zonder dat de webtoepassing daardoor belast wordt.

Aangezien de vorige implementatie voor het exporteren van grids tijdelijk is uitgeschakeld door de grote problemen met de huidige werking. Is het de bedoeling dat Pension Architects op het einde van mijn stage mijn nieuwe implementatie ook effectief onmiddelijk kan implementeren. Zodat klanten deze functie binnenin de webtoepassing zo snel mogelijk weer kunnen gebruiken.

Dit eindwerk is opgedeeld in vier hoofdstukken;
In het eerste hoofdstuk krijgt u meer informatie over Pension Architects. Het bedrijf waarvoor ik de stage heb gemaakt. In het tweede hoodstuk vindt u de analyse die ik heb gemaakt vooraleer ik ben begonnen met de uitwerking van de stage, die u in het derde hoofdstuk vindt. Ten slotte is er ook nog een hoofdstuk over de beveiliging van de applicatie.

2. Pension Architects

Ik heb mijn stageopdracht uitgevoerd bij Pension Architects in Zoersel. Dit bedrijf is een redelijk kleinschalig bedrijf met ongeveer 20 werknemers en is in 2009 opgericht door meneer Ivan Eulaers.

Het beheren van aanvullende pensioenen is voor veel bedrijven een moeilijke zaak. Het vergt enorm veel kennis en mankracht om als bedrijf een volledig interne administratie te houden van alle aanvullende pensioenen van de werknemers. Hier biedt Pension Architects een oplossing.

Ze hebben ondertussen al meer dan 15 jaar ervaring met alle aspecten van de steeds wijzigende pensioen- en benefitsmarkt. Ze hebben expertise opgebouwd in zowel de actuariële, fiscale, juridische en boekhoudkundige aspecten die gerelateerd zijn aan pensioenplannen. Ze helpen ook bij de optimalisatie en consolidatie van uw aanvullende pensioensystemen.

Pension Architects kan dus het volledige beheer van de aanvullende pensioenen overnemen van haar klanten. Dit wil bijvoorbeeld zeggen dat Pension Architects instaat voor het uitbetalen van de pensioenen aan de werknemers van de klant. Ook kunnen ze helpen bij het opzetten of hertekenen van de pensioenplannen. Verder bezitten ze ook meerdere webapplicaties (Waaronder ook adminpanel, de webapplicatie waarin ik mijn stage doe). Deze aplicaties zijn intern door de developpers ontwikkeld om het bewer van pensioenplannen nog verder te vereenvoudigen.

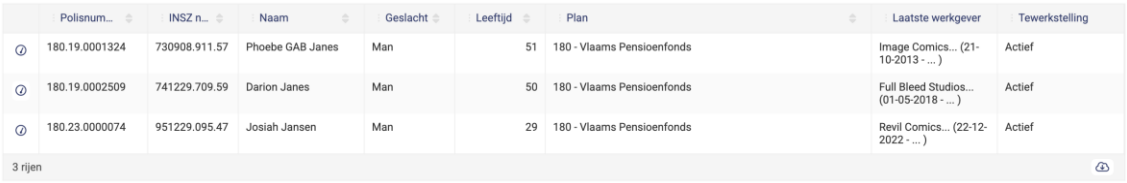
De oplossingen die zij bieden zijn ook begrijpbaar voor niet-experten, zoals de werknemers die uitbetaald worden of andere stakeholders. Pension Architects richt zich op pensionfondsen, bedrijven, sectoren, en dienstverleners.

3. Analyse

De opdracht was vanuit het bedrijf vrij abstract geformuleerd. De aanleiding van deze stageopdracht was zeer duidelijk. En ik wist snel waarom de vorige implementatie voor het exporteren van de grids niet goed genoeg was. Maar ik ben volledig vrijgelaten in het uitwerken van een oplossing voor dit probleem. Ik heb me daarom de eerste weken beziggehouden met het opstellen van een goede plan van aanpak. Die ik dan samen met mijn stagementoren heb overlopen en aangepast, totdat we allemaal akkoord waren met welke richting we wouden uitgaan voor het uitvoeren van deze stage.

3.1. Aanleiding en achtergrond van de stageopdracht

Adminpanel is de interne naam voor de web toepassing die klanten kunnen gebruiken voor het beheren van aanvullende pensioenen. Overal en op bijna elke pagina is er wel een tabel te vinden met data (een grid). Dit is heel handig voor ons, aangezien we in die grids niet enkel tekst, maar ook knoppen, tekens, of zelfs kleuren kunnen zetten. Naargelang van de informatie die we naar de klant willen doorgeven. Echter is dit voor de klant niet altijd handig, want zij hebben deze data vaak nodig voor andere, interne doeleinden. Hiervoor geven we ze de optie om elk grid te downloaden naar een Excel bestand.

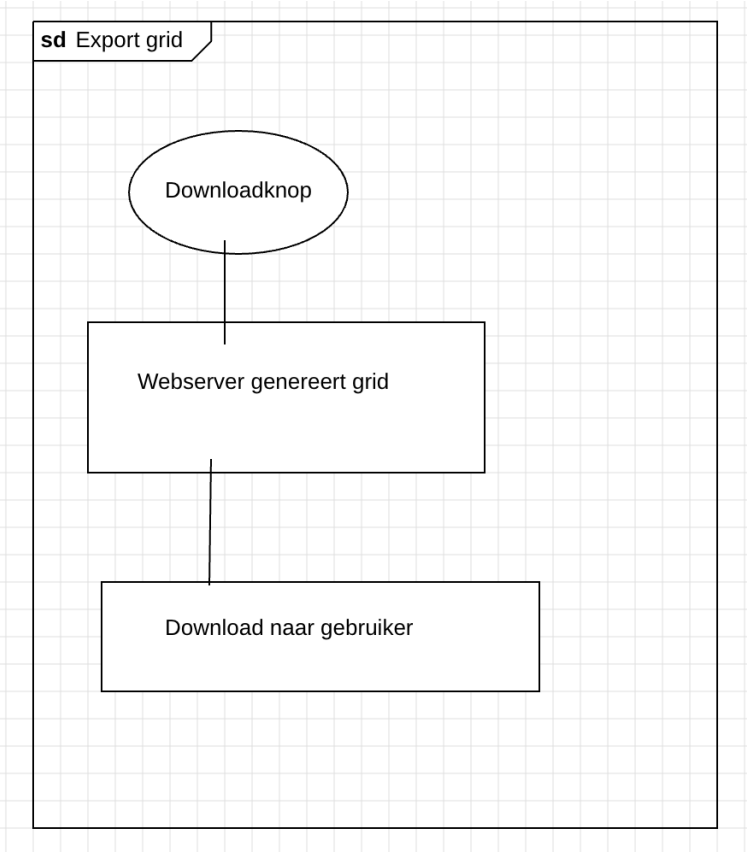


	Polisnum...	INSZ n...	Naam	Geslacht	Leeftijd	Plan	Laatste werkgever	Tewerkstelling
	180.19.0001324	730908.911.57	Phoebe GAB Janes	Man	51	180 - Vlaams Pensioenfonds	Image Comics... (21-10-2013 - ...)	Actief
	180.19.0002509	741229.709.59	Darion Janes	Man	50	180 - Vlaams Pensioenfonds	Full Bleed Studios... (01-05-2018 - ...)	Actief
	180.23.0000074	951229.095.47	Josiah Jansen	Man	29	180 - Vlaams Pensioenfonds	Revil Comics... (22-12-2022 - ...)	Actief

3 rijen

Figuur 3.1.1: oude tabel met downloadknop rechtsonder

Echter draait de web toepassing op een heel lichte server die losstaat van alle andere services (zoals de database of de job-scheduler). De implementatie voor het exporteren van de grids was hierin gemaakt. Het probleem is dat vanaf een bepaalde grootte van het grid, de web toepassing overbelast werd en volledig crashte. Dit had als resultaat dat helemaal niemand meer op de web toepassing kon totdat deze herstart werd.



Figuur 3.1.2: flowchart oude implementatie

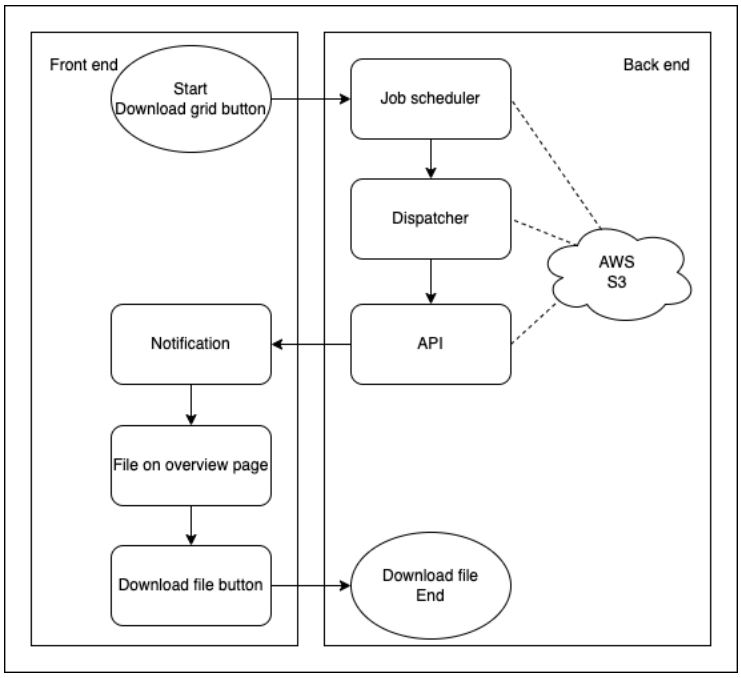
Dit is uiteraard een worst-case scenario. Omdat één enkele eindgebruiker de web toepassing voor iedereen kan neerhalen.

3.2. Verwacht resultaat

Om te voorkomen dat klanten de toepassing neerhalen voor anderen, moest ik een nieuwe implementatie verzinnen zodat klanten de grids juist en snel konden exporteren, zonder dat dit enige hinder veroorzaakte.

In het nieuwe systeem zal een nieuw component genaamd *export-grids* de effectieve export als backend overnemen. Er heeft al een vorige stagiair een deeltje van de nieuwe werking gemaakt.

PA werkt voor grote backend transacties met een *job-scheduler*. Deze zal jobs asynchroon van de frontend uitvoeren. Het *adminpanel* kan een job aanmaken dat een programma opstart (dit zal in ons geval het programma *export-grids* zijn) met bepaalde parameters. (Bv het id van de grid, username, en filters).



Figuur 3.2.1: flowchart nieuwe implementatie

Wanneer de job wordt opgestart zal export grids een grid moeten maken op basis van de argumenten die gegeven zijn.
Wanneer de Excel is gemaakt, zal een *dispatcher* het bestand opslaan op de juiste plaats en de eindgebruiker op de hoogte stellen dat zijn/haar download beschikbaar is d.m.v. een notificatie.
De implementatie van het schedulen van een job, en het dispatchen van het resultaat zijn deels gedaan door de vorige stagiair. Het is nu aan mij om de juiste argumenten mee te geven, de grids te bouwen in de applicatie server in plaats van de webserver, en te exporteren.

Dit moet uiteraard zeer goed beveiligd zijn zodat enkel geautoriseerde gebruikers een grid kunnen exporteren (Deze grids kunnen *zéér* gevoelige data bevatten). Daarnaast kunnen verschillende gebruikers op basis van hun permissies andere data zien op hetzelfde grid. (één persoon ziet bv 100 rijen terwijl een andere er maar 10 kan zien). Dit moet ook meegegeven worden zodat een gebruiker die een grid exporteert niet te veel of te weinig data krijgt.

3.3. Business case

Voor Pension Architects heeft dit natuurlijk enorm veel voordelen. De meerderheid van de grids bevat te veel rijen om goed geëxporteerd te kunnen worden. In de meeste gevallen blijft de web toepassing een 20-30-tal seconden hangen vooraleer het grid gedownload wordt. En in de slechtste gevallen crasht de volledige applicatie.

Met mijn nieuwe implementatie kunnen de klanten met een gerust hart de export toepassing gebruiken zodat zij verder kunnen rekenen met de data die PA hen aanlevert.

Ook zullen de downloads op een centrale plek beschikbaar worden gesteld. Dus mocht het downloaden langs de kant van de gebruiker mislukken, of de klant verwijdert het document, kan dit alsnog opnieuw gedownload worden zonder dat er nieuwe berekeningen moeten worden gedaan.

3.4. Planning

Bij het plannen van de uitwerking van de stage heb ik alle grote onderdelen opgedeeld in verschillende sprints. Ik heb het project opgedeeld in zes grote onderdelen en deze toegewezen aan elke sprint.
Elke sprint stemt overeen met een tijdspanne van twee werkweken.

Sprint	Taak
1	Doorgeven van filter naar de job-scheduler
2	Opbouwen van de filter in export-grids
3	Grids overzetten naar service-layer
4	Grids opbouwen in export-grids
5	Grids exporteren
6	Grids Dispatchen naar S3 server + Downloadpagina voor de gebruiker

3.5. Primaire doelgroep en andere stakeholders

De primaire doelgroep voor dit project zijn de klanten van PA die geregeld grids willen exporteren. Op dit moment stoten ze vaak op zeer trage exports, of zelfs crashes. Daarnaast zijn ook de medewerkers van PA ook stakeholders in dit project, aangezien zij nu vaak de klachten van de klanten moeten horen en oplossen.

3.6. Informatie en rapportering

Voor de algemene rapportering van de stage aan de stagebegeleider werken we met een wekelijkse update. Ik schrijf elke dag een korte paragraaf met daarin informatie omtrent wat ik die dag gedaan heb, welke problemen ik heb ondervonden, en allerlei andere dagelijkse zaken. Ook heb ik een takenlijst waarin ik opschrijf wat ik heb gedaan en hoe ver ik daar mee sta. Deze documenten stuur ik wekelijks naar mijn stagebegeleider en stagementor. Zij kijken deze dan na en geven feedback indien nodig.

Binnen het bedrijf zelf werken we met twee-wekelijkse sprints. Op het begin van elke sprint wordt een planning opgemaakt met daarin zeer concrete afgebakende taken (tickets). Deze tickets worden dat opgenomen door de developers doorheen de sprint. Op het einde van de sprint bekijken we samen wat er wel/niet is afgewerkt en passen we de planning aan op basis van de prioriteiten.

Ten slotte heb ik wekelijks ook kort informeel even samengezeten met de stagementoren om kort te bespreken wat ik die week had gedaan en wat de planning was voor de volgende week.

3.7. Gebruikte Tools en Software

Pension Architects gebruikt veel verschillende tools en software voor de ontwikkeling en beheer van hun applicaties. Er wordt vooral gewerkt met en rond de programmeertaal Java. Daarrond gebruiken we nog enkele frameworks die gebaseerd zijn op Java, alsook databeheer tools die samenwerken met de toepassingen.

3.7.1. Java

Java is een programmeertaal en computerplatform dat voor het eerst werd uitgebracht door Sun Microsystems in 1995. Binnen PA wordt deze programmeertaal voor alle toepassingen gebruikt. Voor zowel de front-end (zie Tapestry) alsook de back-end.

3.7.2. IntelliJ

IntelliJ is de software waarin ik Java code ontwikkel. Het is een geïntegreerde ontwikkelomgeving (IDE) voor professionele ontwikkeling in Java en Kotlin. Het is ontworpen om de productiviteit van ontwikkelaars te maximaliseren en heeft een sterke focus op privacy en beveiliging.

Verder doet het ook statische codeanalyse en bied het verschillende vormen van automatische code aanvulling aan. Ik heb alle code die ik tijdens mijn stage ontwikkeld heb, in deze IDE geschreven.

3.7.3. Apache Tapestry

Tapestry is een Java framework dat het enorm makkelijk maakt om webapplicaties te ontwikkelen met Java. Het is gebaseerd op het MVC (Model - View - Controller) concept. Waarbij elk component verantwoordelijk is voor een deel van de applicatie.

Ik ben zelf met alle aspecten van Tapestry in contact gekomen. Aangezien ik zowel in de View (front-end), alsook de Model en Controller (Back-end) heb gewerkt.

3.7.4. MySQL

MySQL is een relationele database die het mogelijk maakt om persistent data op te slagen en deze razendsnel aan te roepen.
Binnen Pension Architects wordt alle data beheert via een MySQL server.

3.7.5. Liquibase

Liquibase is een tool voor het beheren van database structuren. Het is een soort centraal beheer van alle aanpassingen die aan een database gebeuren. Zodat deze altijd op de zelfde manier worden uitgevoerd. Wanneer een developer een aanpassing wil doen aan de database. Schrijft hij hiervoor een stukje code in Liquibase. Deze code wordt dan automatisch uitgevoerd bij alle andere developers, Alsook op de test- en livedatabases in productie. Hierdoor heeft iedereen altijd de juiste en zelfde versie van de databasestructuur.

3.7.6. Jira

Jira is een ticketing tool voor devops teams. Het is een omgeving waaring teams via de Agile manier sprints kunnen uitwerken.
Bij Pension Architects werken we met tweewekelijkse sprints. Alle developers werken dan met tickets uit Jira die ze dan opnemen en uitwerken.

Dit is een belangrijke tool voor ons team aangezien we hiermee een goede en duidelijke planning kunnen maken. We zorgen hiermee ook voor een goede opvolging van de development cyclus.

3.7.7. Confluence

Confluence is een andere tool die we als devops team gebruiken. Het is een soort van "Wikipedia" waarin alle developers documentatie kunnen schrijven omtrent changes, code, of andere belangrijke kennis die we hebben.

4. Uitvoering van de stage

In dit hoofdstuk behandelen we de effectieve uitvoering van de stage. Hier vindt u hoe de export wordt aangemaakt met schermafbeeldingen en code-snippets met bijbehorende uitleg.

Om dit makkelijker te maken om te begrijpen is de volgorde van de onderwerpen is de chronologische volgorde van hoe een Excel wordt gemaakt, niet de volgorde waarin ikzelf heb gewerkt. Dit omdat ik tijdens de uitvoering van de stage niet altijd chronologisch heb gewerkt, en vaak van het ene naar het andere onderwerp ben gesprongen

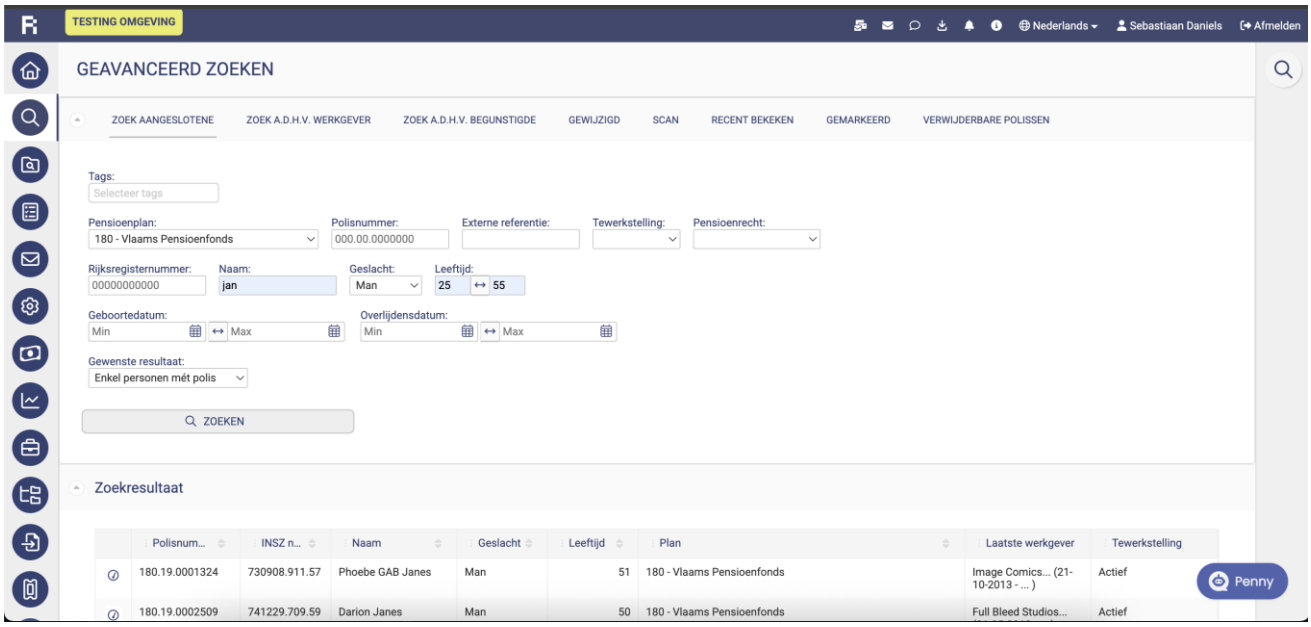
4.1. Doorgeven van filter naar de job-scheduler

Allereerst moeten we de nieuwe downloadknop configureren voor de download. Aangezien elk grid individueel zal moeten worden overgezet vanuit adminpanel naar de service-layer, heb ik ook een implementatie gemaakt die er voor zorgt dat de front-end eerst kijkt óf er een grid geconfigureerd is in de service-layer. Indien dit het geval is, toont hij de nieuwe knop. Anderzijds is dat de oude implementatie.

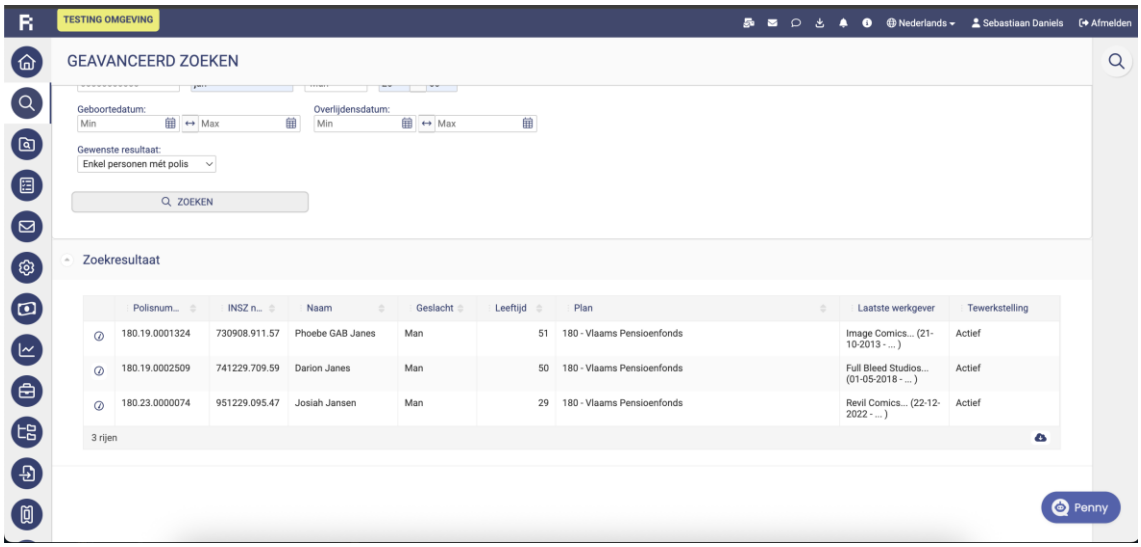
Om intern een onderscheid te kunnen maken van welke implementatie een grid heeft, heb ik besloten om op een hele subtiele manier aan te geven of het grid op de nieuwe, of oude manier zal worden geëxporteerd. Je kan zien of een grid is geconfigureerd wanneer het download icoontje volledig blauw is gevuld. Als het enkel de omtrek is, dan is het nog de oude configuratie.

Figuur 4.1.1: oud en nieuw icoon

Een grid kan allerlei data bevatten. Sommige grids worden automatisch opgebouwd op basis van vaste argumenten, maar de meeste grids worden opgebouwd door een filter dat de gebruiker kan meegeven.



Figuur 4.1.2: Voorbeeld van een filter



Figuur 4.1.3: Grid met filter

Aangezien deze filter door de gebruiker in de front-end is meegegeven. Kunnen we deze niet zomaar gebruiken in de export die in de backend draait. Daarom moeten we de filter en de data van deze filter encoden en meegeven aan de job-scheduler.

Wanneer de gebruiker op de exportknop drukt, roepen we de functie *exportGridAsync()* aan. De *eventContext* is de filter en is een filter die overerft van een superFilter:

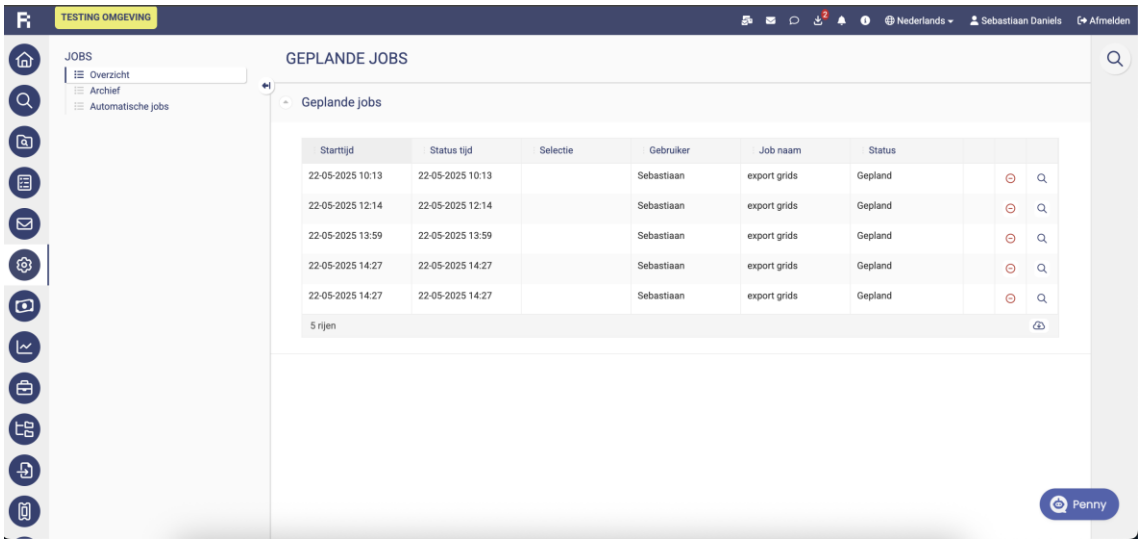
```
1. void exportGridAsync(EventContext eventContext) {
    processEventContext(eventContext, false);

    // Sanity check: This should never error as there is no button to press if not
configured
    GridAsyncExporter asyncExporter = gridAsyncExporterSource.getExporter(id)
          .orElseThrow(() -> new IllegalStateException("No async grid exporter found for
id: " + id));

    asyncExporter.scheduleExport(id, getFilterArguments(eventContext));
    rvlJobConfirmation.show("rvlJobConfirmation");
}

private String getFilterArguments(EventContext eventContext) {
    String[] values = eventContext.toStrings();
    URLEncoderImpl encoder = new URLEncoderImpl();
    return Arrays.stream(values)
          // TODO: check when this is this included
          .filter(value -> !processedClientId.equals(value))
          .map(encoder::encode)
          .collect(Collectors.joining(","));
}
2.
```

De toStrings() methode van de eventContext is een methode die de argumenten van de filter slim om kan zetten naar een uri-safe string:

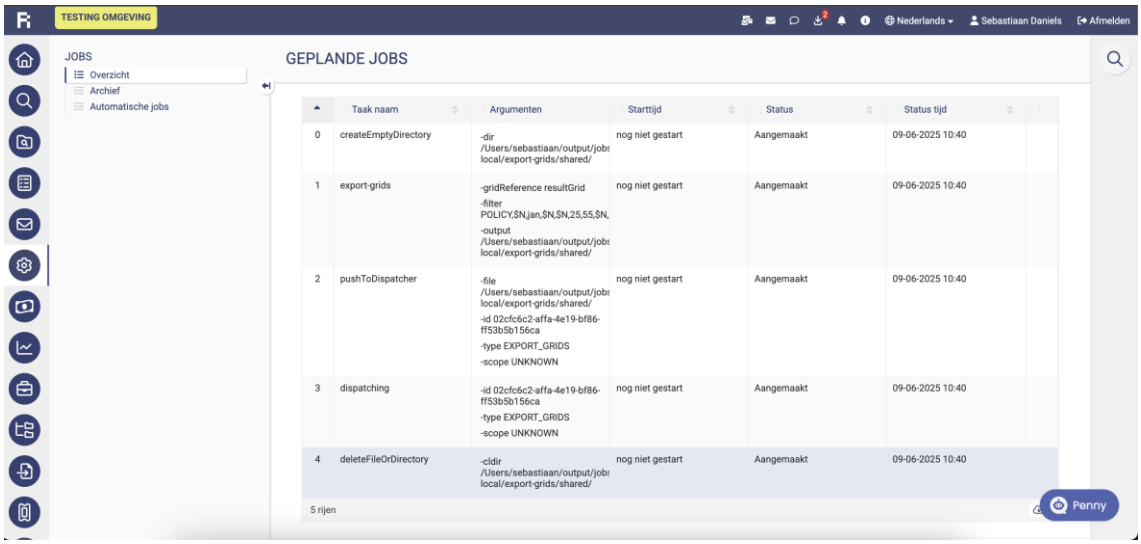
```
1. @Override public String[] toStrings() {
    return new String[] {
          toContext(plan),
          toContext(planned),
          toContext(unplanned),
          toContext(purchase),
          toContext(sale),
          toContext(useInterval),
          toContext(startPeriod),
          toContext(endPeriod),
    };
}
2.
```

Deze string wordt dan samen met de gebruikersnaam alsook de andere argumenten (zoals de naam van het grid) meegegeven aan de job-scheduler. Deze scheduler bestaat al lang en heeft een makkelijke interface waarmee ik kon werken om mijn eigen job met mijn eigen argumenten aan te maken.

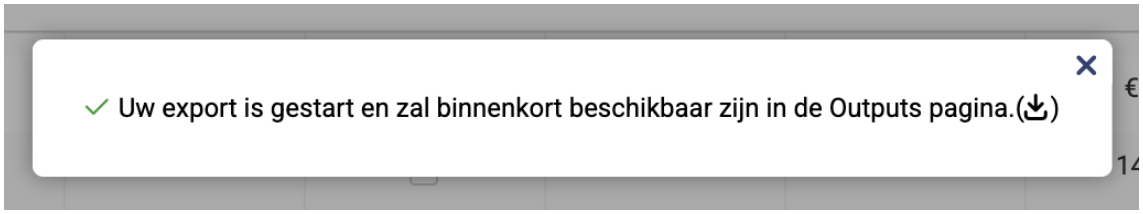
Wanneer dit allemaal in orde is, zien we dat de job correct is aangemaakt en alle argumenten zijn meegegeven. De job-scheduler zal deze taak automatisch opstarten wanneer de queue vrijkomt.



Figuur 4.1.4: Een aantal aangemaakt jobs



Figuur 4.1.5: Taken van een aangemaakte job



Figuur 4.1.6: Melding die de gebruiker krijgt bij het exporteren

4.2. Opbouwen van de filter in export-grids

Wanneer de job-scheduler een plekje vrij heeft in de queue wordt de job opgestart. Het programma export-grids wordt aangeroepen met de argumenten die we hebben meegegeven tijdens het aanmaken van de job.

We moeten nu de argumenten ophalen uit de “args” en deze terug decoden vanuit een string naar een filter. Allereerst haalt export-grids de argumenten uit de job zodat we hier mee kunnen verder werken.

```
1. FILTER("filter", true, false, "the filters used to create the grid");
    private final String argName;
    private final boolean argument;
    private final boolean required;
    private final String description;

    ExportGridsOption(String argName, boolean argument, boolean required, String
description) {
       this.argName = argName;
       this.argument = argument;
       this.required = required;
       this.description = description;
    }

    @Override     public String getArgName() {
       return argName;
    }

    @Override     public boolean hasArgument() {
       return argument;
    }

    @Override     public boolean isRequired() {
       return required;
    }

    @Override     public String getDescription() {
       return description;
    }
}
2.
```

```
1. public class TaskRunnerImpl extends AbstractTaskRunner implements TaskRunner {

    @Override     public Options getOptions() {
       return new OptionBuilder()
             .withOptions(ExportOption.values())
             .withOptions(ExportGridsOption.values())
             .build();
    }

    @Override     public void executeTask(CommandLine commandLine, Registry registry) {

       String gridReference =
commandLine.getOptionValue(ExportGridsOption.GRID_REFERENCE.getArgName());
       Path outputDir =
Path.of(commandLine.getOptionValue(ExportOption.OUTPUT.getArgName()));
       String filter = commandLine.getOptionValue(ExportGridsOption.FILTER.getArgName());
2.
```

We hebben nu de filter terug opgebouwd in export-grids. Nu kunnen we verder met het opbouwen van het Grid in de back-end.

4.3. Grids overzetten naar service-layer

Nu we de filter en weten welk grid we moeten exporteren, moeten we ook aan de effectieve data kunnen. Echter is dit momenteel een probleem. Alle grids zijn geconfigureerd in adminpanel. Dit wil zeggen dat ze volledig worden aangemaakt op de webserver. Dit betekent ook dat de Jooq queries in de front-end staan. We weten dus niet welke data we uit de database moeten halen. Daarom moet elk grid worden overgezet naar een nieuwe configuratie in de service-layer.

De oude implementatie gebeurt in 3 projecten: *adminpanel, admindb-service-layer en components*.

Wanneer een grid moet getoond worden zal adminpanel eerst uit components een "gridModel" maken, dit is een component. Dit component zal de service layer aanspreken om de effectieve data eruit te halen en terug te sturen naar adminpanel waar het grid getoond wordt. Wanneer de user op de downloadknop drukt. Zal adminpanel de excel file genereren en downloaden naar het toestel van de gebruiker.

In de nieuwe implementatie is de configuratie iets complexer aangezien we een nieuw project hebben voor het exporteren van de grids: export-grids. Dit project draait op de applicatieserver, en staat volledig los van de webserver. Ook is de volgorde en locatie van bepaalde zaken aangepast.

Hoe een grid zou moeten opgebouwd worden werd eerst beschreven in adminpanel. In verschillende files wordt er telkens bepaald hoe de filter eruitziet, waar onze database query's te vinden zijn. En welke attributen en kolommen het grid moet tonen. Aangezien export-grids op een andere server draait, kunnen we niet aan deze files. Daarom zal de beschrijving van de grids moeten verhuizen naar de service layer.

export-grids moet dus op basis van deze gegevens het juiste grid kunnen opbouwen. Dit kunnen we doen door een algemene configuratie module te maken in de service layer.

De algemene configuratie van de grids zit in een bestand genaamd "ServiceLayerGridConfigurationModule". Een voorbeeld van een geconfigureerd grid ziet er zo uit:

```
 1. @Contribute(GridConfigSource.class)
 2. public static void contributePolicyGridConfigs(
 3.         MappedConfiguration<String, GridModelConfig> gridConfigs,
 4.         PolicySearchFrontService policySearchService) {
 5.     gridConfigs.add("resultGridPerson", GridModelConfigBuilder.createPagedConfig(
 6.                                     PersonDto.class,
 7.                                     filter -> personService.find(new
PersonFilter(filter)))
 8.                                     .addProperty(INSZ, "inszNumber")
 9.                                     .addProperty(DETAIL, ID)
10.                                     .includeProperties(ID, INSZ, "fullname", "gender",
"age", DETAIL)
11.                                     .build());
12. }
13.
```

Dit is de configuratie voor het “Search” grid op de zoek-pagina. De “GridConfigSource” is een mapping tussen een string (Het ID van elk grid), en de “GridModelConfig”.

Hiermee kunnen we de configuratie van een grid ophalen op basis van het ID van het grid. Belangrijk om te weten is dat nieuwe grids dus **uniek** moeten zijn van naam. Ook als ze op andere pagina’s staan. (Tenzij dit uiteraard exact hetzelfde grid is dat ook dezelfde data moet tonen.)

De gridModelConfig is de effectieve configuratie van het grid. Dit bepaald uit welke source we de data halen en met welke filter(s) we dit doen.

In “gridConfigs.add” geven we dus een string mee (In dit geval “resultGrid”), en een gridModelConfigBuilder, hierin definieren we eerst welke klasse/ Dto we gaan teruggeven. Daarna geven we een lambda functie mee om te definieren waar we de data gaan ophalen en welke filter we gaan aanroepen om de data juist te filteren. In ons geval is dat de policySearchService.

Uiteindelijk kunnen we ook nog properties toevoegen en excluden. Hiermee weet zowel *adminpanel,* alsook *export-grids* welke colomnamen en properties getoond moeten worden.

```
@Contribute(GridConfigSource.class)  no usages  Sebastiaan Daniels +1
public static void contributePendingPaymentsGridConfigs(
        MappedConfiguration<String, GridModelConfig> gridConfigs,
        PaymentOrderFrontService paymentOrderService) {

    gridConfigs.add(
            k: "grdIncomingPaymentsSummary", GridModelConfigBuilder.createCollectionConfig(
                    IncomingPaymentsSummary.class,
                    String[] filter -> paymentOrderService.createIncomingPaymentsSummary(
                            new IncomingPaymentsFilter(filter)))
                    .build());
```

Figuur 4.3.1: Voorbeeld van een geconfigureerd grid

4.4. Grids opbouwen in export-grids

Eenmaal dat we de configuratie hebben aangemaakt in de service-layer, kunnen we het grid opbouwen in export-grids. Aan de hand van het gridID kunnen we in de config mapping de juiste configuratie ophalen zodat we weten hoe we het grid moeten opbouwen

Eenmaal we de gridconfiguratie hebben kunnen we een beanModel maken. Dit is in principe gewoon een abstracte superklasse die we gebruiken zodat elk mogelijk grid geexporteerd kan worden. Dit ziet er als volgt uit:

```
1. private BeanModel<?> createBeanModel(GridModelConfig gridConfig) {
    // Fixme: what are these messages used for?
    BeanModel<?> beanModel = beanModelSource.createDisplayModel(
          gridConfig.getBeanType(), new MapMessages(Locales.BELGIUM_NL, Map.of()));

    for (Map.Entry<String, String> mappedColumn : gridConfig.getProperties().entrySet()) {
       beanModel.add(
             mappedColumn.getKey(),
             propertyConduitSource.create(beanModel.getBeanType(),
mappedColumn.getValue()));
    }

    for (Map.Entry<String, PropertyConduit> mappedColumn :
gridConfig.getPropertyConduits().entrySet()) {
       beanModel.add(mappedColumn.getKey(), mappedColumn.getValue());
    }
    beanModel.include(gridConfig.getIncludedProperties());
    beanModel.exclude(gridConfig.getExcludedPropertiesForExport());
    return beanModel;
}

private Object getCellValue(BeanModel<?> beanModel, String propertyName, Object rowValue)
{
    PropertyModel props = beanModel.get(propertyName);
    try {
       return props.getConduit().get(rowValue);
    } catch (Exception e) {
       return e.getLocalizedMessage();
    }
}
2.
```

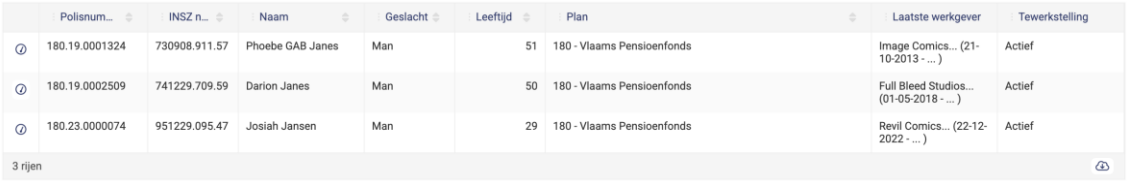
Dit geeft ons een object van het type BeanModel<?> terug. Dit bevat de juiste en correcte data die we nodig hebben om de data om te vormen naar een Excel bestand.

4.5. Grids exporteren

Voor het exporteren van een grid en het omzetten van een BeanModel naar een Excel file, gebruiken we een module genaamd POI.

POI is een java module onderhouden door de Apache Foundation. Het wordt gebruikt om met officebestanden te werken vanuit Java, waaronder ook Excel.

Wanneer het gridModel (BeanModel) is opgebouwd vanuit de gegevens die we hebben verkregen vanuit de job-scheduler, kunnen we het ook effectief beginnen te exporteren.



	Polisnum...	INSZ n...	Naam	Geslacht	Leeftijd	Plan	Laatste werkgever	Tewerkstelling
	180.19.0001324	730908.911.57	Phoebe GAB Janes	Man	51	180 - Vlaams Pensioenfonds	Image Comics... (21-10-2013 - ...)	Actief
	180.19.0002509	741229.709.59	Darion Janes	Man	50	180 - Vlaams Pensioenfonds	Full Bleed Studios... (01-05-2018 - ...)	Actief
	180.23.0000074	951229.095.47	Josiah Jansen	Man	29	180 - Vlaams Pensioenfonds	Revil Comics... (22-12-2022 - ...)	Actief

3 rijen

Figuur 4.5.1: Voorbeeldgrid om te exporteren

De code om een grid te exporteren is als volgt:

```
1. @Override public void export(String gridReference, Path outputDir, String filter) {
    UserDetails previousUser =
(UserDetails)SecurityContextHolder.getContext().getAuthentication().getPrincipal();
    UserDetails currentUser = userDetailsService.loadUserByUsername(getUser());

    try {
       setUser(currentUser);
       GridModelConfig gridConfig = getGridConfig(gridReference);
       PageIterator<?> iter = gridConfig.getPageIterator(getDecodedFilterParts(filter));

       ProgressLog progressLog = progressLogFactory.create("export.grids", "records", 0,
false);
       progressLog.info("Started exporting of grid: " + gridReference);

       File file = new File(outputDir.toFile(), buildFileName(gridReference));

       gridBuilder.startMerge(progressLog, file.toPath(), gridConfig);

       gridBuilder.merge(iter, gridConfig);

       gridBuilder.endMerge(file.toPath());

       createDispatchMessage(file.toPath(), getCurrentTaskId(),
currentUser.getUsername());

       progressLog.info("Grid export successful for: " + gridReference);
       progressLog.stop();
       sessionManager.commit();
    } catch (Exception e) {
       throw new ProcessingException("Error exporting grid for: ", gridReference + ".
Reason: ", e);
    } finally {
       setUser(previousUser);
```

Als eerste is er een deel van de security. Hier kom ik later op terug. Allereerst maken we een File object aan. Dit is het bestand waarin we de data gaan

wegschrijven. Daarnaast zijn er ook drie belangrijke functies die we gebruiken om de file op te bouwen, namelijk;

- *startMerge()*
- *merge()*
- *endMerge()*

Deze functies gaan we iets verder op in.

4.5.1. Start Merge

De startMerge methode is de methode waarin we de echte File aanmaken.

```
1. @Override public void startMerge(ProgressLog log, Path templateFile, GridModelConfig
gridConfig) {
    progressLog = log;
    workbook = new XSSFWorkbook();
    sheet = workbook.createSheet("Grid Data");
    currentRow = 0;

    Row row = sheet.createRow(currentRow);

    BeanModel<?> beanModel = createBeanModel(gridConfig);
    ArrayList<String> columns = new ArrayList<>(beanModel.getPropertyNames());
    for (int i = 0; i < columns.size(); i++) {
       Cell cell = row.createCell(i);
       cell.setCellValue(columns.get(i));
    }
    currentRow++;
}
2.
```

Als eerste maken we een nieuwe workbook aan. Dit is het grote Excel bestand. Daarna maken we een nieuw blad aan. Aangezien je maximaal één grid kan exporteren, hebben we geen extra bladen nodig.

Tenslotte maken we ook de headers van de sheet aan. Dit zijn de namen van de kolommen. (Bv: naam, voornaam, geboortedatum, etc..)

4.5.2. Merge

Wanneer de kolomnamen zijn aangemaakt, gaan we de rijen met data wegschrijven naar de workbook. Dit doen we aan de hand van de merge() methode:

```
1. @Override public void merge(PageIterator<?> iter, GridModelConfig gridConfig) {
    List<?> currentItem;

    BeanModel<?> beanModel = createBeanModel(gridConfig);
    do {
       currentItem = iter.next();
       for (Object transaction : currentItem) {
          Row row = sheet.createRow(currentRow);

          ArrayList<String> columns = new ArrayList<>(beanModel.getPropertyNames());
          for (int j = 0; j < columns.size(); j++) {
             String prop = columns.get(j);
             Cell cell = row.createCell(j);

             Object cellValue = getCellValue(beanModel, prop, transaction);

             if (cellValue != null) {
                cell.setCellValue(cellValue.toString());
             }
          }
          currentRow++;
          progressLog.increment();
       }
    } while (iter.hasNext());
}
2.
```

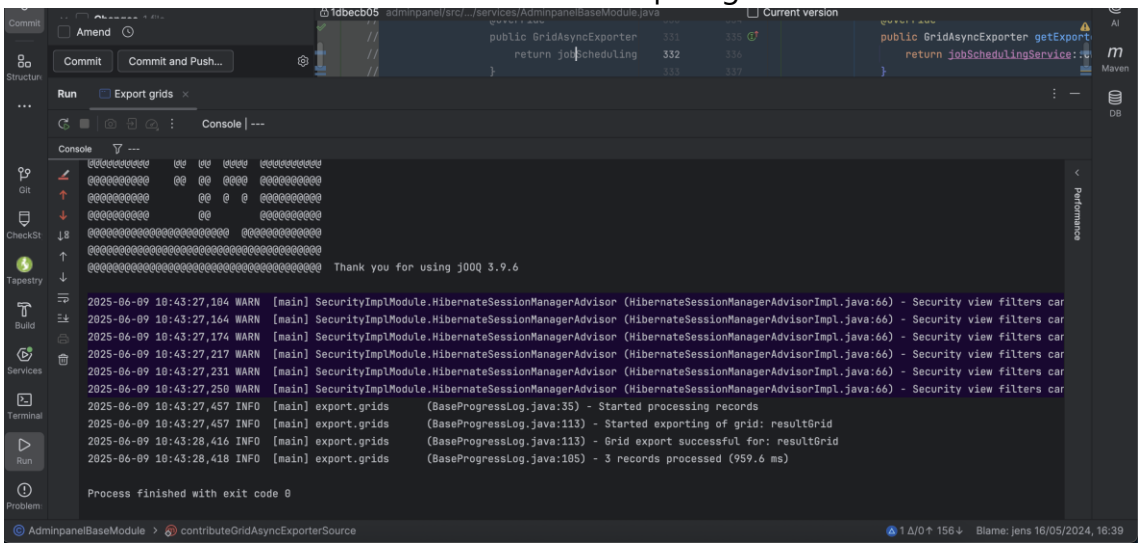
We itereren hier over alle data en schrijven deze in de juiste rij en kolom weg.

4.5.3. End merge

Wanneer alle data is weggeschreven, sluiten we de workbook af en slaan we het bestand op in een lokale map op de applicatie server (dit staat geconfigureerd in een argument in de job-scheduler)

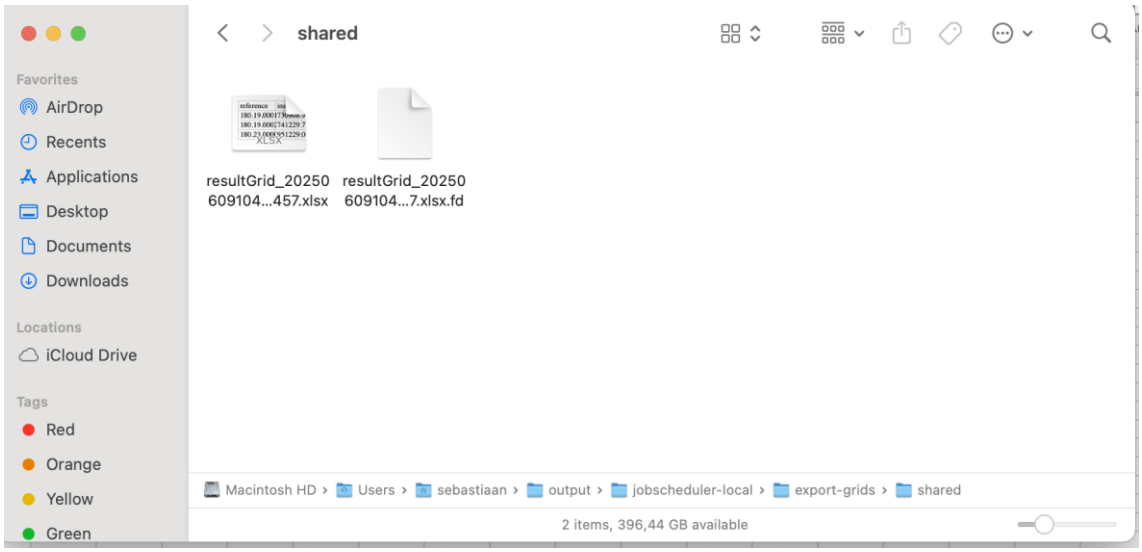
```
1. @Override public void endMerge(Path outputFile) {
    try (OutputStream fos = Files.newOutputStream(outputFile)) {
       workbook.write(fos);
       workbook.close();
    } catch (IOException e) {
       throw new RuntimeException("Error writing to Excel file: " + e.getMessage(), e);
    }
}
2.
```

In de console kunnen we ook zien dat deze export gelukt is.



Figuur 4.5.2: Export log van een grid

Het grid is nu geëxporteerd en staat op de lokale applicatie server.



Figuur 4.5.3: Resulterende bestanden na het exporteren

	A	B	C	D	E	F	G	H
1	reference	insz	name	gender	age	plan	employer	status
2	180.19.0001324	730908.911.57	Phoebe GAB Janes	MALE	51	Minima	Image Comics... (21-10-2013 - ...)	ACTIVE
3	180.19.0002509	741229.709.59	Darion Janes	MALE	50	Minima	Full Bleed Studios... (01-05-2018 - ...)	ACTIVE
4	180.23.0000074	951229.095.47	Josiah Jansen	MALE	29	Minima	Revil Comics... (22-12-2022 - ...)	ACTIVE
5								
6								
7								
8								
9								

Figuur 4.5.4: Voorbeeld van een geëxporteerd Excel bestand

4.6. Grids Dispatchen naar S3 server

De eindgebruiker kan niet aan de applicatieserver. Dit is gedaan uit veiligheidsoverwegingen. Om er voor te zorgen dat de gebruiker zijn/haar bestand kan downloaden, moeten we het verplaatsen vanuit de applicatie server naar een Amazon S3 fileserver. In adminpanel zullen we dan een link leggen naar dit bestand zodat de eindgebruiker deze kan downloaden.

Om een bestand over te zetten gebruikt PA een software genaamd FileDispatcher. Dit is proprietary software vanuit het bedrijf zelf. Ze hebben dit zelf geschreven omdat ze dit enorm vaak moeten doen, en deze code het enorm makkelijk maakt om dit te doen.

```
1. private void createDispatchMessage(Path path, Integer taskId, String user) {

    DispatchMessage dispatchMessage = new DispatchMessage.DispatchMessageBuilder()
          .setDocumentPath(path.toString())
          .setDispatchDate(today())
          .setDocumentType("EXPORT_GRID")
          .setTaskId(taskId)
          .setScope(Scope.USER)
          .setScopeValue(user)
          .build();

    String json = JsonUtilities.toJson(dispatchMessage);
    try {
       Files.writeString(path.resolveSibling(path.getFileName() + ".fd"), json);
    } catch (IOException e) {
       throw new RuntimeException(e);
    }
}
2.
```

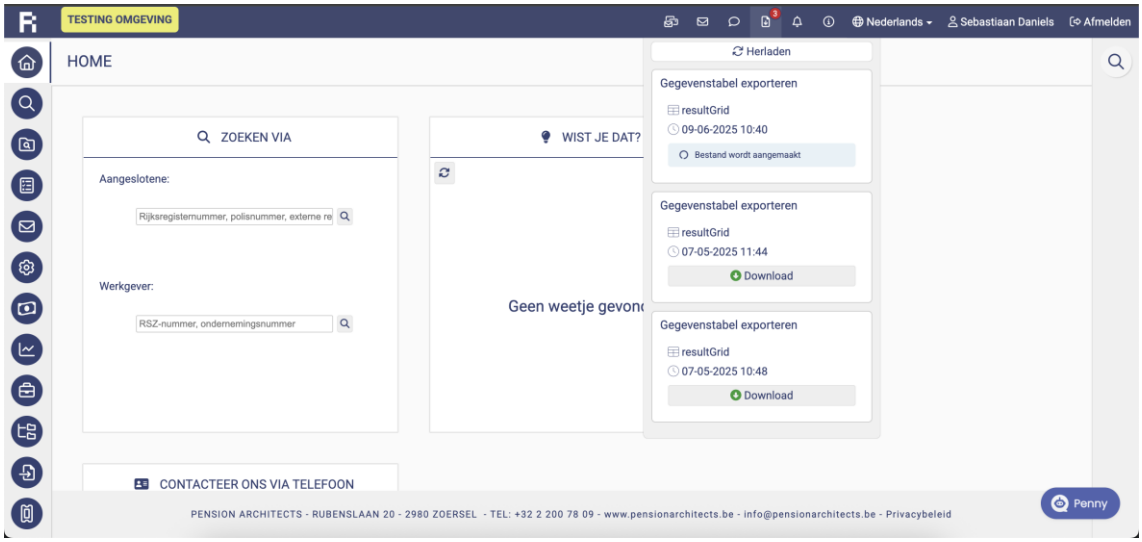
Met deze code wordt het bestand vanzelf naar de juiste plek gestuurd. Elke gebruiker heeft zijn eigen map op de S3 server die geconfigureerd is om te communiceren met de front-end. Adminpanel kan dus op elk moment de inhoud van deze map zien.

4.7. Downloadpagina voor de gebruiker

Wanneer het grid volledig is geexporteert, kan de gebruiker het downloaden in de webtoepassing. Hiervoor heb ik een nieuw component aangemaakt. In de menubalk verschijnt is er nu een icoon waar de gebruiker zijn downloads kan zien.

Figuur 4.7.1: Menubalk met nieuw download icoon

Wanneer de gebruiker hierover met zijn muis beweegt, krijgt deze een component met alle downloads te zien, alsook de status van de downloads die mogelijks nog bezig zijn.



Figuur 4.7.2: Downloadcomponent in de webtoepassing

Door op de download knop te drukken, wordt het geëxporteerde bestand naar de downloadfolder van de eindgebruiker gedownload. De gebruiker kan hier nu mee verder.

5. Security (Beveiliging)

Een enorm belangrijk deel van mijn stage is de beveiliging van het exporteren van deze grids. Het mag onmogelijk gebeuren dat iemand een grid exporteert en dan andere data krijgt te zien dan wat op de webtoepassing gegeven is.

Elke user heeft een heel complex rechtenpakket op basis van hun bedrijf en rol. Sommigen mogen een bepaald plan zien, anderen weer niet. Dit betekent dat twee bijna identieke gebruikers op hetzelfde grid toch andere data kunnen krijgen, en dit is enorm belangrijk om dat ook zo te houden.

Het probleem waarop ik was gestoten is dat de applicatieserver niet gebaseerd is op dit rechten systeem, maar gewoon permanent is ingelogd als systeem gebruiker met alle rechten. Dit betekent dus dat een gebruiker die misschien maar 10 rijen op een grid te zien kreeg, na het exporteren van een grid alle persoonsgegevens te zien kreeg.

Dit heb ik opgelost door de applicatie server bij elke export van user scope te laten veranderen. Bij het schedulen van een job, wordt automatisch de gebruiker meegegeven die de job heeft opgestart. Deze gebruik ik dan om de applicatieserver ook van gebruiker te laten veranderen, zodat deze dezelfde permissies heeft als de gebruiker zelf. Op het einde zet ik de gebruiker op de applicatie server terug naar de systeem gebruiker.

```
1. UserDetails previousUser =
(UserDetails)SecurityContextHolder.getContext().getAuthentication().getPrincipal();
UserDetails currentUser = userDetailsService.loadUserByUsername(getUser());

try {
    setUser(currentUser);
2.
```

```
1. private static void setUser(UserDetails currentUser) {
    SecurityContext ctx = new SecurityContextImpl();
    ctx.setAuthentication(
          new TestingAuthenticationToken(
                currentUser,
                "",
                new ArrayList<>(currentUser.getAuthorities())
          )
    );
    SecurityContextHolder.setContext(ctx);
}
2.
```

```
1. } finally {
    setUser(previousUser);
}
2.
```

6. Besluit

Tijdens mijn stageperiode bij Pension Architects heb ik het project om grids te kunnen exporteren naar Excel volledig kunnen afronden. Elk geconfigureerd grid wordt vanzelf met filter gecodeerd, doorgestuurd naar de job-scheduler, geëxporteerd, en doorgestuurd naar de eindgebruiker om te kunnen downloaden. Dit allemaal met juist geconfigureerde beveiliging en gebruikersgemak.

De volgende stap voor Pension Architects is om het systeem dat ik heb gemaakt voor alle overgebleven grids te implementeren. Ik heb tijdens mijn stage de meest belangrijke en grootste grids al zelf overgezet naar de nieuwe implementatie als zodat deze alvast in productie kunnen gaan en zodat toekomstige jobstudenten een goed en gemakkelijk voorbeeld hebben. Een week na de afloop van mijn stage is mijn project ook in productie genomen en staat momenteel online.

Daarnaast heb ik ook zeer concrete documentatie geschreven om het voor de personen die grids willen overzetten makkelijker te maken.

De stageperiode van 13 weken was een grote uitdaging. Ik had hiervoor nog nooit met Tapestry gewerkt, en ook de andere frameworks zoals POI en Liquibase moest ik vanuit het begin uit leren. Met behulp van de vriendelijke, behulpzame, en geduldige hulp van mijn collega's heb ik in deze 13 weken enorm veel kunnen leren over zowel de technische aspecten van het IT-werkveld, alsook de professionele sfeer. Hiervoor zou ik graag nog eens de collega's Jenthe Mariën, Kevin Aerts, en Nigel Riemis nogmaals extra voor willen bedanken.

Verder wil ik graag nog eens alle medewerkers van Pension Architects willen bedanken voor de opportuniteiten die ze mij hebben gegeven binnen het bedrijf om zowel op technisch alsook professioneel vlak te kunnen groeien.

Dankzij deze stage heb ik een eerste echte kennismaking kunnen maken met het bedrijfsleven. Deze stage heeft mij klaargemaakt voor een job als professionele bachelor in de IT. Hier kijk ik ook enorm naar uit.

LITERATUURLIJST

- *Pension Architects* (2009) Opgeroepen op 24 mei 2025, van pensionarchitects.be: https://www.pensionarchitects.be/
- *Pension Architects* (2025) Opgeroepen op 24 mei 2025, van pensionarchitects.be: https://www.pensionarchitects.be/services/
- *Jetbrains* (2000) Opgeroepen op 24 mei 2025, van jetbrains.com: https://www.jetbrains.com/help/idea/discover-intellij-idea.html#multi-platform-IDE
- *Java* (1995) Opgeroepen op 25 mei 2025, van java.com: https://www.java.com/en/download/help/whatis_java.html
- *Tapestry* (2006) Opgeroepen op 25 mei 2025, van apache.org: https://tapestry.apache.org/
- *Liquibase* (2006) Opgeroepen op 25 mei 2025, van liquibase.com: https://www.liquibase.com/how-liquibase-works
- *POI* (2001) Opgeroepen op 26 mei 2025, van apache.org: https://poi.apache.org/