

Consultant Beheer en Logging in Multi- Tenant Systemen

Realisatiedocument

Moussa Ramzi

Student Bachelor in de Toegepaste Informatica – Applicatieontwikkeling

Inhoudsopgave

1. INLEIDING	3
2. ANALYSE	4
2.1. Voordelen en nadelen van Visual Studio Code (VSCode)	4
2.2. Voordelen en nadelen van Node.js	5
2.3. Voordelen en nadelen van Knex	5
2.4. VOORDELEN en nadelen van SQL	6
2.5. Voordelen en nadelen van Swagger	6
3. REALISATIE	7
3.1. Gemeenschappelijke Structuur en Testaanpak per Module	7
3.1.1. Map Structuur	7
3.1.2. Controllers	9
3.1.3. Filters	10
3.1.4. In-Memory Testing	11
3.2. Consultant Management	12
3.2.1. Functies en implementaties	13
3.3. Login Logs	18
3.3.1. Functies en implementaties	18
3.4. Dashboard	21
3.4.1. Functies en implementaties	21
3.5. Audit logging	26
3.6. Wisselen van Tenant	28
4. BESLUIT	30
LITERATUURLIJST	31

1. Inleiding

Dit realisatiedocument bouwt voort op het projectplan dat aan het begin van mijn stage werd opgesteld. In dat plan werd de opdracht omschreven om bestaande interne tooling uit te breiden met een overzichtelijke en betrouwbare manier om consultant logins te registreren en te raadplegen binnen verschillende tenants van het systeem.

De kern van deze opdracht was het verbeteren van de transparantie en controle over consultantactiviteit: waar logt een consultant in, wanneer en waarom? Hiervoor werd gevraagd om een nieuwe module te bouwen die het mogelijk maakt deze gegevens op een gestructureerde manier te verzamelen en op te vragen.

De technische basis van de applicatie zoals de in-memory repositories, het CQRS-architectuurprincipe (Command Query Responsibility Segregation) en het gebruik van DTO's was al aanwezig in andere modules van het systeem. Mijn taak bestond erin om deze bestaande aanpak correct toe te passen in een nieuwe context: de Consultant Management Module, die specifiek ontwikkeld werd om loginlogs per consultant inzichtelijk te maken.

Daarnaast werden enkele bijkomende verbeteringen aangebracht, zoals:

- Het toevoegen van consultantinformatie aan het bearer token om audit logging te verbeteren,
- Het ondersteunen van tenant switching zonder dat gebruikers eerst moeten uitloggen,
- En het correct verwerken van consultantimpersonatie, met behoud van traceerbaarheid in de audit logs.

In dit document wordt de realisatie van deze componenten uitgebreid toegelicht. Hoofdstuk 2 beschrijft de gebruikte tools en technologieën. In hoofdstuk 3 volgt een gedetailleerde bespreking van de structuur, implementatie en werking van de verschillende modules, inclusief voorbeeldcode en technische onderbouwing. Tot slot wordt in hoofdstuk 4 gereflecteerd op het eindresultaat en de toegevoegde waarde van de uitgevoerde werkzaamheden.

2. Analyse

In dit document presenteren we een overzicht van de gebruikte tools en technologieën die we hebben ingezet voor de uitvoering van onze opdracht. De keuze voor deze tools is gebaseerd op de bestaande infrastructuur en standaarden binnen ons bedrijf. Hoewel er geen uitgebreide vergelijking van mogelijke ontwikkelomgevingen en technologieën is uitgevoerd, is het belangrijk om de gebruikte tools en de redenen voor hun gebruik te documenteren.

Voor deze opdracht hebben we de volgende tools en technologieën gebruikt:

- Visual Studio Code: Een veelzijdige en populaire code-editor die ondersteuning biedt voor verschillende programmeertalen en extensies.
- Node.js: Een JavaScript-runtime die het mogelijk maakt om server-side applicaties te ontwikkelen.
- Knex: Een SQL query builder voor Node.js, die het schrijven van database queries vereenvoudigt.
- SQL: De standaardtaal voor het beheren en manipuleren van relationele databases.
- Swagger: Een tool voor het ontwerpen, bouwen en documenteren van RESTful API's.

In de volgende hoofdstukken zullen we elk van deze tools in detail bespreken, inclusief de redenen waarom deze tools binnen ons bedrijf zijn gekozen en hoe ze bijdragen aan de efficiëntie en kwaliteit van onze ontwikkelprocessen. We beginnen met een analyse van Visual Studio Code.

2.1. Voordelen en nadelen van Visual Studio Code (VSCode)

Voordelen:

- Gebruiksvriendelijkheid: VSCode biedt een intuïtieve interface die gemakkelijk te navigeren is, zelfs voor beginners.
- Extensies: Er is een breed scala aan extensies beschikbaar, waardoor de functionaliteit van de editor kan worden uitgebreid naar verschillende programmeertalen en frameworks.
- Integratie: VSCode integreert goed met versiebeheersystemen zoals Git, wat essentieel is voor teamontwikkeling.
- Prestaties: Ondanks de vele functies blijft VSCode lichtgewicht en snel, wat bijdraagt aan een efficiënte ontwikkelervaring.

Nadelen:

- Complexiteit van configuratie: Voor sommige gebruikers kan het configureren van VSCode en het instellen van de juiste extensies complex en tijdrovend zijn.
- Resourcegebruik: Hoewel VSCode relatief lichtgewicht is, kan het bij intensief gebruik van extensies meer systeembronnen verbruiken.
- Beperkte ingebouwde functies: Sommige functies die in andere IDE's standaard zijn ingebouwd, vereisen in VSCode het gebruik van externe extensies.

2.2. Voordelen en nadelen van Node.js

Voordelen:

- **Asynchrone verwerking:** Node.js maakt gebruik van een event-driven, non-blocking I/O model, wat zorgt voor efficiënte verwerking van meerdere gelijktijdige verzoeken.
- **Snelheid:** Dankzij de V8 JavaScript-engine van Google is Node.js zeer snel in het uitvoeren van JavaScript-code.
- **Grote community:** Er is een uitgebreide en actieve community die bijdraagt aan een groot aantal beschikbare modules en pakketten via npm (Node Package Manager).
- **JavaScript:** Het gebruik van JavaScript aan zowel de client- als serverzijde vereenvoudigt de ontwikkeling en vermindert de noodzaak om meerdere talen te leren.

Nadelen:

- **Single-threaded:** Hoewel Node.js asynchrone verwerking ondersteunt, is het single-threaded, wat kan leiden tot prestatieproblemen bij CPU-intensieve taken.
- **Callback hell:** Het intensieve gebruik van callbacks kan leiden tot moeilijk leesbare en onderhoudbare code, hoewel dit deels kan worden opgelost met Promises en async/await.
- **Minder geschikt voor CPU-intensieve taken:** Node.js is minder geschikt voor taken die veel CPU-kracht vereisen, zoals beeld- of videobewerking.

2.3. Voordelen en nadelen van Knex

Voordelen:

- **Flexibiliteit:** Knex biedt ondersteuning voor meerdere database-engines, zoals PostgreSQL, MySQL, SQLite en MSSQL, waardoor het een veelzijdige keuze is voor verschillende projecten.
- **Query builder:** De SQL query builder maakt het schrijven van complexe queries eenvoudiger en leesbaarder.
- **Migraties:** Knex biedt ingebouwde ondersteuning voor database-migraties, wat helpt bij het beheren van database-schema's over verschillende omgevingen.
- **Transacties:** Ondersteuning voor transacties zorgt voor veilige en betrouwbare databasebewerkingen.

Nadelen:

- **Leercurve:** Voor ontwikkelaars die niet bekend zijn met SQL-query builders, kan Knex in het begin een steile leercurve hebben.
- **Documentatie:** Hoewel de documentatie van Knex uitgebreid is, kan deze soms verwarrend zijn voor beginners.
- **Afhankelijkheid van SQL-kennis:** Hoewel Knex het schrijven van queries vereenvoudigt, blijft een goede kennis van SQL noodzakelijk.

2.4. VOORDELEN en nadelen van SQL

Voordelen:

- **Standaardisatie:** SQL is een gestandaardiseerde taal, wat betekent dat kennis van SQL overdraagbaar is tussen verschillende relationele databases.
- **Krachtig en veelzijdig:** SQL biedt krachtige mogelijkheden voor het beheren, manipuleren en ophalen van gegevens.
- **Breed ondersteund:** SQL wordt ondersteund door vrijwel alle relationele databasesystemen, zoals MySQL, PostgreSQL, Oracle en SQL Server.
- **Complexe queries:** SQL maakt het mogelijk om complexe queries te schrijven voor geavanceerde data-analyse en rapportage.

Nadelen:

- **Complexiteit:** Het schrijven van complexe SQL-queries kan moeilijk en foutgevoelig zijn, vooral voor beginners.
- **Prestaties:** Slecht geoptimaliseerde SQL-queries kunnen leiden tot prestatieproblemen, vooral bij grote datasets.
- **Beperkte schaalbaarheid:** Relationele databases kunnen minder goed schalen dan sommige NoSQL-oplossingen, vooral bij zeer grote hoeveelheden ongestructureerde data.

2.5. Voordelen en nadelen van Swagger

Voordelen:

- **API-documentatie:** Swagger biedt uitgebreide mogelijkheden voor het automatisch genereren van API-documentatie, wat de communicatie en samenwerking tussen ontwikkelaars vergemakkelijkt.
- **Interactiviteit:** De interactieve API-documentatie stelt gebruikers in staat om API-eindpunten direct vanuit de documentatie te testen.
- **Standaardisatie:** Swagger ondersteunt de OpenAPI-specificatie, wat zorgt voor gestandaardiseerde en consistente API-definities.
- **Tooling:** Er is een breed scala aan tools beschikbaar binnen het Swagger-ecosysteem, zoals Swagger Editor, Swagger UI en Swagger Codegen.

Nadelen:

- **Complexiteit:** Het opzetten en configureren van Swagger kan complex zijn, vooral voor grote en uitgebreide API's.
- **Overhead:** Het gebruik van Swagger kan extra overhead met zich meebrengen in termen van onderhoud en updates van de API-documentatie.
- **Leercurve:** Voor ontwikkelaars die niet bekend zijn met de OpenAPI-specificatie, kan het leren en implementeren van Swagger een uitdaging zijn.

3. Realisatie

Dit hoofdstuk en de volgende bevatten een gedetailleerde beschrijving van het resultaat van mijn stageopdracht. In deze secties zal ik de verschillende functionaliteiten, de opzet van de toepassing, en de toegepaste principes en patronen toelichten. Het doel is om een duidelijk beeld te geven van wat er is gerealiseerd, zonder in alle details te treden.

We zullen ingaan op de belangrijkste componenten van de toepassing, ondersteund door kleine stukjes code en schermvoorbeelden om de werking te illustreren. Deze beschrijving biedt inzicht in de technische keuzes en implementaties die zijn gedaan om de uiteindelijke oplossing te realiseren.

Ter verduidelijking: mijn werkzaamheden tijdens deze stage waren volledig gericht op de backend-ontwikkeling. De schermvoorbeelden die in dit hoofdstuk worden getoond dienen uitsluitend ter illustratie van hoe de door mij ontwikkelde backend-functionaliteiten in de praktijk werken via de bestaande frontend-interface.

3.1. Gemeenschappelijke Structuur en Testaanpak per Module

Elke module binnen de toepassing is opgebouwd volgens een uniforme structuur. Dit zorgt voor herbruikbaarheid, onderhoudbaarheid en een gestroomlijnde ontwikkel- en testaanpak. In deze sectie worden de generieke patronen besproken die in elke module zijn toegepast, met betrekking tot structuur, controllers, filters en in-memory testing.

3.1.1. Map Structuur

Elke module binnen het project volgt een consistente en duidelijke mappenstructuur, waarbij de logica is opgesplitst in drie hoofdcomponenten: infrastructure, domain, en query. Deze indeling bevordert de scheiding van verantwoordelijkheden en ondersteunt het CQRS-principe.

1. Infrastructure

Deze map bevat de infrastructuurspecifieke onderdelen van de module, en bestaat doorgaans uit:

- **Repositories:** Hier worden de klassen geplaatst die verantwoordelijk zijn voor het benaderen van de onderliggende opslaglaag (zoals een database of een in-memory datastructuur).
- **API:** Bevat de publieke interface van de module naar de buitenwereld, en is meestal opgedeeld in:
 - **Controllers:** Definiëren de verschillende endpoints van de module en verwerken inkomende API-verzoeken.
 - **DTOs (Data Transfer Objects):** Structuren die worden gebruikt om gegevens over te dragen tussen de client en de back-endlogica op een type-veilige manier.

2. Domain:

De domain-map bevat de kernlogica van de module die verantwoordelijk is voor het uitvoeren van schrijfoperaties (commando's). Dit omvat:

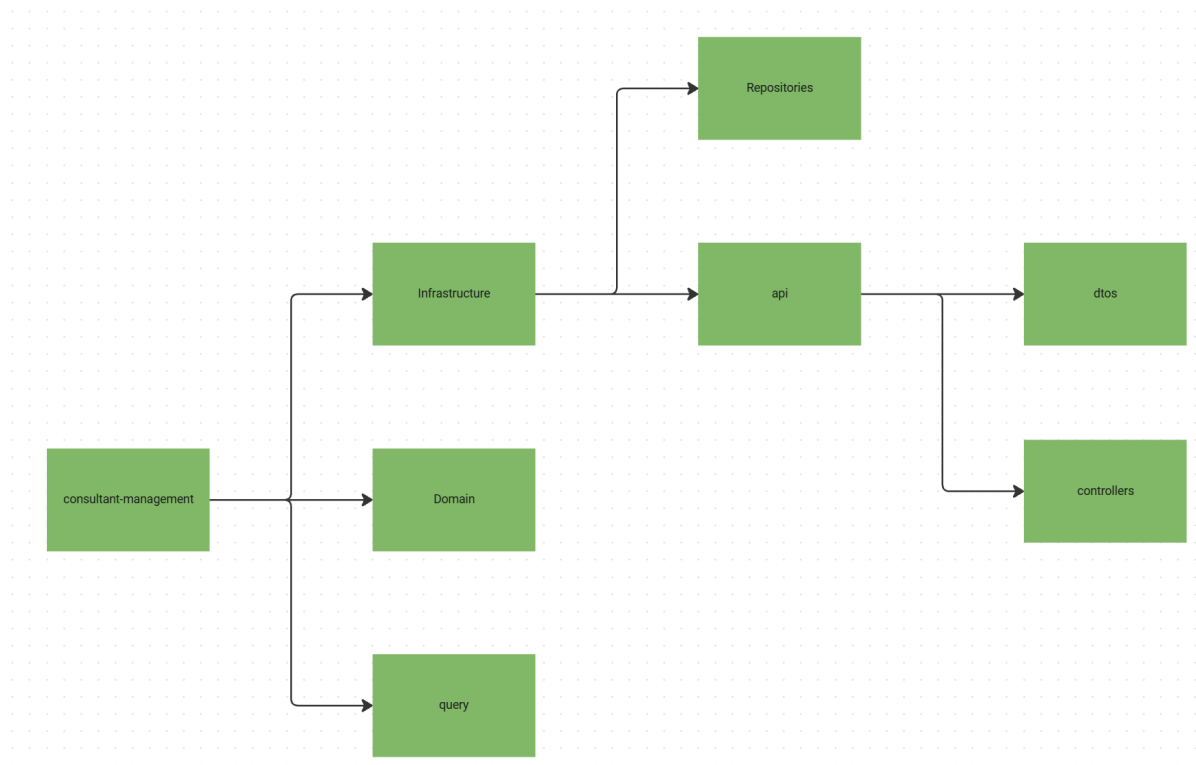
- Interfaces voor repositories
- In-memory implementaties of businesslogica die domeinregels afdwingt

3. Query:

Deze map bevat de logica die gericht is op het uitlezen van data. Net als in domain bestaat dit uit:

- Repositoryinterfaces voor leesoperaties
 - In-memory implementaties die data samenstellen en structureren voor presentatie
- Voor een visuele

Een illustratie van de map structuur:



3.1.2. Controllers

Elke module bevat één of meerdere controllers die verantwoordelijk zijn voor het afhandelen van inkomende API-aanvragen. De controllers bevinden zich steeds in de map `api/controllers` van de module, en volgen overal een gelijkaardige opbouw. Hieronder worden de belangrijkste kenmerken van deze structuur toegelicht:

- Uniforme structuur en naming: Elke controller is duidelijk gekoppeld aan een bepaald domein of entiteit, bijvoorbeeld `TenantsController` of `ConsultantsController`, en wordt voorzien van een routeprefix zoals `/consultant-management/tenants`.
- Toegangscontrole met decorators: Bovenaan de controller wordt via de `@RequirePermission(...)` decorator aangeduid welke gebruikersrechten vereist zijn om toegang te krijgen tot de endpoints. Dit zorgt voor consistente en modulebrede toegangscontrole op basis van rollen of permissies.
- Swagger-documentatie: Elke endpoint is gedocumenteerd met behulp van Swagger-decorators zoals `@ApiOkResponse`, `@ApiCreatedResponse`, `@ApiBearerAuth` en `@ApiTags`. Hierdoor wordt automatisch API-documentatie gegenereerd die bruikbaar is voor zowel front-end developers als testers.
- Gebruik van DTO's en Validatie: Input- en outputgegevens worden strikt gedefinieerd via DTO's (Data Transfer Objects), zoals `AssignConsultantDto` of `ConsultantFilterDto`. Voor validatie wordt gebruikgemaakt van `ValidationPipe`, wat zorgt voor betrouwbare en veilige verwerking van gebruikersinput.
- Exception handling: Veelvoorkomende fouten, zoals het niet vinden van een entiteit (`NotFoundException`) of conflictgevallen, worden expliciet afgehandeld en op een uniforme manier teruggegeven aan de gebruiker.
- Delegatie naar Facade of Service: De controllers bevatten zelf geen businesslogica. Ze delegeren taken aan een facade of serviceklasse (zoals `ConsultantManagementFacade`), wat de controller eenvoudig en goed testbaar houdt.

Een voorbeeld van een typische controller:

```
...
@RequirePermission(Permission.ReadMessage)
@Controller('/reporting/consultant')
@ApiTags('reporting')
@ApiBearerAuth()
@ApiOAuth2([])
export class ConsultantReportingController {
  constructor(private readonly reporting: ReportingFacade) {}

  @Get('logins-weekday')
  @ApiOkResponse({ description: 'Logins per weekday', type: ConsultantLoginsPerWeekdayDto })
  async getConsultantLoginsPerWeekday(@Query() filter: DateFilterDto): Promise<ConsultantLoginsPerWeekdayDto> {
    return this.reporting.getConsultantLoginsPerWeekday({
      startDate: filter.startDate ? new Date(filter.startDate) : undefined,
      endDate: filter.endDate ? new Date(filter.endDate) : undefined,
    });
  }
}
```

3.1.3. Filters

Voor het verwerken van query parameters, zoals zoektermen of paginatie, wordt in alle modules gebruikgemaakt van filters:

- Deze filters zijn geïmplementeerd als DTO's met optionele en verplichte parameters.
- Ze ondersteunen zaken zoals gebruikersnaamfilters, tenantenamen, of toegang tot alle tenants.
- De filters worden direct doorgegeven aan de query-methodes, wat zorgt voor een duidelijke scheiding tussen inputverwerking en businesslogica.

Dit maakt de controllercode eenvoudiger en de filtering flexibeler.

Een voorbeeld van een filter DTO:

```
export class ConsultantTenantsFilterDto extends PaginationBaseDto {
  @ApiPropertyOptional({
    name: 'tenantName',
    required: false,
    type: String,
    examples: {
      none: { value: undefined },
      DEV_STAGE: { value: 'DEV_STAGE' },
    },
  })
  @IsOptional()
  @IsString()
  tenantName?: string;
}
```

3.1.4. In-Memory Testing

Voor deze module werd gekozen om de dataopslag tijdens ontwikkeling en testing in-memory te houden. Deze keuze vermijdt afhankelijkheden van externe systemen en maakt het mogelijk om snel en gecontroleerd functionaliteit te valideren.

De implementatie is bewust opgesplitst in twee afzonderlijke repositorytypes, in lijn met het CQRS-principe:

- De query repository biedt methodes om gegevens op te vragen die nodig zijn voor leesoperaties. Deze repository verwerkt interne state en transformeert die naar DTO's (Data Transfer Objects) die rechtstreeks door de API kunnen worden teruggegeven.
- De domain repository beheert alle schrijfoperaties, zoals het toekennen of verwijderen van toegang en het aanmaken van nieuwe domeinobjecten. Hierbij wordt alle interne state bijgehouden in lokale datastructuren (zoals arrays), die functioneren als tijdelijke opslag van consultants, tenants en hun onderlinge relaties.

Elke repository maakt gebruik van eenvoudige JavaScript/TypeScript-dataobjecten om domeinstaat bij te houden. Bij leesoperaties worden de nodige relaties opgebouwd aan de hand van deze state. Bij schrijfoperaties wordt de state doelgericht aangepast volgens de domeinregels, bijvoorbeeld door relaties toe te voegen, te filteren of te verwijderen.

Deze aanpak maakt het mogelijk om de werking van de use cases realistisch te simuleren, zonder connectie met een echte database. Dit is met name nuttig tijdens unit tests of bij lokale ontwikkeling.

Een voorbeeld van een in-memory repository:

```
...
export interface ConsultantRepository {
  assignSpecificTenants(consultantId: string, tenantIds: string[]): Promise<void>;
}
...
export class InMemoryConsultantRepository implements ConsultantRepository {
  constructor(
    private consultants: ConsultantState[] = [],
    private consultantTenantAssignments: ConsultantTenantAssignmentState[] = [],
  ) {}

  public async assignSpecificTenants(consultantId: string, tenantIds: string[]): Promise<void> {
    if (tenantIds.length) {
      tenantIds.forEach((tenantId) => {
        this.consultantTenantAssignments.push({
          consultantId,
          tenantId,
        });
      });
    }
  }
}
```

3.2. Consultant Management

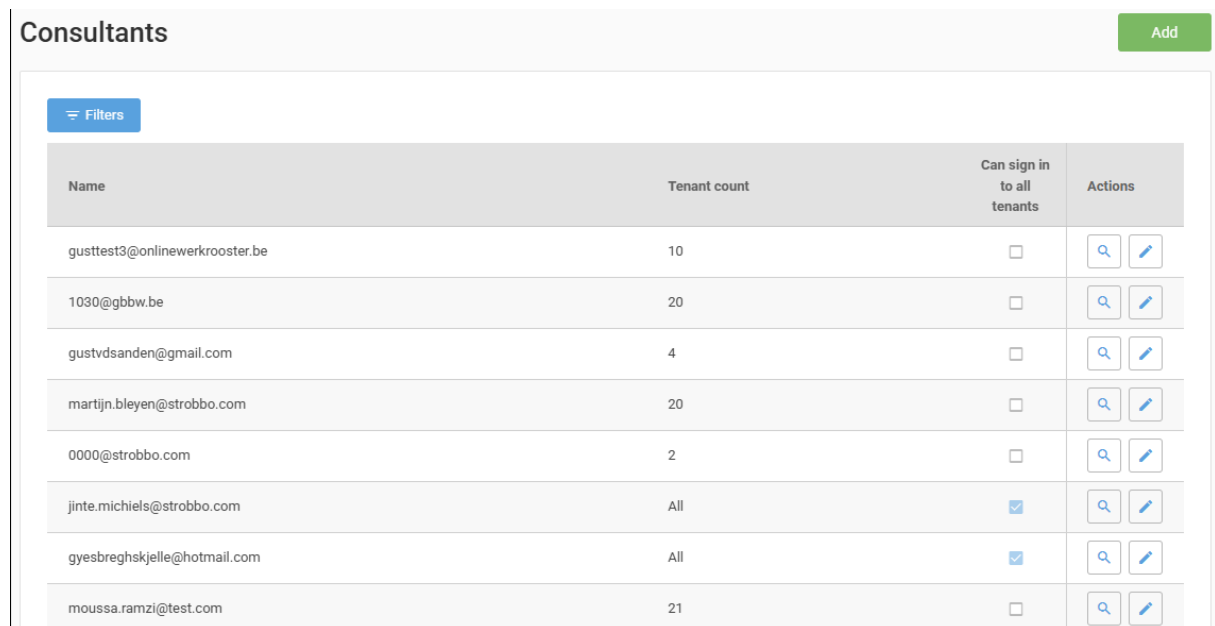
De Consultant Management Module is de eerste en ook grootste module die ik tijdens mijn stage volledig vanaf nul heb ontwikkeld. Deze module bevat de kernfunctionaliteit voor het beheren van consultants en hun toegang tot verschillende tenants binnen het systeem.

Bij de ontwikkeling is gekozen voor een modulaire en overzichtelijke structuur, die in lijn ligt met de architecturale keuzes zoals besproken in het vorige hoofdstuk. Zo is er gebruikgemaakt van een duidelijke opdeling in infrastructuur, domain en query, en is het CQRS-principe (Command Query Responsibility Segregation) toegepast om lees- en schrijfflogica van elkaar te scheiden.

Daarnaast zijn voor de opslag en afhandeling van gegevens in deze module in-memory implementaties gebruikt. Deze benadering maakte het mogelijk om de functionaliteiten te ontwikkelen en testen zonder directe afhankelijkheid van een echte database. Dit was met name handig in de beginfase van het project. De in-memory repositories zijn zowel toegepast in het domain-gedeelte (voor schrijfacties) als in het query-gedeelte (voor leesacties).

Ook zijn er Data Transfer Objects (DTO's) gebruikt om gegevens gestructureerd over te brengen tussen de API-laag en de interne applicatielogica. De werking van deze DTO's is eerder besproken in hoofdstuk 2, en wordt hier toegepast bij onder andere het ophalen van consultantgegevens of het aanmaken van een nieuwe consultant.

In onderstaande afbeelding is een overzicht te zien van de gebruikersinterface waarin alle consultants binnen het systeem worden weergegeven. Dit scherm illustreert hoe de ontwikkelde functionaliteiten in de praktijk worden gebruikt en toont de resultaten van de query-operaties die via de DTO's worden afgehandeld.



Name	Tenant count	Can sign in to all tenants	Actions
gusttest13@onlinewerkrooster.be	10	<input type="checkbox"/>	Search Edit
1030@gbbw.be	20	<input type="checkbox"/>	Search Edit
gustvdsanden@gmail.com	4	<input type="checkbox"/>	Search Edit
martijn.bleyen@strobbo.com	20	<input type="checkbox"/>	Search Edit
0000@strobbo.com	2	<input type="checkbox"/>	Search Edit
jinte.michiels@strobbo.com	All	<input checked="" type="checkbox"/>	Search Edit
gyesbreghtskjelle@hotmail.com	All	<input checked="" type="checkbox"/>	Search Edit
moussa.ramzi@test.com	21	<input type="checkbox"/>	Search Edit

In de volgende sectie worden de belangrijkste functies van deze module toegelicht, ondersteund door visuele afbeeldingen die de praktische werking illustreren. Elke functie is ontworpen met aandacht voor foutafhandeling, validatie, en duidelijke scheiding van verantwoordelijkheden. De bijbehorende logica wordt afgehandeld via de eerdergenoemde repositories, waarbij de data via controllers en DTO's in de infrastructuurlaag wordt verwerkt.

3.2.1. Functies en implementaties

In deze sectie worden de belangrijkste functies van de Consultant Management Module beschreven. Elke functie speelt een cruciale rol in het beheren van consultants en hun toewijzingen aan tenants. Hieronder volgt een gedetailleerde uitleg van de werking van elke functie.

1. **getConsultant:**

- **Doel:** Haalt de gegevens van een specifieke consultant op.
- **Werking:** Controleert eerst of de consultant bestaat in het systeem. Als de consultant niet gevonden wordt, wordt een foutmelding gegenereerd. Vervolgens worden de gegevens van de consultant opgehaald en geretourneerd.

2. **getTenant:**

- **Doel:** Haalt de gegevens van een specifieke tenant op, inclusief de toegewezen consultants.
- **Werking:** Controleert eerst of de tenant bestaat in het systeem. Als de tenant niet gevonden wordt, wordt een foutmelding gegenereerd. Vervolgens worden de gegevens van de tenant opgehaald en geretourneerd.

3. **findConsultants:**

- **Doel:** Haalt een gepagineerde lijst van consultants op op basis van de opgegeven zoekcriteria.
- **Werking:** Voert een zoekopdracht uit waarbij de pagina, grootte, gebruikersnaam en toegangsrechten worden meegenomen. Deze functie retourneert een gepagineerd overzicht met de gevonden consultants.

4. **getAssignedTenantIds:**

- **Doel:** Haalt een lijst van tenant IDs op die aan een specifieke consultant zijn toegewezen.
- **Werking:** Voert een opvraging uit om alle tenant toewijzingen voor de opgegeven consultant te verkrijgen. Deze functie retourneert een lijst van tenant IDs.

5. **getTenantsByConsultantId:**

- **Doel:** Haalt een gepagineerde lijst van tenants op die aan een specifieke consultant zijn toegewezen.
- **Werking:** Controleert eerst of de consultant bestaat in het systeem. Als de consultant niet gevonden wordt, wordt een foutmelding gegenereerd. Vervolgens worden alle tenants opgehaald die aan deze consultant zijn toegewezen, rekening houdend met paginering en eventuele naamfilters.

In onderstaande afbeelding is het detailscherm van een consultant weergegeven, waarin alle aan deze consultant toegewezen tenants zichtbaar zijn. Dit scherm toont de praktische toepassing van de `getTenantsByConsultantId` functionaliteit en illustreert hoe de gepagineerde lijst van tenants wordt gepresenteerd aan de gebruiker.

Consultant details: moussa.ramzi@test.com

Back

Tenant access

Revoke all access

Edit

Name	Actions
DEV-mystroautomatedtesting	
DEV_Barbet	
DEV_BAVET	
DEV_colmar_payroll_data	

6. `getConnectedConsultants`:

- **Doel:** Haalt een gepagineerde lijst van consultants op die verbonden zijn met een specifieke tenant.
- **Werking:** Controleert eerst of de tenant bestaat in het systeem. Als de tenant niet gevonden wordt, wordt een foutmelding gegenereerd. Vervolgens worden alle consultants opgehaald die toegang hebben tot deze tenant, rekening houdend met paginering en toegangsrechten.

In onderstaande afbeelding is het omgekeerde perspectief weergegeven van de `getTenantsByConsultantId` functionaliteit. Vanuit het klantenniveau (tenant) wordt een overzicht getoond van alle consultants die toegang hebben tot deze specifieke klant. Dit scherm illustreert hoe de `getConnectedConsultants` functionaliteit wordt gebruikt om vanuit een tenant-perspectief inzicht te krijgen in welke consultants er toegangsrechten hebben, wat essentieel is voor het beheren van klanttoegang en het waarborgen van de juiste beveiligingsniveaus

Tenant DEV_stage (#404)

<< DEV_Sfen / DEV_StrobboMain02_E2E01 >>

Go back to list

Environment

Failed messages

File storage

Message board

User profile

User attachments

Legal agreements

Logs

Consultants

Filters

Add

Consultant Name	Can sign in to all tenants	Actions
cypress.consultant@owr.be	<input checked="" type="checkbox"/>	
gusttest@onlinewerkrooster.be	<input checked="" type="checkbox"/>	
gyesbregghskjelle@hotmail.com	<input checked="" type="checkbox"/>	
info@onlinewerkrooster.be	<input checked="" type="checkbox"/>	
jinte.michiels@strobbo.com	<input checked="" type="checkbox"/>	
michael.voorhaen@gmail.com	<input checked="" type="checkbox"/>	

7. assignTenants:

- **Doel:** Wijst specifieke tenants toe aan een consultant of verleent toegang tot alle tenants.
- **Werking:** Controleert eerst of de consultant bestaat in het systeem. Als de consultant niet gevonden wordt, wordt een foutmelding gegenereerd. Vervolgens worden de toegangsrechten bijgewerkt. Als de consultant al volledige toegang heeft, wordt een conflictmelding gegenereerd. Bij specifieke tenant toewijzingen worden deze individueel toegewezen.

Onderstaande afbeelding toont de functionaliteit voor het toewijzen van tenant toegang aan een consultant. In deze modal kunnen specifieke tenants geselecteerd worden uit een lijst van beschikbare tenants. Daarnaast is er een checkbox "Link the consultant with all the tenants" waarmee in één keer volledige toegang kan worden verleend tot alle tenants in het systeem. Dit illustreert de praktische implementatie van de assignTenants functionaliteit, waarbij zowel granulaire als globale toegangsrechten kunnen worden beheerd.

8. createConsultant:

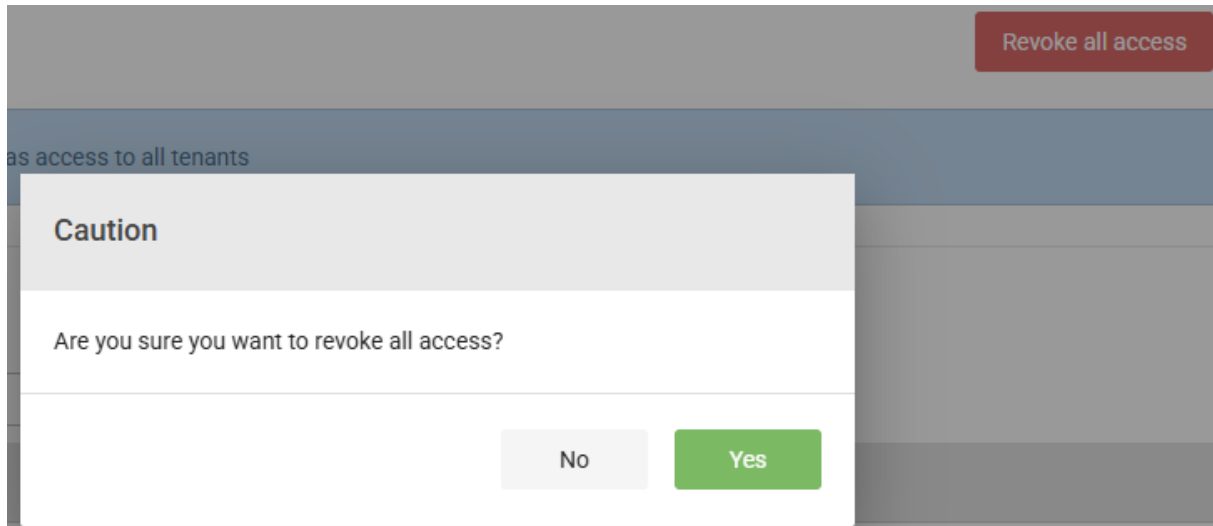
- **Doel:** Creëert een nieuwe consultant en wijst eventueel specifieke tenants toe.
- **Werking:** Controleert eerst of de gebruiker bestaat in het systeem. Als de gebruiker niet gevonden wordt, wordt een foutmelding gegenereerd. Vervolgens wordt gecontroleerd of de consultant al bestaat. Als dit het geval is, wordt een conflictmelding gegenereerd. Daarna wordt de nieuwe consultant aangemaakt en worden eventuele tenant toewijzingen uitgevoerd.

Onderstaande afbeelding toont het aanmaakproces van een nieuwe consultant in de gebruikersinterface. Hierin is te zien hoe eerst een gebruiker wordt geselecteerd uit het systeem, waarna direct meerdere tenants kunnen worden toegewezen aan de nieuwe consultant. Dit illustreert de praktische implementatie van de createConsultant functionaliteit.

9. deleteTenantAccess:

- **Doel:** Verwijdert alle tenant toegang voor een specifieke consultant.
- **Werking:** Controleert eerst of de consultant bestaat in het systeem. Als de consultant niet gevonden wordt, wordt een foutmelding gegenereerd. Als de consultant volledige toegang heeft, wordt deze ingetrokken. Vervolgens worden alle individuele tenant toewijzingen verwijderd.

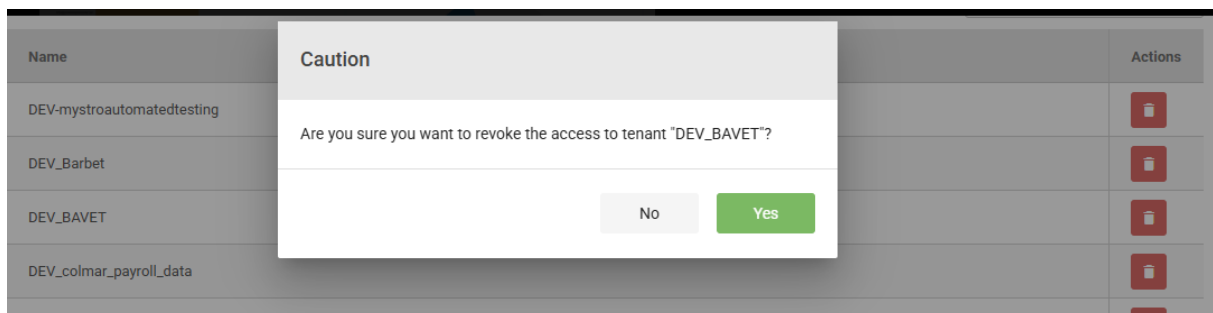
In onderstaande afbeelding is de functionaliteit voor het intrekken van alle tenant toegang zichtbaar. Rechtsboven in het scherm bevindt zich de knop "Revoke All Access" waarmee alle toegangsrechten van een consultant kunnen worden ingetrokken. Het bevestigingsvenster (modal) toont de veiligheidsmaatregel die gebruikers waarschuwt voordat de actie definitief wordt uitgevoerd.



10. revokeSelectedAccess:

- **Doel:** Trekt specifieke tenant toegang in voor een consultant.
- **Werking:** Controleert eerst of de consultant bestaat in het systeem. Als de consultant niet gevonden wordt, wordt een foutmelding gegenereerd. Vervolgens wordt gecontroleerd of de opgegeven tenants bestaan. Als geen van de tenants gevonden wordt, wordt een foutmelding gegenereerd. Daarna wordt de toegang tot de opgegeven tenants ingetrokken.

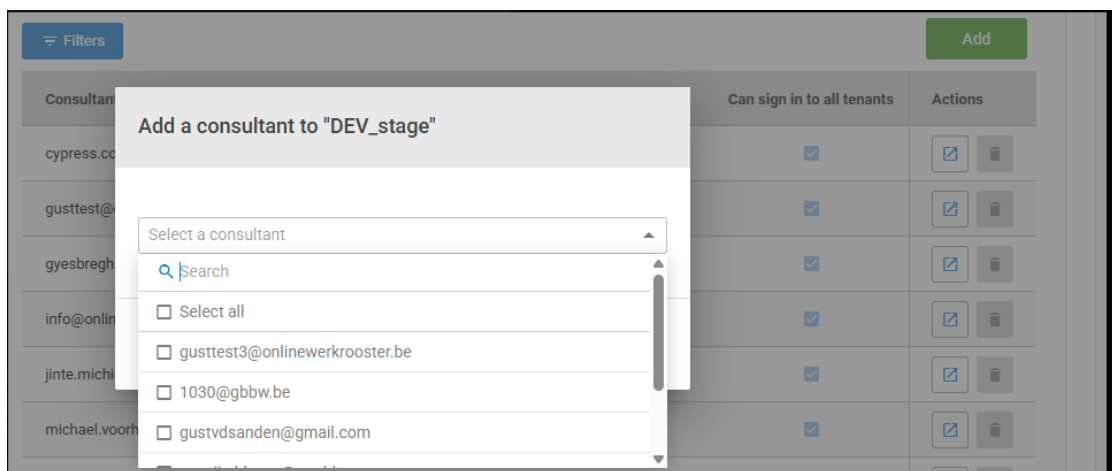
Onderstaande afbeelding illustreert hoe individuele tenant toegang kan worden ingetrokken. Naast elke tenant in de lijst staat een prullenbak-icoon waarmee specifieke tenant toewijzingen kunnen worden verwijderd. Ook hier wordt een bevestigingsmodal getoond om onbedoelde acties te voorkomen, wat de veiligheid van het systeem waarborgt.



11. assignConsultantsToTenant:

- **Doel:** Wijst consultants toe aan een specifieke tenant.
- **Werking:** Controleert eerst of de tenant bestaat in het systeem. Als de tenant niet gevonden wordt, wordt een foutmelding gegenereerd. Vervolgens wordt gecontroleerd welke consultants bestaan en toegang kunnen krijgen tot de tenant. Consultants die al toegang hebben worden gefilterd. Als alle opgegeven consultants al toegang hebben, wordt een conflictmelding gegenereerd. Daarna worden de nieuwe toewijzingen uitgevoerd.


Onderstaande afbeelding toont het omgekeerde perspectief van de tenant toewijzing functionaliteit. Vanuit het tenant-niveau kan nu een lijst van beschikbare consultants worden weergegeven die toegewezen kunnen worden aan deze specifieke tenant. Deze modal illustreert de praktische implementatie van de assignConsultantsToTenant functionaliteit, waarbij meerdere consultants tegelijkertijd kunnen worden geselecteerd en toegewezen aan een tenant. Het systeem filtert automatisch consultants die al toegang hebben tot de tenant, waardoor alleen relevante consultants in de lijst verschijnen.



12. deleteConsultantsFromTenant:

- **Doel:** Verwijdert consultants van een specifieke tenant.
- **Werking:** Controleert eerst of de tenant bestaat in het systeem. Als de tenant niet gevonden wordt, wordt een foutmelding gegenereerd. Vervolgens wordt gecontroleerd of de consultants bestaan. Consultants met volledige toegangsrechten worden gefilterd. Als alle opgegeven consultants volledige toegang hebben, wordt een conflictmelding gegenereerd. Daarna worden de consultants verwijderd van de tenant.

Onderstaande afbeelding illustreert de deleteConsultantsFromTenant functionaliteit waarbij een belangrijk onderscheid zichtbaar is tussen consultants met verschillende toegangsrechten. Consultants met volledige toegang tot alle tenants hebben een uitgeschakeld prullenbak-icoon en kunnen niet worden verwijderd van een specifieke tenant. Consultants die specifiek aan deze tenant zijn toegewezen hebben een actief prullenbak-icoon en kunnen wel worden verwijderd. Deze afbeelding toont de implementatie van de backend-logica in de frontend interface.

michael.voorhaen@strobbo.com	<input checked="" type="checkbox"/>	
moussa.ramzi@test.com	<input type="checkbox"/>	
moussa.ramzi@test1.com	<input checked="" type="checkbox"/>	

3.3. Login Logs

De Login Logs Module is een aanvullende module die ik tijdens mijn stage heb ontwikkeld om auditlogs bij te houden van gebruikersactiviteit, met een focus op logins binnen tenants. Deze module maakt het mogelijk om login- en consultatiegegevens op te halen per gebruiker of per tenant, inclusief filtering op datum en naam.

Net zoals de andere modules is deze opgebouwd volgens een vaste structuur, waarbij de volgende principes zijn toegepast:

- Controllers verwerken API-aanvragen en gebruiken DTO's om input te structureren en te valideren.
- DTO's worden gebruikt om consistente data-uitwisseling tussen de infrastructuur- en applicatielaag te garanderen.
- In-memory query repositories maken het mogelijk om snel functionaliteiten te testen zonder afhankelijkheid van externe opslag.
- Unit tests maken gebruik van deze in-memory implementaties om de werking van de logica betrouwbaar en herhaalbaar te valideren.

De module bestaat hoofdzakelijk uit leesoperaties en volgt daardoor volledig het query-gedeelte van CQRS. Alle logica wordt aangeroepen via een façadeklasse (TenantAuditLogsFacade), die op zijn beurt communiceert met de AuditLogQueryRepository.

3.3.1. Functies en implementaties

Onderstaande functies zijn gedefinieerd in de façade en vormen de kern van de module. De schermvoorbeelden illustreren hoe mijn backend-functionaliteiten werken via de bestaande frontend-interface (frontend niet door mij ontwikkeld).

1. findAuditLogs

- **Doel:** Haalt een gepagineerde lijst op van auditlogs, gefilterd op optionele parameters zoals tenantnaam, consultantnaam of datumrange.
- **Werking:** Gebruikt de onderliggende repository om auditlogs op te halen volgens de gespecificeerde filtercriteria.

Onderstaande afbeelding toont de praktische implementatie van de findAuditLogs functionaliteit. In dit overzicht zijn alle auditlogs zichtbaar met essentiële informatie zoals de consultant die heeft ingelogd, de specifieke tenant waartoe toegang werd verkregen, het tijdstip van de login, en de reden die de consultant heeft opgegeven voor het inloggen bij die tenant. Deze gepagineerde weergave illustreert hoe de ontwikkelde backend-logica de gefilterde auditlog-gegevens presenteert en zorgt voor volledige traceerbaarheid van consultantactiviteit binnen het systeem.

Created at	Tenant Name	Consultant Name	Tenant ID	Action	Subject	Additional Info
21/05/2025, 15:16:21	DEV_stage	moussa.ramzi@test1.com	34566F19-3461-4A48-8141-DADAD2EBBA18	signInRequested	Support	support hehe
21/05/2025, 12:28:11	DEV_McDoLeuven_Michael	info@onlinewerkrooster.be	B433D7B3-F6B6-79A2-4EB7-BB83AC58BE1A	signInRequested	Other	sdfgh
21/05/2025, 01:20:38	DEV_E2E_02	cypress.consultant@owr.be	E6581F6F-50A9-4B8E-93EF-C7CA17CD5B26	signInRequested		cypress
21/05/2025, 01:20:20	DEV_E2E_02	cypress.consultant@owr.be	E6581F6F-50A9-4B8E-93EF-C7CA17CD5B26	signInRequested	Other	cypress
20/05/2025, 11:18:36	DEV_DEVPOOL_BREAKOUT_1	info@onlinewerkrooster.be	D086EE2B-2999-4AAB-BB6B-CC63C684C1D9	signInRequested	Other	aaa
20/05/2025, 01:21:15	DEV_E2E_02	cypress.consultant@owr.be	E6581F6F-50A9-4B8E-93EF-C7CA17CD5B26	signInRequested		cypress
20/05/2025, 01:20:57	DEV_E2E_02	cypress.consultant@owr.be	E6581F6F-50A9-4B8E-93EF-C7CA17CD5B26	signInRequested	Other	cypress
19/05/2025, 14:40:30	DEV_stage	info@onlinewerkrooster.be	34566F19-3461-4A48-8141-DADAD2EBBA18	signInRequested	Other	sldfqdfq

2. getTotalCount

- **Doel:** Geeft het totaal aantal auditlogs terug zonder filters.
- **Werking:** Raadpleegt de repository om het totale aantal records te bepalen.


3. findAuditLogsByUserId

- **Doel:** Haalt een lijst van auditlogs op die gelinkt zijn aan een specifieke gebruiker.
- **Werking:** Filtert auditlogs op basis van gebruikers-ID, met optionele aanvullende filters zoals tenantnaam of datumbereik.

Onderstaande afbeelding toont de findAuditLogsByUserId functionaliteit in de praktijk, waarbij de auditlogs gefilterd worden voor een specifieke consultant. Op de detailpagina van de consultant zijn alle loginactiviteiten van deze gebruiker zichtbaar, inclusief de tenants waartoe toegang werd verkregen, de tijdstippen van de logins, en eventuele redenen die werden opgegeven. Deze gefilterde weergave maakt het mogelijk om snel inzicht te krijgen in de logingeschiedenis van een individuele consultant, wat essentieel is voor audit- en compliance-doeleinden.

Consultant details: moussa.ramzi@test1.comBack

Tenant accessRevoke all accessEdit

 moussa.ramzi@test1.com has access to all tenants

Logs

Filters

From

Until

Created at	Tenant Name	Tenant ID	Action	Subject	Additional Info
13/05/2025, 08:54:33	DEV_stage	34566F19-3461-4A48-8141-DADAD2EBBA18	signInRequested	Other	t
12/05/2025, 15:00:35	DEV_stage	34566F19-3461-4A48-8141-DADAD2EBBA18	signInRequested	Other	f
12/05/2025, 13:55:17	DEV_Barbet	D5FAC32C-BB62-4716-9B8A-B33C39B7B6AE	signInRequested	Other	qq
12/05/2025, 13:54:46	DEV_DEVPOOL_BREAKOUT_1	D086EE2B-2999-4AAB-BB6B-CC63C684C1D9	signInRequested	Other	qqq

4. getTotalCountByUserId

- **Doel:** Geeft het totaal aantal auditlogs terug dat gekoppeld is aan een specifieke gebruiker.
- **Werking:** Telt het aantal auditlog-records voor een specifieke gebruiker.

5. findAuditLogsByTenantId

- **Doel:** Haalt een lijst van auditlogs op die behoren tot een specifieke tenant.
- **Werking:** Filtert auditlogs op tenant-ID, met optionele aanvullende filters zoals consultantnaam en datumrange.

Onderstaande afbeelding illustreert de findAuditLogsByTenantId functionaliteit, waarbij het omgekeerde perspectief wordt getoond ten opzichte van de consultant-gebaseerde filtering. Vanuit het tenant-niveau worden alle auditlogs weergegeven die betrekking hebben op deze specifieke tenant. In dit overzicht is zichtbaar welke consultants toegang hebben gehad tot de tenant, wanneer deze logins plaatsvonden, en welke redenen daarbij werden opgegeven. Deze tenant-gerichte weergave biedt waardevolle inzichten in wie er toegang heeft gehad tot een specifieke klantomgeving en ondersteunt effectieve monitoring van tenant-activiteit.

Tenant DEV_stage (#404)

<< DEV_Sfen / DEV_StrobbMain02_E2E01 >>

Go back to list

Environment	Filters	From	Until	
Failed messages				
File storage				
Message board				
User profile				
User attachments				
Legal agreements				
Logs				
Consultants				
	Created at	Consultant Name	Subject	Additional Info
	15/05/2025, 09:30:43	info@onlinewerkrooster.be	Other	switch feature flag
	15/05/2025, 09:28:25	info@onlinewerkrooster.be	Other	sdf
	15/05/2025, 09:13:10	info@onlinewerkrooster.be	Other	wd
	15/05/2025, 08:31:14	moussa.ramzi@test1.com	Other	command menu
	15/05/2025, 08:27:22	moussa.ramzi@test1.com	Other	command menu

6. getTotalCountByTenantId

- **Doel:** Geeft het totaal aantal auditlogs terug dat gekoppeld is aan een specifieke tenant.
- **Werking:** Telt het aantal auditlog-records voor een specifieke tenant.

3.4. Dashboard

Om het onderdeel rond Consultant Management en Login Logs op een overzichtelijke en gebruiksvriendelijke manier af te sluiten, is er een dashboard ontwikkeld dat een visuele representatie biedt van verschillende gebruikstatistieken binnen het systeem. Dit dashboard werd ontworpen om snel inzicht te geven in loginpatronen, redenanalyses, en activiteit per tenant of consultant.

De backendfunctionaliteiten van dit dashboard werden volledig door mij ontwikkeld. De frontend werd verzorgd door een collega, en dient ter visualisatie van de data die via mijn backend wordt aangeleverd. De communicatie verloopt via een centrale façadeklasse (ReportingFacade), die op zijn beurt de ConsultantReportingQueryRepository aanspreekt om de benodigde data op te halen via geoptimaliseerde SQL-queries.

Net zoals de andere modules is ook deze opgedeeld volgens de CQRS-architectuur. De logica bestaat hoofdzakelijk uit leesoperaties, waarbij gebruik wordt gemaakt van DTO's en heldere structuren om data op een consistente en performante manier beschikbaar te maken.

3.4.1. Functies en implementaties

Onderstaande functies zijn gedefinieerd in de façade en vormen de kern van de dashboardmodule. De schermvoorbeelden illustreren hoe de backend-functionaliteiten in de frontend worden weergegeven (frontend niet door mij ontwikkeld).

1. `getAllConsultantData`

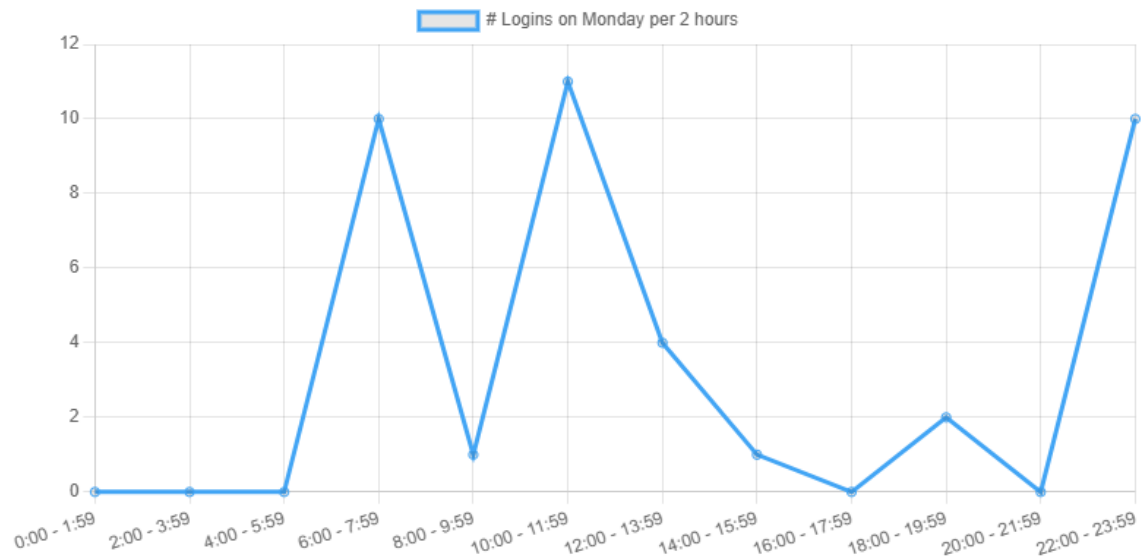
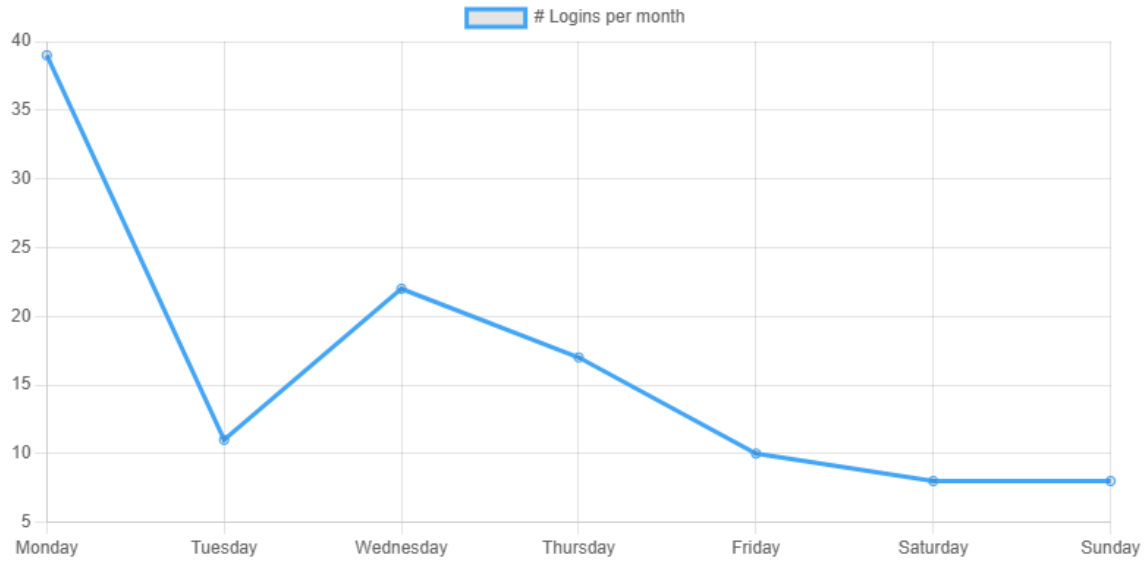
- **Doel:** Bundelt alle benodigde gegevens voor het dashboard in één centrale datastructuur.
- **Werking:** Roept intern verschillende querymethodes aan om loginstatistieken, loginredenen, tenantactiviteit, consultantactiviteit, redenlengtes en maandtotalen op te halen. Deze gegevens worden samengebracht in één overzichtelijk resultaat, zodat het dashboard alle nodige informatie in één keer ontvangt.

Dankzij deze aanpak hoeft de frontend slechts één enkele API-aanroep te doen om het volledige dashboard te kunnen vullen. Dit verhoogt de efficiëntie en vermindert de belasting op de server en netwerkcommunicatie.

2. transformLoginsPerWeekdayData

- **Doel:** Zet ruwe loginstatistieken om naar een structuur per weekday en per uur.
- **Werking:** Tel alle logins op per dag van de week, en per uur binnen die dag. Deze structuur wordt gebruikt voor de lijn-grafiek.

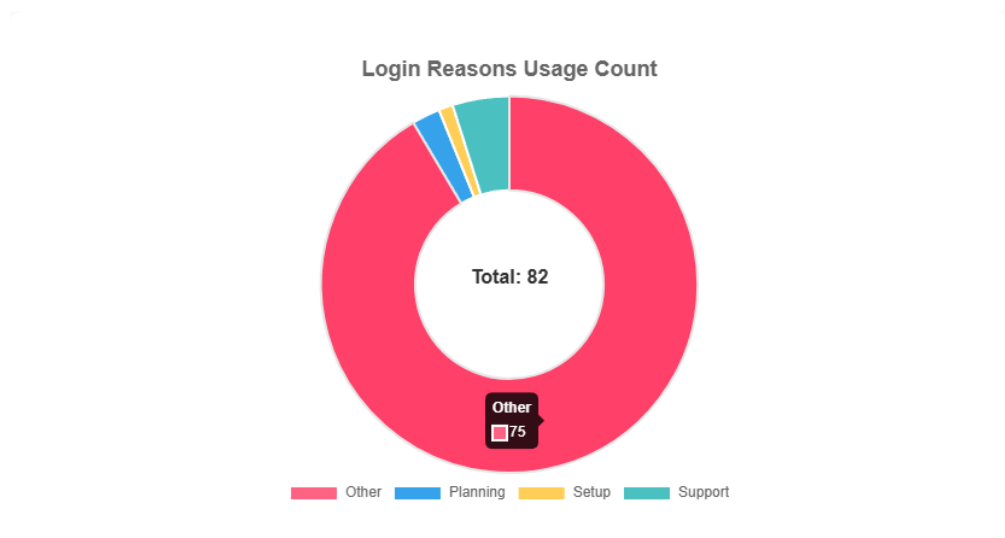
Onderstaande afbeelding toont de lijn-grafiek, die initieel de loginactiviteit per weekday toont. Wanneer een gebruiker op een dag klikt, zoomt de grafiek automatisch in en worden de logins per uur voor die specifieke dag weergegeven.



3. getConsultantLoginSubject

- **Doel:** Berekent hoe vaak bepaalde loginredenen werden opgegeven door consultants.
- **Werking:** Haalt alle "subjects" op uit loginacties en telt het aantal per type reden.

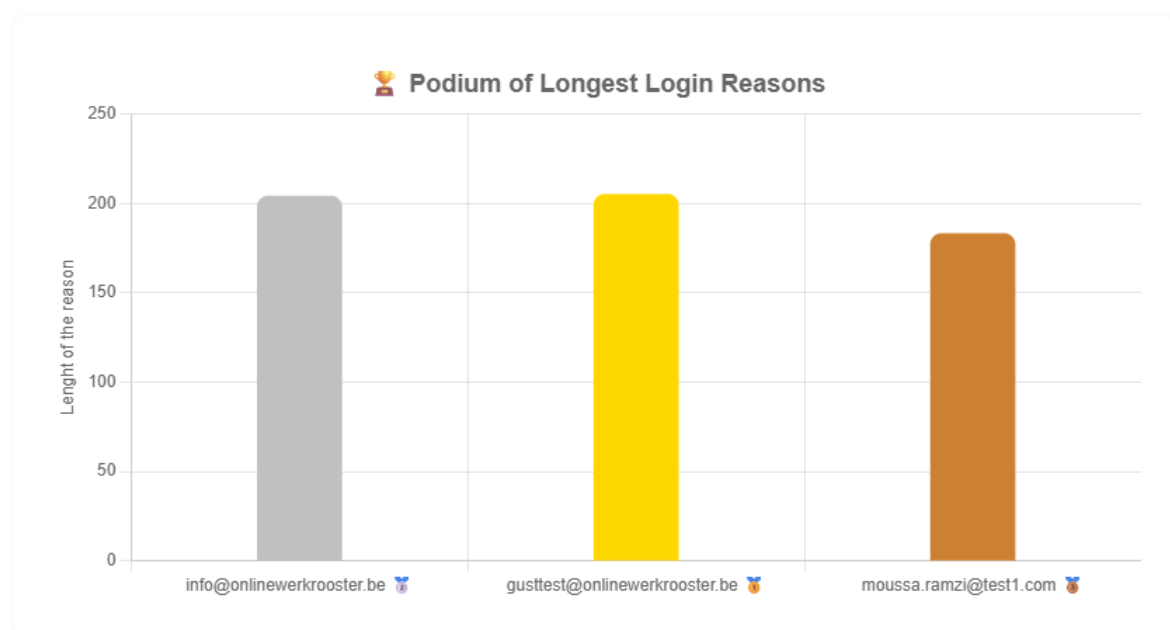
Onderstaande afbeelding toont de donutgrafiek, waarin duidelijk zichtbaar is welke redenen het vaakst zijn opgegeven bij het inloggen.



4. getLongestLoginReasons

- **Doel:** Selecteert de drie langste opgegeven loginredenen.
- **Werking:** Sorteert de loginredenen op lengte en geeft de top 3 terug.

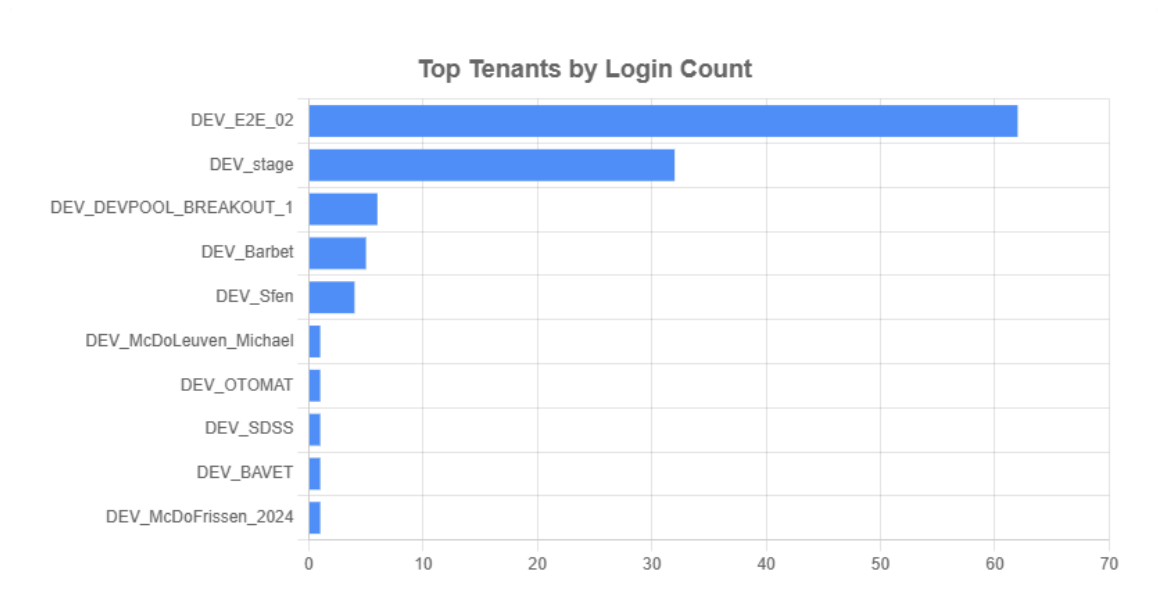
Onderstaande afbeelding toont een podiumvormige balkgrafiek, waarbij de consultant met de langste reden in het midden staat en de tweede en derde plaats links en rechts weergegeven worden.



5. getConsultantTenantLoginCounts

- **Doel:** Berekent per tenant hoeveel logins er in een bepaalde periode plaatsvonden.
- **Werking:** Groepeert logins op tenantniveau.

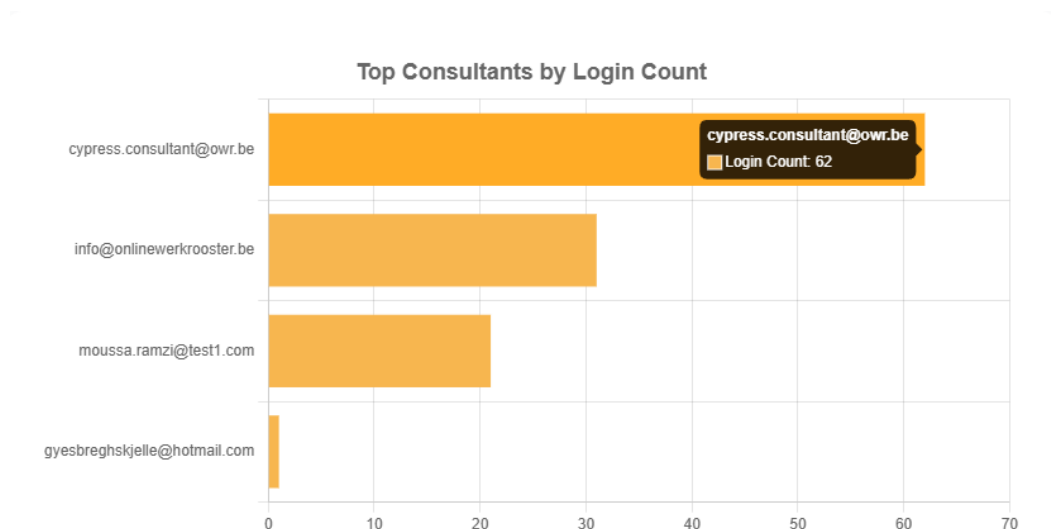
Onderstaande afbeelding toont een horizontale staafgrafiek, waarmee snel duidelijk wordt op welke tenants het meest wordt ingelogd.



6. getConsultantsLoginCount

- **Doel:** Haalt op hoeveel logins er per consultant zijn uitgevoerd.
- **Werking:** Groepeert de loginacties per consultant en sorteert op hoogste aantal.

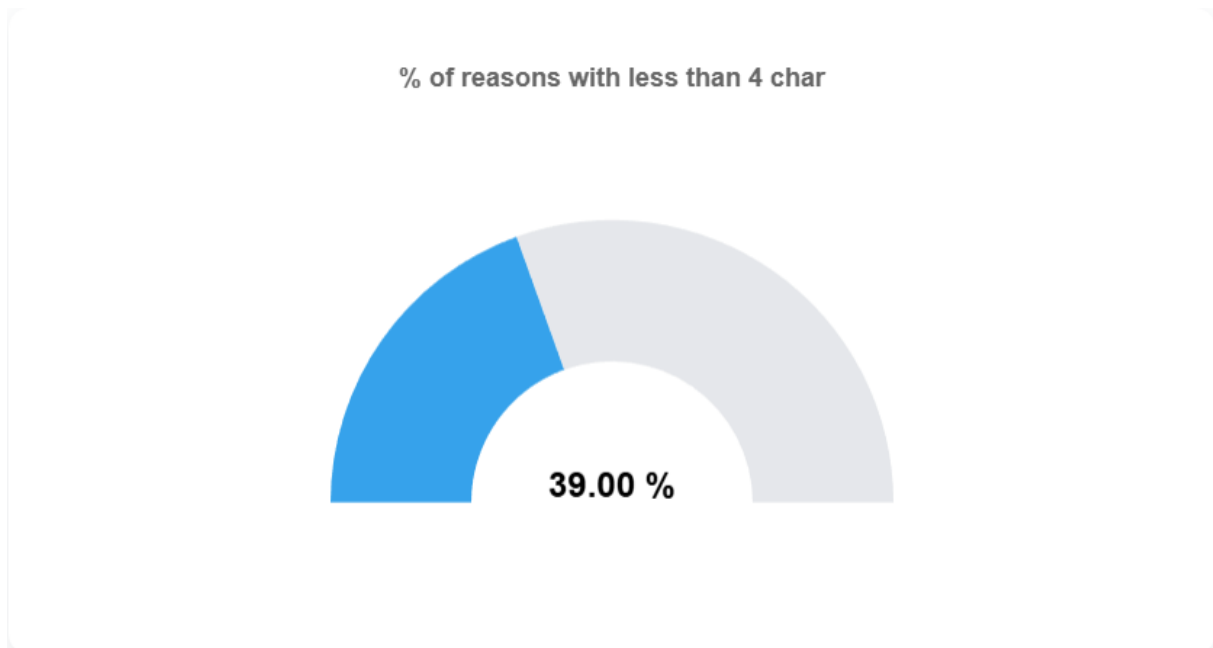
Onderstaande afbeelding toont een staafdiagram, waarmee zichtbaar wordt welke consultants het vaakst actief zijn binnen het systeem.



7. getReasonLengthStats

- **Doel:** Verzamelt statistieken over de lengte van opgegeven loginredenen.
- **Werking:** Telt hoeveel loginredenen korter of langer zijn dan een drempelwaarde (bv. 4 tekens) en berekent de percentages.

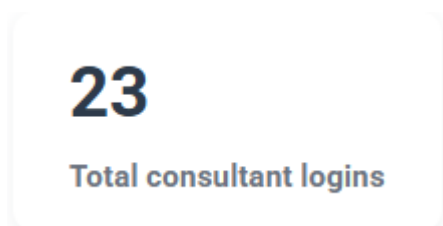
Onderstaande afbeelding toont een gauge-diagram, waarin het percentage van loginredenen met minder dan 4 tekens wordt weergegeven.



8. getCurrentMonthLoginCount

- **Doel:** Bepaalt het totale aantal loginacties dat in de huidige maand heeft plaatsgevonden.
- **Werking:** Telt alle loginrecords met een timestamp binnen de huidige maand.

Onderstaande afbeelding toont een KPI-tegel, waarin het totaal aantal logins van de maand zichtbaar is.



3.5. Audit logging

Binnen deze module heb ik gezorgd voor een uitbreiding van de audit logging waarbij de identiteit van de consultant die een login uitvoert expliciet wordt vastgelegd in het bearer token. Een bearer token is een soort toegangsbewijs dat bij elke aanvraag naar de server wordt meegestuurd en informatie bevat over de geauthenticeerde gebruiker. Door de identiteit van de consultant hierin op te nemen, kan het systeem achteraf exact traceren welke consultant verantwoordelijk was voor bepaalde acties, zelfs wanneer er gebruik wordt gemaakt van impersonatie.

UITBREIDING VAN HET BEARER TOKEN MET CONSULTANTINFORMATIE

Bij het aanmaken van een access token is de standaard payload uitgebreid met extra gegevens over de ingelogde consultant. Concreet is de e-mail van de consultant toegevoegd aan de tokenpayload via de issue methode van de TokenIssuer. Hierdoor bevat elk gegenereerd token niet alleen de gebruikersgegevens en rollen, maar ook de consultantinformatie die relevant is voor auditdoeleinden.

De tokengeneratie vindt plaats in de generateToken functie, waarbij, naast gebruikers- en tenantgegevens, optioneel ook de consultantinformatie wordt meegegeven en opgenomen in het JWT. Dit maakt het mogelijk dat de applicatie bij elke verzoek de identiteit van de consultant kan uitlezen en koppelen aan de uitgevoerde handelingen.

Dit is hoe de bearer token er uiteindelijk uit ziet:

```
{
  "data": {
    "user": {
      "id": 2,
      "email": "info@onlinewerkrooster.be",
      "language": "en",
      "roles": [
        "ROOT_ADMIN"
      ]
    },
    "tenant": {
      "id": 404,
      "name": "DEV_stage",
      "database": "DEV_stage",
      "employeeId": 1,
      "customerKey": 404,
      "companies": []
    },
    "strobboApiUrl": null,
    "consultantEmail": "moussa.ramzi@test1.com"
  },
  "iat": 1748537179,
  "exp": 1749746779
}
```

LOGGING BIJ LOGIN EN ACTIES


Tijdens het loginproces (signInToTenant) wordt gebruikgemaakt van bovenstaande tokengeneratie. Bij een succesvolle login worden meerdere auditlog-acties geregistreerd, zoals het aanvragen van login, het goedkeuren, en het daadwerkelijk inloggen. Alle logregels bevatten het userId en tenantId, waarbij dankzij de tokenuitbreiding ook de consultant die de login initieerde traceerbaar is.

ONDERSTEUNING VAN IMPERSONATIE MET BEHOUD VAN CONSULTANTINFORMATIE

Een belangrijke functionaliteit is impersonatie, waarbij een consultant zich tijdelijk voordoeft als een andere gebruiker. Ook bij impersonatie is het cruciaal om de originele consultant te kunnen identificeren. Dit is gerealiseerd door bij het genereren van een nieuw token tijdens impersonatie de consultant-e-mail uit het oorspronkelijke bearer token te extraheren en opnieuw mee te geven in de nieuwe tokenpayload.

Hierdoor blijft in het gehele traject – van login tot impersonatie en vervolgacties – de bron van de wijziging herkenbaar en transparant. Dit verhoogt de betrouwbaarheid van de audit logging en draagt bij aan een betere naleving van beveiligings- en compliance-eisen.

Dit is hoe de logging zichtbaar wordt aan de gebruikerszijde:

Actie	Gemaakt door
UPDATE	Admin Strobbo
UPDATE	Noah Van den Berg  (moussa.ramzi@test1.com)
UPDATE	Admin Strobbo (moussa.ramzi@test1.com)

3.6. Wisselen van Tenant

Voorheen moesten gebruikers volledig uitloggen en opnieuw inloggen om van tenant te kunnen wisselen. Dit zorgde voor een omslachtige gebruikerservaring, vooral voor gebruikers die regelmatig tussen meerdere tenants moeten schakelen. Tijdens mijn stage heb ik deze beperking opgelost door het introduceren van een naadloze tenant switch-functionaliteit. Hiermee kunnen gebruikers eenvoudig wisselen tussen tenants, zonder de applicatie te verlaten of hun sessie te verliezen.

Deze functionaliteit ondersteunt twee scenario's:

1. **Algemene tenant switch:** beschikbaar voor alle gebruikers met toegang tot meerdere tenants.
2. **Consultant tenant switch:** specifiek ontwikkeld voor consultants, waarbij hun identiteit behouden blijft tijdens de switch.

ALGEMENE TENANT SWITCH (VOOR ALLE GEBRUIKERS)

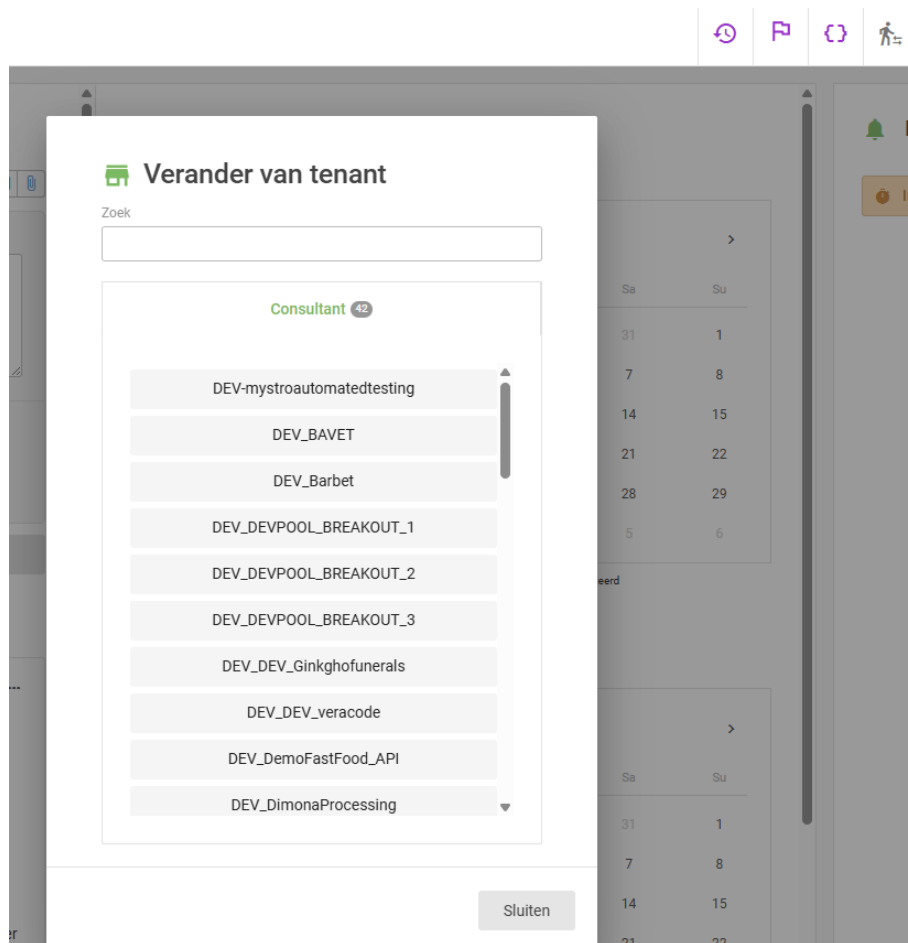
Voor reguliere gebruikers is de GET /tenants-endpoint aangepast zodat deze nu, naast de lijst van beschikbare tenants, ook _links bevat met sign-in instructies. Deze links maken het mogelijk om via de front-end direct naar een andere tenant te wisselen met één klik.

Elke tenant in de lijst bevat nu een signIn-link die:

- De juiste HTTP-methode (POST) bevat
- De route naar de sign-in endpoint specificeert
- Gebruikt kan worden door de front-end om automatisch een nieuwe sessie te starten in de geselecteerde tenant

Deze aanpak zorgt ervoor dat gebruikers snel kunnen wisselen van tenant zonder handmatig uit te loggen of hun inloggegevens opnieuw in te voeren.

Onderstaande afbeelding toont hoe de tenant switch-interface eruitziet voor eindgebruikers. De frontend werd ontwikkeld door mijn medestagiair, op basis van de backendfunctionaliteit die ik heb gerealiseerd.



CONSULTANT TENANT SWITCH (ALLEEN VOOR CONSULTANTS)

Voor consultants is een aparte functionaliteit geïmplementeerd om te wisselen tussen tenants, waarbij de oorspronkelijke consultantinformatie (zoals het e-mailadres) behouden blijft in de sessie.

De switch wordt uitgevoerd via een nieuwe endpoint. Hierin wordt de consultant-email uit het bestaande bearer token gehaald en gebruikt om de juiste consultant op te zoeken. Vervolgens wordt een nieuwe access token gegenereerd waarin diezelfde consultant-email opnieuw wordt opgenomen.

Deze aanpak is essentieel om ervoor te zorgen dat:

- De identiteit van de consultant consistent blijft tijdens de hele sessie
- Audit logs correct blijven verwijzen naar de consultant die de acties uitvoert, ook na het wisselen van tenant

Intern wordt hiervoor de switchTenant-methode aangeroepen, die automatisch een nieuwe login uitvoert in de gewenste tenant, zonder dat de gebruiker handmatig hoeft uit te loggen.

4. Besluit

Tijdens deze stage heb ik meegewerkt aan het uitbreiden van de interne tooling met als doel meer zicht te krijgen op de logins van consultants binnen verschillende tenants. Concreet werd een nieuwe module ontwikkeld waarmee nu kan worden nagegaan welke consultant wanneer, waar en waarom heeft ingelogd.

Daarnaast werd een belangrijke verbetering gerealiseerd op vlak van gebruiksvriendelijkheid: gebruikers en consultants kunnen nu van tenant wisselen zonder eerst uit te loggen, iets wat voorheen enkel mogelijk was via een omslachtige loginprocedure. Dit maakt het werken met meerdere tenants aanzienlijk efficiënter.

Om de verzamelde data op een toegankelijke en overzichtelijke manier te presenteren, werd er ook een dashboard toegevoegd. Dit dashboard biedt visuele inzichten in loginpatronen, redenanalyses en activiteit per tenant of consultant. De backend die deze grafieken voedt werd volledig door mij ontwikkeld en zorgt ervoor dat alle relevante informatie via één efficiënte API-call wordt opgevraagd. Hierdoor kunnen gebruikers sneller en doelgerichter werken met de beschikbare informatie.

Ook op het vlak van audit logging werd een grote stap vooruitgezet. Er is nu niet alleen zichtbaar waar en wanneer een consultant is ingelogd, maar ook welke wijzigingen hij of zij heeft uitgevoerd. Zelfs wanneer een consultant een andere gebruiker impersonate, blijft in de logging duidelijk wie de oorspronkelijke consultant was die de actie uitvoerde. Dit draagt sterk bij aan de transparantie en traceerbaarheid binnen het systeem.

De doelstellingen zoals vastgelegd in het projectplan zijn volledig behaald. De Consultant Management Module werd succesvol toegevoegd, de logging werd uitgebreid, de tenant switch-functionaliteit werd correct geïmplementeerd, en het dashboard werd toegevoegd als centrale plek om alle data inzichtelijk te maken.

Tot slot kijk ik met tevredenheid terug op het traject. Ik heb niet alleen functionele waarde toegevoegd aan de applicatie, maar ook veel bijgeleerd over het uitbreiden van bestaande systemen, het structureren van complexe functionaliteiten en het belang van goede logging, gebruikservaring en visualisatie. Deze ervaring neem ik mee als waardevolle stap in mijn professionele ontwikkeling.

LITERATUURLIJST

Geraadpleegde documentatie en bronnen (geschreven):

1. Knex.js documentation. <https://knexjs.org/>
2. Node.js documentation. <https://nodejs.org/en/docs/>
3. Swagger (OpenAPI) documentation. <https://swagger.io/specification/>
4. TypeScript documentation. <https://www.typescriptlang.org/docs/>
5. Microsoft SQL Server DATEPART documentation. <https://learn.microsoft.com/en-us/sql/t-sql/functions/datepart-transact-sql>
6. JWT (JSON Web Token) Introduction. <https://jwt.io/introduction>

Geraadpleegde personen (mondelinge bronnen):

1. Jinte Michiels en Martijn Bleyen, stagebegeleider – toelichting bij het CQRS-principe en opbouw van backendcomponenten.
1. Jinte Michiels en Martijn Bleyen, stagebegeleider – feedback en begeleiding bij de implementatie van login logging en tenant switching