FOBIS Platform Verkenning

None

FOBIS

None

Table of contents

1. FOBIS Next

Distributed service framework voor de volgende generatie van FOBIS.

Moderniseer je FOBIS-landschap stap voor stap, met behoud van betrouwbaarheid én ruimte voor nieuwe digitale diensten.

2. FOBIS Platform – Verkenning met FOBIS Next

Welkom bij de documentatiesite voor de **FOBIS Platform verkenning** en **FOBIS Next**.
Deze site biedt een overzicht van de visie, architectuur en technische uitwerking van het moderne, gedistribueerde serviceframework voor de volgende generatie van FOBIS-oplossingen.

2.1 Over FOBIS en FOBIS Next

FOBIS is een product van RBK Group dat bij klanten on-premise draait en vooral wordt ingezet door productiebedrijven in de levensmiddelenindustrie (vleesverwerkers, bakkerijen, etc.). Het ondersteunt kritische productieprocessen: planning, uitvoering en registratie, met MES/ERP-achtige functionaliteit dicht op de fabriek.

FOBIS Next is het antwoord op de veranderende eisen aan IT-landschappen. Het is geen compleet nieuw systeem, maar een **distributed service framework** dat:

- **FOBIS centraal houdt** als bron van waarheid waar dat nodig is
- **Nieuwe diensten en componenten** op een moderne manier ontsluit en met FOBIS verbindt
- **Gefaseerde migratie** mogelijk maakt, zodat organisaties niet in één keer hoeven te "big-bangen"
- Een **brug vormt** tussen het bewezen bestaande en het innovatieve nieuwe – met behoud van stabiliteit én ruimte voor vernieuwing

Het platform is ontworpen als **"SaaS-ready, on-prem capable"**: de architectuur, componenten en processen zijn geschikt voor een echte SaaS-uitrol, maar kunnen tegelijkertijd ook on-premise of hybride worden ingezet.

2.2 Waar te beginnen?

Afhankelijk van je rol en interesse:

- **Ben je nieuw bij FOBIS Next?**
  Start met de Visie en framing en het Merkverhaal om de context en positionering te begrijpen.
- **Ben je architect of technisch ontwerper?**
  Begin met de Doelarchitectuur en verdiep je vervolgens in specifieke onderdelen zoals Authenticatie en Autorisatie of Logging en Observability.
- **Wil je het huidige FOBIS-landschap begrijpen?**
  Bekijk de Architectuur - Now sectie voor een overzicht van de bestaande componenten en technische uitwerking.
- **Ben je geïnteresseerd in deployment en implementatie?**
  Lees de Deploymentmodellen voor verschillende inzetscenario's.

2.3 Wat vind je op deze site?

Deze documentatiesite is opgedeeld in drie hoofdgebieden:

Verkenning

Visie, doelarchitectuur, deploymentmodellen, logging & observability, licensing en meer. Hier vind je de strategische en technische verkenning van het FOBIS Platform, inclusief inspiratiedocumenten en stakeholdercommunicatie.

Architectuur - Now

Technische uitwerking van het huidige FOBIS-landschap en de eerste schetsen voor de toekomst. Overzichten van componenten zoals Libraries, Pipelines, Utilities, de Verkoop-app en Customs.

Marketing & verhaal

Positionering, merkverhaal en communicatiemateriaal rond FOBIS Next. Hier vind je de kernboodschap, elevator pitch en belangrijkste merkwaarden.

2.4 Belangrijkste concepten

Het FOBIS Platform is gebouwd rond een aantal kernconcepten:

- **Control plane vs. Data plane**
  Scheiding tussen centrale diensten (authenticatie, licensering, logging) en datastromen dicht bij de klant (FOBIS Core, shopfloor-integraties).
- **SaaS-ready, on-prem capable**
  Architectuur die zowel geschikt is voor multi-tenant SaaS-uitrol als voor on-premise of hybride deployment.
- **Gefaseerde migratie**
  Geen "big bang"-aanpak, maar stap-voor-stap modernisering met behoud van stabiliteit en continuïteit.
- **Serviceframework**
  Consistent raamwerk voor nieuwe en bestaande functionaliteit, met duidelijke integratiepunten en standaardpatronen voor security, logging en deployment.
- **Multi-tenant design**
  Zelfs in single-tenant on-prem installaties wordt multi-tenant gedacht in design (tenant-id in data/logging/security), wat toekomstige SaaS-uitrol vergemakkelijkt.

Gebruik het menu aan de linkerkant om door de verschillende onderdelen te navigeren.

2.5 Hoe deze site lokaal te draaien is (MkDocs)

1. **Installeer MkDocs (bij voorkeur in een virtualenv of Conda-omgeving):**

```
pip install mkdocs mkdocs-material
```

1. **Start de ontwikkelserver vanuit de projectroot:**

```
mkdocs serve
```

1. Open in je browser: `http://127.0.0.1:8000/`
2. Om een **statische site te bouwen**:

   ```
   mkdocs build
   ```

   De HTML-bestanden komen dan in de map `site/` te staan.

3. Verkenning

3.1 Platform

3.1.1 Inspiratie

Fobis Platform – Inspiratiedocument (eerste chat)

Je opdracht is eigenlijk: **zorg dat er begin januari een helder kader ligt** waar Henk en Bob meteen in kunnen stappen (architectuur, werkwijze, scope, eerste backlog). Dit document vat de eerste verkenning samen en kan als inspiratie en basis dienen.

---

1. Overzicht: twee sporen

We werken met **twee sporen**:

- **Fobis Platform – kader & architectuur**
- **Drive App – visie, scope & eerste backlog**

Doel: dat er een **concreet pakket aan kaders en documenten** ligt voordat Henk en Bob van 4DotNet aanhaken.

---

2. Fobis Platform – Kader & Architectuur

2.1 DOEL & SCOPE

- **Doel platform**:
- **Fobis moderniseren** zodat nieuwe (cloud) apps snel, veilig en herhaalbaar gebouwd kunnen worden.
- **Partners** (zoals 4DotNet en externe partijen) in staat stellen om zelfstandig op het platform te ontwikkelen binnen duidelijke kaders.
- **Scope eerste fase**:
- **Basisplatform** + **eerste app (Drive App)** als “referentie-implementatie”.
- Focus op **authenticatie, autorisatie, API-laag en integratie met bestaand Fobis**.
- **Out-of-scope (nu)**:
- Volledige herbouw van bestaande Fobis-functionaliteit.
- Complexe analytics/BI-oplossingen (alleen basis-logging en -metrics).

2.2 PRINCIPES & RICHTLIJNEN

Formuleer een korte set platformprincipes, bijvoorbeeld:

- **Cloud-first**: nieuwe oplossingen draaien primair in de cloud (bijv. Azure/AWS – invullen afhankelijk van jullie keuze).
- **API-first**: alle functionaliteit wordt via goed ontworpen API’s ontsloten.
- **Security-by-design**: identity, autorisatie, logging en audittrail zijn basis, geen nice-to-have.
- **Re-use over rebuild**: waar mogelijk bestaande Fobis-logica hergebruiken via integraties.
- **Configurabel, niet customizebaar**: voorkeur voor configuratie boven maatwerk.

2.3 ARCHITECTUUR (HIGH-LEVEL)

- **Doelarchitectuur**:
- **Backend**: bijvoorbeeld een set services (modulaire monoliet of microservices) met een duidelijke API-laag.
- **Frontend/apps**: losse apps (web/mobile) die via de API-laag praten met Fobis Platform.
- **Integratielaag met bestaand Fobis**: queues, API’s of database-integratie (afhankelijk van huidige situatie).
- **Technologiestack (voorstel)**:

- **Cloud**: Azure/AWS? (hier jouw voorkeur/organisatie-standaard invullen).
- **Backend**: bijv. .NET 8 + REST/GraphQL API's.
- **Frontend**: bijv. React/React Native of Flutter (voor Drive App).
- **Database**: relationeel (bijv. SQL) + eventueel document store (bijv. CosmosDB) voor specifieke use cases.
- **Integraties**:
- Hoe praat Fobis Platform met het bestaande Fobis-systeem?
- Welke data wordt leidend waar? (master data vs transactie data).

2.4 SECURITY & IDENTITY

- **Identity provider**:
- Bepaal of jullie **Azure AD / Entra ID** of een andere IdP gebruiken.
- **Gebruikers en rollen**:
- Chauffeur, planner, beheerder, partner-ontwikkelaar, etc.
- **Autorisatie model**:
- Role-based of claims-based, hoe wordt dit centraal geregeld?
- **Compliance & privacy**:
- AVG: welke persoonsgegevens in Drive App? Bewaartermijnen, logging, audittrail.

2.5 KWALITEIT, DEVOPS & WERKWIJZE

- **Development proces**:
- Scrum/Kanban, sprints, refinement, retro's.
- **Branching & code review**:
- Git-flow/Trunk-based, pull requests, code review regels.
- **CI/CD**:
- Minimale eisen: automatische build, tests, quality checks, deployment naar test/acceptatie/ productie.
- **Quality gates**:
- Unit/integration tests, code coverage, static code analysis, performance-sanity checks.

---

3. Drive App – Visie, Scope & Eerste Backlog

3.1 VISIE & DOELSTELLINGEN

- **Visie**:
- De Drive App is de primaire tool voor chauffeurs om **ritten te zien en leveringen af te melden**, met een gebruikservaring vergelijkbaar of beter dan Distririjden.
- **Businessdoelen**:
- Minder fouten in afleverregistratie.
- Snellere verwerking in Fobis (real-time waar mogelijk).
- Minder papier/telefoontjes met planning.

3.2 BELANGRIJKSTE GEBRUIKERSFLOWS

Beschrijf per flow kort de stappen (later uitwerken met Henk & Bob):

- **Flow 1 – Inloggen chauffeur**
- **Flow 2 – Ritselectie / overzicht ritten van vandaag**
- **Flow 3 – Overzicht stops/leveringen binnen een rit**
- **Flow 4 – Levering afmelden (geslaagd / niet-geslaagd + reden)**
- **Flow 5 – Foto's / handtekening toevoegen (optioneel)**
- **Flow 6 – Offline gebruik en synchronisatie** (als relevant in jullie context)

3.3 MVP SCOPE VS LATERE UITBREIDINGEN

• **MVP (voorstel)**:
• Inloggen chauffeur.
• Rittenoverzicht (dagniveau).
• Leveringen per rit tonen (basisgegevens).
• Levering afmelden met status (geleverd / niet geleverd) en standaardredenen.
• Basis logging / monitoring.
• **Post-MVP ideeën**:
• Foto's van leverbon/omstandigheden.
• Digitale handtekening klant.
• Navigatie-integratie (Google Maps/Waze).
• Push-notificaties aan chauffeurs.

3.4 EERSTE USER STORIES (VOORBEELD)

Je kunt alvast een eerste backlog-skelet maken, bijvoorbeeld:

• **Als chauffeur** wil ik kunnen **inloggen** met mijn bedrijfsaccount zodat alleen ik bij mijn ritten kan.
• **Als chauffeur** wil ik een **overzicht zien van mijn ritten van vandaag** zodat ik weet welke ritten ik moet rijden.
• **Als chauffeur** wil ik per rit **alle stops met basisinformatie** zien zodat ik weet waar ik moet leveren.
• **Als chauffeur** wil ik een levering **als 'geleverd' kunnen markeren** zodat de planning realtime inzicht heeft.
• **Als chauffeur** wil ik een levering **als 'niet geleverd' met reden kunnen markeren** zodat de planning weet wat er misging.
• **Als planner** wil ik **in Fobis de actuele afleverstatus** zien zodat ik klanten kan informeren.

(Hier kun je later met 4DotNet dieper op ingaan met acceptatiecriteria en technische tasks.)

3.5 RANDVOORWAARDEN & NON-FUNCTIONALS

• **Performance**: app moet soepel werken op gemiddelde Android-toestellen van chauffeurs.
• **Offline**: minimaal "graceful" om kunnen gaan met tijdelijk geen netwerk (opslaan in queue, later sync).
• **Beveiliging**: toestelverlies, sessietimeout, minimaliseren van data op device.
• **Logging & monitoring**: per rit en per chauffeur kunnen zien wat er is gebeurd.

---

4. Roadmap tot januari (voorstel)

Wat jij de komende weken kunt doen, zodat Henk & Bob goed kunnen starten:

• **Nov–december**:
• **Platformkader-document** uitwerken met: doel, principes, high-level architectuur, security, werkwijze.
• **Drive App-inceptiedocument**: visie, flows, MVP-scope, eerste user stories.
• Inventarisatie huidige Fobis-landschap: welke data, welke interfaces, waar zitten de "haakjes".
• **Begin januari**:
• Gezamenlijke **inception/workshop** met 4DotNet op basis van jouw documenten.
• Architectuurschets aanscherpen + technische spikes bepalen.
• MVP-backlog finaliseren en starten met implementatie.

---

5. Mogelijke vervolgstappen

Mogelijke vervolgstappen op basis van dit inspiratiedocument:

- Een **formeler “Fobis Platform Kader”-document** maken met hoofdstukindeling volgens jullie standaard.
- Een **Drive App-inceptiedocument** met meer detail (user journeys, schermschetsen, acceptatiecriteria).
- Samen met Henk en Bob deze kaders reviewen en aanscherpen tijdens een kick-off/inception begin januari.

3.1.2 Visie en framing

Fobis Platform – Visie en framing

1. FOBIS VANDAAG

- **Fobis** is een product van RBK Group dat bij klanten **on-premise** draait.
- Doelgroep: vooral productiebedrijven in de **levensmiddelenindustrie** (vleesverwerkers, bakkerijen, etc.).
- Rol van Fobis:
- Ondersteunen van **productieprocessen**: planning, uitvoering, registratie.
- MES/ERP-achtige functionaliteit dicht op de fabriek.
- Belangrijke kenmerken van de huidige situatie:
- Veel **gespreide installaties** bij klanten, elk met eigen versie en configuratie.
- Technische onderdelen zoals **mailservice, licenseserver, logging** zitten nog vrij dicht op de applicatie zelf.
- **Authenticatie, autorisatie en licensering** zijn historisch gegroeid en op verschillende manieren geïmplementeerd.

2. WAAROM EEN FOBIS PLATFORM?

De markt en klantverwachtingen veranderen:

- Klanten willen steeds vaker **moderne, cloudgebaseerde oplossingen**:
- Web- en mobiele applicaties (zoals een Drive App voor chauffeurs).
- Real-time inzicht, integraties met andere systemen (Exact, Odoo, etc.).
- Partners (zoals 4DotNet) moeten op een **voorspelbare manier** kunnen ontwikkelen:
- Heldere API's en events.
- Standaard patronen voor security, logging, deployment.

Daarom is het nodig om Fobis te ontwikkelen van een **monolithisch product** naar een **platform**:

- Een **open integratie- en applicatiebasis** waar Fobis Core, nieuwe apps en partneroplossingen bovenop draaien.
- Een platform dat **Fobis ontsluit**: data, processen en services die nu "binnen" het product zitten, beschikbaar maken via moderne interfaces.

3. VISIE: SAAS-READY, ON-PREM CAPABLE

De kern van de visie:

- Fobis Platform wordt ontworpen als **"SaaS-ready, on-prem capable"**:
- Architectuur, componenten en processen zijn geschikt voor een echte **SaaS-uitrol** (multi-tenant, centraal beheer, observability).
- Tegelijkertijd moet hetzelfde platform **ook on-premise of hybride** inzetbaar zijn.
- In plaats van een éénmalige SaaS-sprong, bouwen we **een platform** dat:
- Vandaag al **waarde levert** aan on-prem klanten.
- Morgen zonder grote herbouw kan opschalen naar **SaaS en industry-platform**.

Concreet betekent dit:

- **12-factor-achtige services** die configuratie uit omgeving/secrets halen en niet vastzitten aan één datacenter.
- **Multi-tenant** denken in design (tenant-id in data/logging/security), ook als een on-prem installatie single-tenant is.
- Heldere **scheiding tussen control plane en data plane**:
- Control plane: zaken als authenticatie, licensering, logging-aggregatie.
- Data plane: koppeling met machines, shopfloor, klant-specifieke data.

4. AUTH, AUTORISATIE EN LICENSERING LOSWEKEN EN STANDAARDISEREN

Vandaag:

- Authenticatie, autorisatie en licensering zijn op verschillende manieren in Fobis ingebouwd.
- Dit maakt:
- Uitbreiding naar nieuwe apps (zoals Drive App) complex.
- Integratie met externe identity-oplossingen moeilijker.
- Beheer en support lastiger.

Toekomstbeeld binnen Fobis Platform:

- **Authenticatie**:
- Gebaseerd op **OpenID Connect (OIDC)** en **OAuth2** met een moderne identity provider (bijv. Entra ID, Keycloak, Auth0, IdentityServer).
- Ondersteuning voor klant-specifieke identity (bijv. on-prem AD) via federatie.
- **Autorisatie**:
- **Claims-based** autorisatie met rollen en rechten die in tokens en een centraal permission model zijn vastgelegd.
- Geen hardcoded permissies per applicatie, maar een gedeeld model dat door alle Fobis Platform-apps wordt gebruikt.
- **Licensering**:
- Een centrale **licensing-service** die licenties per klant/tenant, per module/feature en per omgeving vastlegt.
- Apps raadplegen de licensing-service om te bepalen **welke functionaliteit beschikbaar is**.

Dit maakt Fobis:

- **Open en uitbreidbaar**: nieuwe apps en partners kunnen op dezelfde manier aanhaken.
- **Consistent**: gebruikerservaring en security-gedrag zijn herkenbaar over alle applicaties heen.
- **Toekomstbestendig**: makkelijker om aan te sluiten op industrie-standaarden en ecosystemen.

5. LOGGING EN OBSERVABILITY ALS EERSTE KLAS ONDERDEEL

Vandaag:

- Logging via **Serilog**, naar bestand en/of SQL-database.
- Prima basis, maar beperkt voor:
- Centrale monitoring over veel installaties heen.
- Proactief detecteren van problemen.
- Geavanceerde (AI-)analyses op klantomgevingen.

Visie:

- Serilog blijft, maar wordt onderdeel van een breder **observability-verhaal**:
- Gestructureerde logging (JSON) met o.a. tenant-id, correlation-id, severity, event type.
- Gebruik van **pluggable sinks**: file/SQL voor pure on-prem, plus een sink naar een centrale log-provider (Seq, Elastic, Application Insights, ...).
- Basis-metrics (health, errors, throughput, performance) en tracing worden standaard.
- Doel:
- **Proactieve serviceorganisatie**: RBK kan problemen signaleren voordat de klant belt.
- Mogelijkheid tot **AI-ondersteunde analyses** op logdata en metrics.

6. OMGAAN MET DE ANGST VOOR SAAS BINNEN RBK

Signalen uit de organisatie:

- Er bestaat een **angst voor SaaS**:
- Men vreest dat **beheersbaarheid** van een echte SaaS-oplossing erg ingewikkeld is.
- Men wil oplossingen kunnen blijven leveren die **on-premise** of **hybride** inzetbaar zijn.

Kernboodschap van deze visie:

- We bouwen **geen** rigide SaaS-only oplossing.
- We bouwen een **platform** dat:
- **SaaS-ready** is (zodat we als RBK commercieel kunnen opschalen en internationaal kunnen groeien).
- Tegelijkertijd op een **controleerbare, voorspelbare manier on-prem/hybride** uit te rollen is.

Belangrijke argumenten:

- De huidige situatie met vele gespreide on-prem installaties is óók complex in beheer:
- Versieverspreiding, maatwerk, verschillende configuraties per klant.
- Moeilijk centraal inzicht in gezondheid en gebruik.
- Een moderne, gestandaardiseerde platformarchitectuur maakt beheer **uiteindelijk makkelijker**:
- Eenduidige deploymentmodellen.
- Centraal gedefinieerde beveiliging, logging en monitoring.
- Tools voor provisioning, updates, incident-response.

7. COMMERCIËLE AMBITIE: FOBIS ALS INDUSTRY-PLATFORM

De gewenste richting:

- Fobis wordt niet alleen “nog een pakket” dat moet koppelen met systemen als Exact of Odoo.
- Fobis Platform wordt een **industry-standaard** integratiepunt:
- Andere partijen willen juist **koppelen met Fobis**.
- Denk aan de vraag: “Kunnen we een koppeling maken met **Fobis Next**?” in plaats van andersom.

Randvoorwaarden om dit te bereiken:

- **Open, goed gedocumenteerde API’s** en event-interfaces.
- Heldere security- en licentie-architectuur.
- Stabiele, voorspelbare deploymentmodellen (zowel on-prem, klant-cloud als RBK SaaS).

8. SAMENVATTING IN ÉÉN ZIN

Fobis Platform is de evolutie van Fobis van een on-prem product naar een **modern, open en SaaS-ready platform** dat tegelijkertijd **on-prem en hybride** inzetbaar blijft, en waarmee RBK de basis legt om een **industry-standaard** in de foodproductie te worden.

3.1.3 Doelarchitectuur

```
flowchart LR
  subgraph "Klantomgeving"
    FC["Fobis Core"]
    SF["Shopfloor-systemen / machines / lokale databases"]
  end

  subgraph "Fobis Platform Services (control plane)"
    APIGW["API Gateway / API Layer"]
    AUTH["Auth & Identity integratie (IdP)"]
    AUTHZ["Authorization Service"]
    LIC["Licensing Service"]
    INTEG["Integration Services"]
    LOG["Logging & Observability"]
  end

  subgraph "Applicaties & extensies"
    DRIVE["Drive App"]
    OTHER["Andere RBK- en partnerapps"]
  end

  subgraph "Externe systemen"
    ERP["ERP / financieel (Exact, Odoo, ...)"]
    CLOUD["Overige cloudservices (BI, planning, analytics)"]
  end

  DRIVE --> APIGW
  OTHER --> APIGW

  APIGW --> AUTH
  APIGW --> AUTHZ
  APIGW --> LIC
  APIGW --> INTEG
  APIGW --> LOG

  INTEG <--> FC
  FC <--> SF

  INTEG <--> ERP
  INTEG <--> CLOUD

  DRIVE -.-> LOG
  OTHER -.-> LOG
  FC -.-> LOG
  INTEG -.-> LOG
```

Fobis Platform – Doelarchitectuur (hoog niveau)

1. DOEL VAN DEZE ARCHITECTUURSCHETS

Deze schets beschrijft de **doelarchitectuur** van Fobis Platform op hoofdlijnen:

- Hoe **Fobis Core** (bestaand product) zich verhoudt tot het nieuwe platform.
- Welke **platformservices** er zijn (auth, licensing, logging, integratie).
- Hoe **apps** zoals de Drive App daarop aansluiten.
- Hoe dit alles zowel **SaaS-ready als on-prem/hybride** inzetbaar is.

Het is een **richtinggevend** document, geen detailontwerp.

2. HOOFDCOMPONENTEN (LOGISCHE ARCHITECTUUR)

Op hoog niveau onderscheiden we vier zones:

1. **Klantomgeving**
2. Fobis Core (bestaande on-prem installatie).
3. Shopfloor-systemen, machines, lokale databases.
4. **Fobis Platform Services**
5. Auth & Identity integratie.
6. Licensing-service.
7. API Gateway / API Layer.
8. Integratieservices (sync, messaging, ETL).
9. Logging & observability.
10. **Applicaties en extensies**
11. Drive App (mobiel / web).
12. Andere RBK-apps (webportalen, dashboards).
13. Partner- en klant-specifieke apps.
14. **Externe systemen**
15. ERP/financiële pakketten (bijv. Exact, Odoo, …).
16. Andere cloudservices (BI, analytics, planning, etc.).

---

3. SCHEIDING TUSSEN CONTROL PLANE EN DATA PLANE

Een belangrijk ontwerpprincipe is de scheiding tussen:

• **Control plane**:

• Centrale diensten zoals authenticatie, autorisatie, licensering, configuratie, logging-aggregatie, monitoring.

• Kan in RBK-cloud, klant-cloud of on-prem draaien (afhankelijk van deploymentmodel), maar is logisch “centrale” laag.

• **Data plane**:

• De datastromen en processen dicht bij de klant: Fobis Core, shopfloor-integraties, lokale databases, message queues.

• Blijft waar nodig **dicht bij de fabriek** (latency, connectiviteit, regelgeving).

Nieuwe apps (zoals Drive App) praten primair met de **control plane** (API’s, auth, licensing) en indirect met de **data plane** via platformservices.

---

4. COMPONENTEN BINNEN FOBIS PLATFORM SERVICES

Binnen de platformlaag onderscheiden we de volgende logische services:

1. **API Gateway / API Layer**
2. Enig toegangspunt voor externe clients (apps, partners).
3. Verzorgt:
   - Routing naar interne services.
   - Authenticatiecontrole (verificatie van tokens).
   - Basic rate limiting / throttling.
   - Versiebeheer van API's.
4. **Auth & Identity Integratie**
5. Sluit aan op een externe of interne **identity provider** (bijv. Entra ID, Keycloak, …) via OpenID Connect.
6. Zorgt voor:
   - Single sign-on voor gebruikers.
   - Mapping van externe identiteiten naar Fobis-rollen/tenants.
7. **Authorization Service**
8. Beheert **rollen, rechten en policies** binnen Fobis Platform.
9. Biedt API's om te bepalen of een gebruiker/klant een bepaalde actie mag uitvoeren.
10. **Licensing Service**
11. Legt vast welke **modules/features** per klant/tenant actief zijn.
12. Biedt API's waarmee apps kunnen controleren of bepaalde functionaliteit beschikbaar is.
13. Ondersteunt zowel SaaS als on-prem scenario's (met caching en offline grace periods).
14. **Integration Services**
15. Koppelen Fobis Platform met:
    - Fobis Core on-prem (bijv. via API's, message queues, database sync).
    - Externe systemen (ERP's, planning, BI).
16. Typische functies:
    - Data-synchronisatie (masterdata, orders, productiegegevens).
    - Eventdistributie (bijv. "levering afgemeld" naar ERP).
17. **Logging & Observability**
18. Centrale logs, metrics en tracing voor alle platformservices en apps.
19. Standaardiseert:
    - Logformaat en -niveau.
    - Correlation-id over keten heen.
    - Tenant- en klantcontext in logs.

---

5. FOBIS CORE IN RELATIE TOT HET PLATFORM

Fobis Core blijft het systeem dat:

- Dicht bij de **productieomgeving** van de klant draait (on-prem of in klant-cloud).
- Verantwoordelijk is voor:
- Productielogica.
- Nauw klant-specifieke configuratie.
- Integratie met machines en lokale systemen.

In de doelarchitectuur:

- Worden **niet-versiegebonden componenten** (mail, licensering, logging, auth) geleidelijk losgehaald naar Fobis Platform.
- Krijgt Fobis Core een **duidelijke integratielaag**:
- Gestandaardiseerde API's of events richting Fobis Platform.
- Minder directe koppelingen richting externe systemen; die verschuiven naar de platformlaag.

Dit creëert ruimte om:

- Nieuwe functionaliteit **buiten** Fobis Core te bouwen (bijv. Drive App).
- Fobis Core stapsgewijs te moderniseren zonder alles tegelijk te moeten herschrijven.

---

6. APPS BOVENOP HET PLATFORM (BIJV. DRIVE APP)

Apps zoals de **Drive App** gebruiken Fobis Platform als fundament:

- **Authenticatie & autorisatie**:
- App laat gebruikers (bijv. chauffeurs) inloggen via de gekozen IdP.
- De app ontvangt tokens met claims over gebruiker, rol, klant, tenant.
- **Businessfunctionaliteit**:
- App praat met de API Gateway van Fobis Platform (bijv. `/api/drive/ritten`, `/api/drive/afmelden`).
- Platformservices vertalen dit naar acties richting Fobis Core (bijv. via integratieservices).
- **Licensering**:
- Voor toegang tot Drive App-functionaliteit controleert de app / API of de klant hiervoor een geldige licentie heeft.
- **Logging**:
- Acties in de app worden gelogd via de platformservices, zodat RBK en klant inzicht hebben in gebruik en problemen.

Zo wordt de Drive App een **referentie-implementatie** van hoe andere toekomstige apps met Fobis Platform communiceren.

---

7. MULTI-TENANT DENKEN

Hoewel Fobis vandaag vaak **per klant geïnstalleerd** wordt, is het belangrijk dat Fobis Platform **multi-tenant** kan werken:

- Elke klant / installatie wordt gezien als een **tenant**.
- In alle lagen wordt de **tenant-id** meegenomen:
- In gebruikersidentiteiten en claims.
- In licenties.
- In data en logregels.

Dit maakt het mogelijk om:

- In een SaaS-scenario meerdere klanten op één platformomgeving te draaien.
- Ook in on-prem/hybride scenario's dezelfde code en patronen te gebruiken, maar dan vaak met één tenant per omgeving.

---

8. DEPLOYMENT OP HOOFDLIJNEN

De doelarchitectuur moet drie hoofdmodellen ondersteunen (later verder uitgewerkt in een apart deploymentdocument):

1. **Volledig on-premise bij klant**
2. Fobis Core en een (gedeeltelijke) set platformservices draaien in de klantomgeving.
3. Optioneel: verbinding naar RBK-cloud voor licentievalidatie en log-forwarding.
4. **Klant-cloud / hybride**
5. Platformservices in de cloud (bijv. Azure tenant van klant).
6. Fobis Core mogelijk nog on-prem, gekoppeld via een secure connect (VPN, private link).
7. **RBK SaaS**
8. Platformservices draaien in RBK-cloud.
9. Klanten worden als tenants beheerd.
10. Fobis Core kan:

- Voor sommige klanten ook in RBK-cloud draaien.
- Voor andere klanten on-prem blijven, met een veilige koppeling.

In alle modellen blijft de logische architectuur gelijk; alleen de **fysieke plaats** van componenten verschilt.

---

9. SAMENVATTING

- Fobis Platform introduceert een duidelijke **platformlaag** tussen Fobis Core, apps en externe systemen.
- De architectuur is gebaseerd op:
- Een scheiding tussen **control plane** en **data plane**.
- Gestandaardiseerde platformservices: auth, autorisatie, licensering, integratie, logging.
- Multi-tenant denken, ook in on-prem scenario's.
- Dit vormt de basis voor:
- Een **SaaS-ready** Fobis-ecosysteem.
- Behoud van **on-prem/hybride** inzetbaarheid.
- Snellere en beter beheersbare ontwikkeling van nieuwe apps zoals de Drive App.

3.1.4 Deploymentmodellen

Fobis Platform – Deploymentmodellen (on-prem, klant-cloud, RBK SaaS)

1. DOEL VAN DIT DOCUMENT

Dit document beschrijft 3 **standaard deploymentmodellen** voor Fobis Platform:

1. Volledig **on-premise** bij de klant.
2. **Klant-cloud / hybride** (platform in cloud van klant, Fobis Core eventueel on-prem).
3. **RBK SaaS** (platform in RBK-cloud, koppeling naar klantomgeving).

Voor elk model beschrijven we:

- Architectuur op hoofdlijnen.
- Hoe **auth**, **licensing** en **logging/observability** zijn ingericht.
- Voordelen/nadelen en typische use-cases.

---

2. MODEL A – VOLLEDIG ON-PREMISE BIJ KLANT

2.1 Beschrijving

- Fobis Core en Fobis Platform Services draaien volledig in de **infrastructuur van de klant**:
- On-prem datacenter of private cloud volledig beheerd door klant.
- Geen of beperkte directe afhankelijkheid van internet voor de kernfunctionaliteit.

Mogelijke reden:

- Strikte regelgeving.
- Beperkte bereidheid tot cloudgebruik.
- Klant wil maximale controle over eigen omgeving.

2.2 Architectuur (hoog niveau)

- **Binnen klantomgeving**:
- Fobis Core (bestaande applicatie).
- Fobis Platform Services:
- API Gateway.
- Auth-integratie (bijv. koppeling met on-prem AD of lokale IdP).
- Licensing-service (on-prem instantie of lokale licentietoken-checker).
- Logging/observability (Serilog + lokale sinks/logviewer).
- Apps (bijv. Drive App) verbinden met de on-prem API's (eventueel via VPN of private netwerk).

2.3 Auth, licensing, logging

- **Authenticatie**:
- Vaak integratie met on-prem **Active Directory** of klant-IdP.
- OIDC-brug (bijv. AD FS of lokale IdP zoals Keycloak) binnen de klantomgeving.
- **Autorisatie**:
- Claims-based op basis van rollen en tenant-id in tokens.
- Rollen en policies worden in de platformlaag beheerd (desnoods lokaal).
- **Licensing**:
- Lokale licensing-service met periodieke export/import vanuit RBK.
- Licentietokens met beperkte geldigheid, geüpdatet via gecontroleerde kanalen.
- **Logging/observability**:

• Serilog naar lokaal bestand/SQL en eventueel lokale logstack (Seq/Elastic).

• Exportmogelijkheden om logs met RBK te delen (voor support) indien klant akkoord is.

2.4 Voors en tegens

• **Voordelen**:

• Maximale controle voor klant.

• Minder afhankelijk van internetconnectiviteit.

• Past bij huidige Fobis-aanpak (on-prem).

• **Nadelen**:

• Meer variatie in infrastructuren → hogere beheerkosten.

• Moeilijker centrale monitoring en proactieve support.

• Upgrades en uitrol kunnen zwaarder zijn.

---

3. MODEL B – KLANT-CLOUD / HYBRIDE

3.1 Beschrijving

• Fobis Platform draait in de **cloudomgeving van de klant** (bijv. Azure-subscriptie van de klant).

• Fobis Core kan:

• Ook in de klant-cloud draaien.

• Of on-prem blijven, gekoppeld via secure connect (VPN, private link).

Dit model is aantrekkelijk voor klanten die wel de voordelen van cloud willen, maar controle willen houden over tenant, beveiliging en kosten.

3.2 Architectuur (hoog niveau)

• **In klant-cloud**:

• Fobis Platform Services (API Gateway, auth integratie, licensing, logging/metrics).

• Eventueel Fobis Core (IaaS/PaaS).

• Cloud-native logging/monitoring (bijv. Application Insights, Log Analytics).

• **In klant-on-prem** (optioneel):

• Machines, shopfloor, legacy-systemen.

• Fobis Core als on-prem installatie.

• Connectie met klant-cloud via VPN/ExpressRoute/private link.

3.3 Auth, licensing, logging

• **Authenticatie**:

• Vaak direct met klant-**Entra ID**/IdP in de klant-cloud.

• OIDC-flow is op cloudniveau gestandaardiseerd.

• **Autorisatie**:

• Policy- en rollenmodel in de platformservices, zoals vastgelegd in het auth-document.

• **Licensing**:

• Licensing-service kan:

• In klant-cloud draaien (met licentiegegevens gesynchroniseerd met RBK).

• Of in RBK-cloud blijven, benaderd via beveiligde API's.

• **Logging/observability**:

• Cloud-native services van klant (Insights/Log Analytics, CloudWatch, etc.).

• RBK kan meekijken via gedeelde dashboards, toegang of data-export (afhankelijk van afspraken).

3.4 Voors en tegens

- **Voordelen**:
- Cloudvoordelen (schaalbaarheid, PaaS-diensten).
- Klant houdt regie over infrastructuur en data.
- Betere mogelijkheden voor logging en monitoring dan pure on-prem.
- **Nadelen**:
- Meer afstemming nodig met klant-IT over beheer, rechten en kosten.
- Complexere netwerkinrichting (hybride connecties).

---

4. MODEL C – RBK SAAS

4.1 Beschrijving

- Fobis Platform draait in de **RBK-cloud** (bijv. RBK's eigen Azure/AWS-omgeving).
- Klanten worden als **tenants** in deze omgeving beheerd.
- Fobis Core:
- Kan voor sommige klanten ook in RBK-cloud draaien.
- Kan voor andere klanten on-prem blijven, gekoppeld via veilige verbinding.

Dit model maakt van Fobis Platform een **volwaardige SaaS-oplossing**.

4.2 Architectuur (hoog niveau)

- **In RBK-cloud**:
- Fobis Platform Services multi-tenant:
- API Gateway.
- Auth-integratie (centrale IdP).
- Licensing-service (centrale licentiedatabase).
- Logging/observability-omgeving.
- Optioneel: Fobis Core-instanties per tenant (multi-tenant of per klant gescheiden).
- **Bij klant**:
- On-prem systemen (machines, shopfloor).
- Eventuele koppelingen via VPN, edge-componenten of API's.

4.3 Auth, licensing, logging

- **Authenticatie**:
- Centrale IdP in RBK-cloud.
- Ondersteuning voor federatie met klant-IdP als gewenst.
- Multi-tenant tokens (met `tenant_id`).
- **Autorisatie**:
- Centraal permissionmodel.
- RBK-supportrollen die meerdere tenants kunnen benaderen.
- **Licensing**:
- Volledig centrale licensing-service.
- Real-time licentiechecks in SaaS-omgeving.
- **Logging/observability**:
- Centrale logging- en metriekstack.
- Per-tenant filters en dashboards.
- Sterke basis voor proactieve support en AI-analyses.

4.4 Voors en tegens

• **Voordelen**:
• Hoogste mate van centralisatie en standaardisatie.
• Beter schaalbaar en beter te monitoren.
• Makkelijker om nieuwe features uit te rollen.
• **Nadelen**:
• Vereist vertrouwen van klanten in RBK als SaaS-provider.
• Juridische en compliance-aspecten (data-opslag, landen, sectorregels).

---

5. VERGELIJKING EN POSITIONERING

Aspect	On-prem (A)	Klant-cloud (B)	RBK SaaS (C)
Beheerverantwoordelijk	Klant	Klant (met support RBK)	RBK
Cloudgebruik	Geen of minimaal	Ja, in tenant klant	Ja, centraal
Multi-tenancy	Nee (per install)	Beperkt (per klantomgeving)	Volledig multi-tenant
Auth	On-prem AD/IdP	Klant-IdP (cloud)	Centrale IdP + federatie
Licensing	Lokaal met export/import	Hybride (cloud/licensing-service)	Volledig centraal
Logging/observability	Lokaal, beperkte centralisatie	Cloud-native met gedeelde toegang mogelijk	Centraal, sterk gestandaardiseerd
SaaS-ambitie	Laag	Middel	Hoog

---

6. RICHTLIJN VOOR KEUZE PER KLANT

Bij nieuwe of bestaande klanten kan het volgende besliskader helpen:

• **Model A (On-prem)**:
• Als regelgeving/beleid geen cloud toestaat.
• Als klant expliciet alles on-prem wil houden.
• **Model B (Klant-cloud)**:
• Als klant al sterk in cloud werkt (bijv. eigen Azure landing zone).
• Als klant wel cloud wil, maar de omgeving zelf wil beheren.
• **Model C (RBK SaaS)**:
• Als klant vertrouwen heeft in RBK als SaaS-provider.
• Als klant ontzorgd wil worden m.b.t. infrastructuur en platformbeheer.

Belangrijk: in **alle modellen** blijft de **logische architectuur hetzelfde**; alleen de fysieke plaatsing van componenten verschilt.

---

7. IMPLICATIES VOOR ONTWIKKELING EN BEHEER

• Ontwikkelteams:
• Bouwen services **deployment-agnostisch** (12-factor principes).
• Testen toepassingen in scenario's die verschillende modellen simuleren.
• Beheer:
• Beschrijft runbooks per deployment model.
• Richt monitoring/alerting per model in, maar met zoveel mogelijk hergebruik.

---

8. SAMENVATTING

- Fobis Platform ondersteunt drie hoofddeploymentmodellen: **on-prem**, **klant-cloud/hybride** en **RBK SaaS**.
- In alle modellen gelden dezelfde principes voor:
- **Auth** (OIDC/OAuth2, claims-based).
- **Licensing** (centrale service of lokale tokens).
- **Logging/observability** (Serilog + centrale of lokale aggregatie).
- Deze flexibiliteit maakt het mogelijk om:
- Vandaag klanten te bedienen in hun voorkeursmodel.
- Tegelijkertijd de **SaaS-ready platformvisie** van RBK waar te maken.

3.1.5 Licensing Service Architectuur

Fobis Platform – Licensing Service Architectuur

1. DOEL VAN DIT DOCUMENT

Dit document beschrijft de **conceptuele en logische opzet** van een centrale licensing-service voor Fobis Platform:

- Hoe licenties voor klanten/tenants worden gemodelleerd.
- Hoe apps (zoals de Drive App) en Fobis Core licenties controleren.
- Hoe dit werkt in zowel **SaaS** als **on-prem/hybride** scenario's.

---

2. UITGANGSPUNTEN EN DOELEN

- Licensering is nu sterk verweven met Fobis zelf en deels technisch geïmplementeerd.
- We willen naar een **centrale, platformbrede licenseringsoplossing** die:
- Eenduidig is voor alle applicaties.
- Als **losse service** beschikbaar is binnen het platform.
- Zowel online (SaaS) als offline/gesloten omgevingen ondersteunt.

Doelen:

- **Transparantie**: duidelijk wat een klant/tenant heeft afgenomen.
- **Beheersbaarheid**: centraal beheer en inzicht voor RBK (en deels voor klanten).
- **Flexibiliteit**: licenties per module, feature, gebruiker, omgeving kunnen defineren.
- **Veiligheid**: misbruik detecteren en beperken, zonder klanten onnodig te blokkeren.

---

3. CONCEPTUEEL LICENTIEMODEL

Belangrijke begrippen:

- **Tenant**:
- Eén klant/omgeving (bijv. "Vleesverwerker A").
- **Product**:
- Een groter geheel aan functionaliteit (bijv. "Fobis Core", "Fobis Drive").
- **Module / Feature**:
- Specifiek onderdeel binnen een product (bijv. "Drive App – Afmelden leveringen", "Reporting-module").
- **Licentie**:
- Toekenning dat een tenant een bepaald product/module mag gebruiken, onder voorwaarden (looptijd, aantal gebruikers, etc.).

Een licentie bevat o.a.:

- `tenant_id`
- `product_id` / `module_id`
- Licentietype (per user, per site, per volume)
- Geldigheidsperiode (start- en einddatum)
- Eventuele limieten (max aantal gebruikers, sites, devices)

---

4. ARCHITECTUUR VAN DE LICENSING-SERVICE

De licensing-service is een **centrale backend-service** met:

- **API's** voor:
- RBK-beheer (aanmaken/wijzigen licenties).
- Applicaties (controleren of een licentie geldig is).
- **Database**:
- Opslag van tenants, producten, modules en licentie-objecten.
- **Integratie** met:
- Authenticatie (koppeling tussen user/tenant en licentie).
- Logging/observability (licentiechecks en schendingen worden gelogd).

4.1 Licentiecontrole-patronen

Apps gebruiken de licensing-service volgens twee hoofdpatronen:

1. **Online check (SaaS / verbonden scenario)**:
2. App of backend-service roept een `/check-license`-endpoint aan:
- Input: `tenant_id`, `product/module_id`, eventueel context (user, omgeving).
- Output: `allowed: true/false`, plus metadata (reden, resterende tijd, limieten).
3. **Token-gebaseerd (offline/vaker gebruik)**:
4. De licensing-service geeft een **licentietoken** (bijv. JWT) uit:
- Bevat claims over de licenties van een tenant.
- Wordt lokaal gecached, met een beperkte geldigheidsduur.
5. Apps kunnen dit token lokaal valideren zonder elke keer de service aan te roepen.

---

5. SAAS VS. ON-PREM/HYBRIDE SCENARIO'S

5.1 SaaS / RBK-cloud

- De licensing-service draait in de RBK-cloud.
- Alle tenants (klanten) staan in één centrale database (multi-tenant).
- Apps en platformservices doen **online licentiechecks** of gebruiken licentietokens.
- RBK heeft centraal inzicht in:
- Welke klanten welke modules gebruiken.
- Gebruikspatronen (via logging).

5.2 On-premise bij klant

Mogelijke varianten:

1. **Connected on-prem** (met internet / connectie naar RBK):
2. On-prem systemen (Fobis Core, platformservices) kunnen periodiek de licensing-service in RBK-cloud benaderen.
3. Licentietokens worden lokaal gecached.
4. Bij uitval van connectie is er een **grace period** waarin bestaande licenties blijven werken.
5. **Largely offline on-prem** (beperkte of geen connectiviteit):
6. Periodieke **licentie-export** vanuit RBK (bijv. licentie-bestand of token) die in de klantomgeving wordt geïmporteerd.
7. Lokale component (lightweight licensing-checker) voert checks uit op basis van het geïmporteerde bestand/token.
8. Expiratie en grace periods zijn belangrijk om te voorkomen dat licenties eeuwig doorlopen zonder controle.

5.3 Klant-cloud / hybride

- Licensing-service kan draaien:

- In RBK-cloud (centrale service).
- Of in een klant-cloudomgeving, gesynchroniseerd met RBK.
- Scenario wordt gekozen op basis van commerciële en technische afspraken.

---

6. RELATIE TUSSEN LICENSERING, AUTHENTICATIE EN AUTORISATIE

Licensering, authenticatie en autorisatie zijn **gescheiden concepten**, maar werken samen:

- **Authenticatie**:
- Bepaalt "wie ben je?" (user/identity).
- **Autorisatie**:
- Bepaalt "wat mag je doen?" op basis van rollen/claims.
- **Licensering**:
- Bepaalt "welke functionaliteit is commercieel vrijgegeven voor deze tenant?".

In de praktijk:

- Een API-call wordt toegestaan als:
- De gebruiker correct is geauthenticeerd en geautoriseerd.
- De tenant een geldige licentie heeft voor de gevraagde functionaliteit.

Bijvoorbeeld in de Drive App:

- Chauffeur met rol `chauffeur` wil leveringen afmelden.
- De API controleert:
- Auth: token is geldig, `role=chauffeur`, `tenant_id=KlantA`.
- Licentie: `KlantA` heeft een geldige `DriveApp`-licentie met module `Afmelden leveringen`.
- Als beide waar zijn: verzoek wordt uitgevoerd.

---

7. BEHEER EN INZICHT (VOOR RBK EN KLANTEN)

De licensing-service biedt:

- Voor RBK:
- Overzicht van alle tenants en licenties.
- Inzicht in welke modules waar worden gebruikt.
- Mogelijkheid om licenties te activeren/deactiveren/aanpassen.
- Voor klanten (afhankelijk van afspraken):
- Portaal waarin ze hun eigen licenties kunnen inzien.
- Eventueel self-service uitbreiding/aanpassing (toekomst).

Licentie-activiteiten worden gelogd:

- Aanmaak en wijzigingen van licenties.
- Licentiechecks (succes/failure).
- Mogelijke misbruikpogingen (bijv. overschrijding limieten).

---

8. SECURITY EN MISBRUIKPREVENTIE

Belangrijke aandachtspunten:

- Licentietokens zijn **ondertekend** (bijv. met JWT + private key).
- Lokale caches hebben een **beperkte geldigheid** en worden regelmatig vernieuwd.
- Grace periods zijn begrensd:
- Voorkomt dat een “vergeten” omgeving onbeperkt doorloopt.
- Monitoring op:
- Onverwacht hoge load vanuit een tenant.
- Verdachte patronen in licentiechecks.

---

9. IMPLEMENTATIERICHTLIJNEN (HOOG NIVEAU)

- Bepaal een **eerste licentiemodel**:
- Start eenvoudig (bijv. per tenant per module), breid later uit met user/site-limieten.
- Definieer een **API-contract** voor licensing:
- `GET /api/licenses/{tenantId}` – overzicht licenties.
- `POST /api/licenses/check` – licentiecheck voor een module/feature.
- `POST /api/licenses/token` – uitgifte van een licentietoken.
- Integreer licensing vanuit nieuwe apps als de Drive App:
- Bij start-up: licentietoken ophalen/cachen.
- Bij cruciale acties: licentievalideren (online of via token).

---

10. SAMENVATTING

- De licensing-service maakt licensering tot een **centrale, herbruikbare capability** van Fobis Platform.
- Het model ondersteunt:
- Meerdere producten en modules per tenant.
- Zowel SaaS als on-prem/hybride scenario’s (met offline opties).
- Licensering werkt samen met auth en autorisatie, maar blijft een **eigen verantwoordelijkheid**:
- Auth: wie ben je?
- Autorisatie: wat mag je?
- Licensering: waarvoor is betaald/vrijgegeven?

3.1.6 Logging en Observability

Fobis Platform – Logging en Observability

1. DOEL VAN DIT DOCUMENT

Dit document beschrijft een **logging- en observability-strategie** voor Fobis Platform:

- Hoe we **Serilog** en andere tools inzetten voor logs, metrics en tracing.
- Hoe we zowel **on-prem** als **SaaS/hybride** scenario's ondersteunen.
- Hoe we toewerken naar **proactieve support** en **(AI-)analyses** op klantomgevingen.

---

2. UITGANGSSITUATIE

- Vandaag:
- Fobis gebruikt **Serilog** voor logging.
- Logs worden geschreven naar **bestand** en/of **SQL-database**.
- Beperkingen:
- Moeilijk om centraal over alle klantomgevingen heen te kijken.
- Beperkte mogelijkheden voor proactieve signalering.
- Analyse van incidenten is vaak **reactief** en per klantomgeving apart.

---

3. DOELARCHITECTUUR VOOR OBSERVABILITY

We onderscheiden drie lagen:

1. **Application Telemetry**
2. Serilog-logs vanuit Fobis Core, platformservices en apps.
3. Basis-metrics (requests/sec, errors, latency).
4. Eventueel distributed tracing (correlation-id).
5. **Collectie en aggregatie**
6. Lokale sinks (bestand/SQL) voor on-prem omgevingen.
7. Centrale sinks bij cloud/SaaS (bijv. Seq, Elasticsearch, Application Insights, Prometheus).
8. Optioneel: forwarding van on-prem logs naar RBK-cloud.
9. **Analyse en visualisatie**
10. Dashboards voor RBK-support en klanten.
11. Alerting op basis van drempelwaardes.
12. Toekomst: AI/ML voor anomaly detection en root cause help.

---

4. EVALUATIE VAN LOGGING LIBRARIES

4.1 Evaluatiecriteria

Voor de keuze van de logging library zijn de volgende criteria van belang:

- **Gestructureerde logging (JSON-first)**: Ondersteuning voor gestructureerde logging met rijke properties, essentieel voor analyse en filtering.
- **Azure Application Insights integratie**: Naadloze integratie met Azure Application Insights als primaire cloud-sink.
- **Performance**: Lage overhead bij hoge logvolumes, asynchrone verwerking.
- **Multi-tenant support**: Eenvoudig toevoegen van `tenant_id` en andere contextuele velden.
- **Correlation-id ondersteuning**: Native ondersteuning voor correlation-id propagatie over servicegrenzen.
- **Flexibiliteit in sinks**: Breed scala aan sinks voor verschillende deploymentmodellen (file, SQL, cloud, on-prem).
- **Maturiteit en community**: Volwassen library met actieve community en goede documentatie.
- **Migratiepad**: Continuïteit met bestaande Serilog-implementatie in Fobis.

4.2 Vergelijking van opties

Serilog

Sterke punten:
- ✅ **Uitstekende gestructureerde logging**: JSON-first design met native property capture.
- ✅ **Rijke Azure Application Insights integratie**: `Serilog.Sinks.ApplicationInsights` biedt directe integratie met Application Insights, waarbij gestructureerde properties automatisch als custom dimensions worden opgeslagen.
- ✅ **Hoge performance**: Asynchrone sinks, efficiënte property capture, lage overhead.
- ✅ **Flexibele context**: `LogContext.PushProperty()` maakt het eenvoudig om `tenant_id`, `correlation_id` en andere context toe te voegen.
- ✅ **Uitgebreide sink-ecosysteem**: 100+ sinks beschikbaar (file, SQL, Seq, Elasticsearch, Application Insights, CloudWatch, etc.).
- ✅ **Mature en stabiel**: Actief onderhouden, veel gebruikt in .NET-ecosysteem.
- ✅ **Bestaande implementatie**: Fobis gebruikt al Serilog, minimale migratiekosten.

Zwakke punten:
- ⚠️ Geen ingebouwde metrics/tracing (alleen logging), maar dit is geen vereiste voor fase 1.
- ⚠️ Voor distributed tracing is aanvullende tooling nodig (bijv. OpenTelemetry), maar dit is voor later.

Azure Application Insights integratie:
- Directe integratie via `Serilog.Sinks.ApplicationInsights`.
- Gestructureerde properties worden automatisch als custom dimensions opgeslagen.
- Ondersteuning voor zowel telemetry channels als directe HTTP-export.
- Goede performance met batching en asynchrone verwerking.

Microsoft.Extensions.Logging (MEL)

Sterke punten:
- ✅ Standaard in .NET, geen extra dependency.
- ✅ Goede integratie met ASP.NET Core middleware.
- ✅ Provider-model biedt flexibiliteit.

Zwakke punten:
- ❌ **Beperkte gestructureerde logging**: MEL is primair ontworpen voor string-based logging. Gestructureerde logging vereist meer boilerplate code en is minder elegant.
- ❌ **Minder performant**: Bij hoge volumes presteert MEL minder goed dan Serilog.
- ❌ **Beperkte Azure integratie**: Application Insights integratie vereist extra configuratie en is minder direct dan Serilog.
- ❌ **Minder rijke property capture**: Contextuele informatie toevoegen is minder intuïtief.
- ❌ **Migratiekosten**: Vereist herschrijven van bestaande Serilog-code.

Azure Application Insights integratie:
- Vereist `Microsoft.ApplicationInsights.AspNetCore` package.
- Minder directe mapping van gestructureerde properties naar custom dimensions.
- Meer configuratie nodig voor optimale integratie.

NLog

Sterke punten:
- ✅ Mature en stabiele library.
- ✅ Goede performance.
- ✅ Veel targets (sinks) beschikbaar.

Zwakke punten:
- ⚠️ **Minder focus op gestructureerde logging**: NLog ondersteunt gestructureerde logging, maar het is niet de primaire focus zoals bij Serilog.
- ⚠️ **Complexere configuratie**: XML-configuratie kan complexer zijn dan Serilog's fluent API.
- ⚠️ **Minder moderne API**: API is minder elegant voor gestructureerde logging.
- ⚠️ **Migratiekosten**: Vereist herschrijven van bestaande Serilog-code.

Azure Application Insights integratie:
- Beschikbaar via `NLog.Targets.ApplicationInsights`.
- Functioneel, maar minder direct dan Serilog's integratie.

OpenTelemetry

Sterke punten:
- ✅ **Unified observability**: Logs, metrics en traces in één framework.
- ✅ **Industry standard**: Wordt de standaard voor observability.
- ✅ **Toekomstbestendig**: Goede integratie met moderne tools.

Zwakke punten:
- ❌ **Te vroeg voor fase 1**: OpenTelemetry in .NET is relatief nieuw en minder mature dan Serilog.
- ❌ **Complexiteit**: Meer complex om op te zetten en te configureren.
- ❌ **Overkill voor alleen logging**: Als je alleen logging nodig hebt (fase 1), is OpenTelemetry mogelijk overkill.
- ❌ **Vaak nog Serilog nodig**: Veel OpenTelemetry-implementaties gebruiken Serilog als logging provider.

Azure Application Insights integratie:
- Goede integratie via OpenTelemetry Azure Monitor exporter.
- Maar vereist meer setup en configuratie.

4.3 Aanbeveling

Serilog blijft de beste keuze voor Fobis Platform.

Rationale:

1. **Voldoet aan alle vereisten**: Serilog voldoet uitstekend aan alle criteria, met name gestructureerde logging, Azure Application Insights integratie en multi-tenant support.
2. **Bestaande investering**: Fobis gebruikt al Serilog. Continuïteit vermindert risico's en migratiekosten.
3. **Superieure Azure integratie**: `Serilog.Sinks.ApplicationInsights` biedt de meest directe en performante integratie met Application Insights, waarbij gestructureerde properties automatisch als custom dimensions worden opgeslagen.
4. **Toekomstige uitbreidbaarheid**: Serilog kan later worden gecombineerd met OpenTelemetry voor metrics en tracing, zonder Serilog te vervangen. Dit biedt een duidelijk migratiepad.
5. **Performance en maturiteit**: Serilog is bewezen performant en mature, met een actieve community en uitgebreide documentatie.

Implementatie-overwegingen:

- Gebruik `Serilog.Sinks.ApplicationInsights` voor Azure-cloud deployments.
- Behoud bestaande sinks (file, SQL) voor on-prem deployments.
- Implementeer een standaard logging-package met middleware voor correlation-id en tenant context.
- Overweeg later OpenTelemetry toe te voegen voor metrics en tracing, naast (niet in plaats van) Serilog.

---

5. SERILOG ALS BASIS

Serilog blijft de **standaard logging library** in .NET-componenten van Fobis Platform.

Afspraken:

- **Gestructureerde logging**:
- Logrecords in JSON-formaat met vaste velden:
- `timestamp`, `level`, `messageTemplate`, `tenant_id`, `user_id` (indien bekend), `correlation_id`, `source`, `event_type`.
- **Correlation-id**:
- Elke request krijgt een unieke `correlation_id`.
- Deze id wordt doorgegeven over de keten (app → API → services → Fobis Core).
- Maakt end-to-end tracering van een probleem mogelijk.
- **Tenantcontext**:
- Waar mogelijk wordt een `tenant_id` meegegeven in logrecords.
- Essentieel voor multi-tenant/SaaS scenario's én voor gescheiden analyse per klant.

---

6. LOGGINGSTRATEGIE PER DEPLOYMENTMODEL

6.1 On-premise bij klant

- **Minimaal**:
- Logging naar lokaal bestand en/of SQL (zoals nu).
- Zorgvuldig configuratiebeheer:
- Rotatie/retentie van logbestanden.
- Max schijfruimte voor logs.
- **Optioneel**:
- Lokale log-viewer (bijv. Seq on-prem, Kibana, of een eigen viewer).
- Exportmogelijkheden (bijv. periodieke upload naar RBK voor analyse).

6.2 Klant-cloud / hybride

- Logging van platformservices en Fobis Core:
- Naar cloud-native oplossingen (bijv. Azure Application Insights, Log Analytics, AWS CloudWatch).
- Eventueel naar een dedicated loggingstack (ELK/EFK, Seq, Grafana Loki).
- Koppeling met RBK:
- RBK kan (met toestemming) meekijken in log- en metriekdata.
- Shared dashboards voor klant + RBK-support.

6.3 RBK SaaS

- Centralisatie:
- Alle SaaS-omgevingen loggen naar een centrale logging/metrics oplossing.
- Multi-tenant filtering via `tenant_id`.
- Proactieve monitoring:
- Alerting per tenant (bijv. verhoogde error rates).
- Baseline-analyses over alle tenants heen.

---

7. METRICS EN HEALTH MONITORING

Naast logging definiëren we een set **basis-metrics**:

- **Technisch**:
- Request volume per service.
- Error rates (4xx/5xx).
- Latency (gemiddelde, percentielen).
- Resourcegebruik (CPU, memory, disk, queue-lengtes).
- **Functioneel** (domein-specifiek):
- Aantal ritten/leveringen verwerkt (Drive App).
- Aantal mislukte leveringen, timeouts naar Fobis Core.
- Aantal licentie-checks en failures.

Alle services implementeren een **health endpoint** (bijv. `/health`):

- Duidelijke status: `Healthy`, `Degraded`, `Unhealthy`.
- Gebruikt door:
- Load balancers/orchestrators (bijv. Kubernetes).
- Monitoring- en alertingtooling.

---

8. PROACTIEVE SUPPORT EN AI-ONDERSTEUNDE ANALYSES

Door logs en metrics centraal en gestructureerd te verzamelen, ontstaat de basis voor:

- **Proactieve incidentdetectie**:
- Alerts op verhoogde foutpercentages bij een specifieke tenant.
- Signalering van performanceproblemen voordat de klant belt.
- **AI-ondersteunde analyse**:
- Gebruik van log- en metriekdata als input voor AI/ML-modellen:
- Anomaly detection (bijv. ongebruikelijk patroon van fouten).
- Suggesties voor mogelijke oorzaken bij incidenten.
- Mogelijk in volgende stappen:
- Conversational debugging ("wat is er gebeurd bij klant X gisteren?").

Voorwaarde: **consistent, rijk gelabelde data** (tenant, correlation-id, eventtype).

---

9. CONFIGURATIE EN BEHEER

Belangrijk is een **goed configureerbare oplossing**:

- Loggingniveau per component instelbaar (bijv. `Information`, `Warning`, `Debug`).
- Schakelen tussen verschillende sinks zonder code-aanpassing:
- Bijv. via config: file + SQL, of file + cloud sink.
- Per tenant of omgeving andere instellingen:
- Meer detail in test/acceptatie.
- Strakkere retentie in productie.

RBK beheert:

- Standaardconfiguraties per deploymentmodel.
- Templates voor Serilog-config (in code en in config-bestanden).

---

10. INTEGRATIE MET SECURITY EN PRIVACY

Logging moet voldoen aan **privacy- en security-eisen (AVG)**:

- Minimaliseer het loggen van **persoonsgegevens**:
- Geen gevoelige data in logmessages (bijv. geen volledige namen/adressen als het niet nodig is).
- Pseudonimiseren waar mogelijk.
- Toegang tot centrale logs:
- Rolgebaseerde toegang (RBK-support, klantbeheerder).
- Audittrail van wie wat heeft ingezien.

---

11. IMPLEMENTATIERICHTLIJNEN (HOOG NIVEAU)

- Gebruik Serilog **enkel nog gestructureerd** (geen losse stringlogs zonder properties).
- Introduceer een standaard **logging-package** voor platformservices met:
- Middleware voor correlation-id.
- Tenant- en usercontext in logs.
- Eenduidige configuratie van sinks.
- Kies 1–2 **voorkeursstacks** voor centrale logging:
- Bijv. Serilog → Seq / Elastic / Application Insights.
- Beschrijf per deploymentmodel de default-keuze.

---

12. SAMENVATTING

- Fobis Platform bouwt verder op Serilog, maar plaatst dit in een bredere **observability-architectuur**.
- Logs, metrics en (eventueel) traces worden:
- Gestructureerd en rijk gelabeld (tenant, correlation-id, eventtype).
- Per deploymentmodel op passende wijze opgeslagen en geaggregeerd.
- Dit maakt:
- **Proactieve service en support** mogelijk.
- Toekomstige **AI-ondersteunde analyses** haalbaar.
- Beheer van SaaS én on-prem omgevingen **beter voorspelbaar en schaalbaar**.

3.1.7 Authenticatie en Autorisatie

Fobis Platform – Authenticatie en autorisatie

1. DOEL VAN DIT DOCUMENT

Dit document beschrijft de **doelarchitectuur** voor authenticatie en autorisatie binnen Fobis Platform:

- Hoe gebruikers en systemen zich **authenticeren** (wie ben je?).
- Hoe wordt bepaald **wat** iemand mag doen (autorisatie).
- Hoe dit past in een **SaaS-ready, on-prem capable** platformarchitectuur.

Het is richtinggevend en bedoeld als basis voor detailontwerp en implementatie met partners zoals 4DotNet.

---

2. UITGANGSSITUATIE EN WENSEN

- Vandaag:
- Authenticatie, autorisatie en licensering zijn **op verschillende manieren** in Fobis ingebouwd.
- Vaak **applicatie-specifieke logica**, weinig centralisatie.
- Integratie met externe identity-oplossingen (bijv. klant-AD) is beperkt of ad-hoc.
- Wensen:
- Een **open, moderne oplossing** gebaseerd op industrie-standaarden.
- Oplossing moet **onafhankelijk** zijn van één specifieke technologie, maar wel goed te beheren.
- Ondersteunen van:
- On-prem ONLY klanten.
- Klanten met eigen cloud / identity (bijv. Entra ID).
- Toekomstige RBK-SaaS-varianten.

---

3. CONCEPTUEEL MODEL

We hanteren een **lagenmodel**:

1. **Identity Provider (IdP)** – authenticatie
2. Verifieert de identiteit van gebruikers (login).
3. Voorbeeld: Entra ID, Keycloak, IdentityServer, Auth0, on-prem AD FS.
4. Spreekt bij voorkeur **OpenID Connect (OIDC)** en **OAuth2**.
5. **Fobis Platform – Identity & Access Layer**
6. Ontvangt tokens van de IdP (ID token + access token).
7. Verwerkt claims (user-id, rollen, tenant-id, etc.).
8. Beheert rollen, permissies en policies.
9. **Applicaties en services**
10. Verifiëren tokens via de platformlaag of rechtstreeks bij de IdP (afhankelijk van scenario).
11. Nemen autorisatiebeslissingen op basis van claims + platform-permissionmodel.

---

4. AUTHENTICATIE – ONTWERP

4.1 Standaarden

- **OpenID Connect (OIDC)** voor user sign-in:
- Gebruiker logt in bij de IdP (bijv. Entra ID).
- Applicatie ontvangt een **ID token** met info over de gebruiker.

- **OAuth2** voor API-toegang:
- Applicaties (bijv. Drive App) vragen een **access token** aan bij de IdP.
- API Gateway en backend-services valideren dit token.

4.2 Identity Provider-keuze

De architectuur laat meerdere IdP's toe, maar we definiëren een **voorkeursrichting**:

- Voor RBK-managed omgevingen (SaaS of shared cloud):
- Een centrale IdP (bijv. Entra ID tenant van RBK of dedicated IdP zoals Keycloak).
- Voor klant-specifieke omgevingen:
- Federatie met klant-IdP (bijv. klant Entra ID / AD) waar mogelijk.
- Anders: een door RBK beheerde IdP waarvoor de klant accounts beheert.

4.3 Multi-tenant en tenants

- Elke klant is een **tenant**.
- Tokens bevatten minimaal:
- `sub` (subject / user-id).
- `tenant_id` (klant / installatie).
- Rollen/claims (bijv. `role: chauffeur`, `role: planner`).

---

5. AUTORISATIE – ONTWERP

We gebruiken een combinatie van:

- **Role-Based Access Control (RBAC)**: rollen (chauffeur, planner, beheerder, RBK-support).
- Eventueel aangevuld met **fijnmazigere permissies** (feature flags, claims voor specifieke modules).

5.1 Rollen- en rechtenmodel

Op platformniveau definiëren we ten minste:

- **Gebruikersrollen** (per tenant):
- Chauffeur, planner, productiemanager, systeembeheerder klant.
- **Platformrollen**:
- RBK-support, RBK-admin.

Rollen bepalen:

- Welke API's iemand mag aanroepen.
- Welke gegevens iemand mag zien.
- Welke tenant(s) iemand kan benaderen (meestal één, soms meer voor RBK-support).

5.2 Claims-based autorisatie

Autorisatiebeslissingen zijn gebaseerd op **claims in het token**, zoals:

- `role` – logische rol van de gebruiker.
- `tenant_id` – tot welke klant/omgeving de gebruiker behoort.
- Optioneel: `permissions` – lijst van fijnmazige rechten.

Voorbeeld in de Drive App:

- Een gebruiker met `role=chauffeur` en `tenant_id=KlantA` mag:
- Alleen ritten van **KlantA** zien.
- Alleen endpoints aanroepen die voor chauffeurs bedoeld zijn.

5.3 Centralisatie vs. lokale checks

• **Centralisatie**:
• Een centrale authorization-service of library met policies (bijv. “behoort tot tenant van deze resource”, “heeft rol X voor module Y”).
• **Lokale checks**:
• Services en apps voeren checks uit op basis van claims, gebruikmakend van gedeelde policies.

Doel: één consistent model, maar zonder onnodige round-trips naar een centrale service bij elke call.

---

6. INTEGRATIE MET FOBIS CORE EN ON-PREM OMGEVINGEN

6.1 On-prem Fobis Core

Voor on-prem Fobis-installaties zijn er meerdere scenario’s:

1. **Fobis Core blijft eigen auth houden (tijdelijk)**:
2. Nieuwe platformapps (zoals Drive App) gebruiken al OIDC en het nieuwe model.
3. Fobis Core heeft nog legacy-auth; bridging is nodig in integratielaag.
4. **Geleidelijke migratie naar het nieuwe model**:
5. Fobis Core accepteert tokens van de IdP (bijv. via een gateway of nieuw login-pad).
6. Langzaam worden interne auth-logica en usermanagement uitgefaseerd.

6.2 Klant-specifieke directories

• Sommige klanten willen inloggen met hun eigen AD/Entra ID-accounts.
• Oplossing:
• Federatie: Fobis Platform vertrouwt tokens van de klant-IdP.
• Mapping: claims uit klant-IdP worden gemapt naar Fobis-rollen en -tenants.

---

7. SECURITY-ASPECTEN

Belangrijke aandachtspunten:

• **Sterke authenticatie**:
• Ondersteuning voor MFA waar relevant (bijv. bij planners/beheer).
• **Tokenbeveiliging**:
• Korte geldigheid van access tokens, refresh tokens waar nodig.
• Secure opslag van tokens op devices (Drive App).
• **Least privilege**:
• Rollen en rechten zo minimaal mogelijk toekennen.
• **Auditing**:
• Belangrijke autorisatiebeslissingen en login-events worden gelogd (via logging/observability-platform).

---

8. IMPLEMENTATIERICHTLIJNEN

Voor partners en interne teams:

• Gebruik standaard **OIDC/OAuth2 libraries** voor de gekozen tech-stack (.NET, React/React Native, etc.).
• Implementeer **centralized auth middleware** in backend-services:
• Validatie van JWT-tokens.
• Extractie van claims en tenant-context.
• Gebruik een **policy-based autorisatiemodel**:
• Policies (bijv. `RequireRole("planner")`, `RequireTenantMatch()`) op controller/endpoint-niveau.

9. SAMENVATTING

- Fobis Platform gebruikt **OpenID Connect** + **OAuth2** voor authenticatie en API-toegang.
- Autorisatie is **claims-based**, met een centraal gedefinieerd rollen- en permissiemodel.
- De architectuur ondersteunt:
- RBK-SaaS, klant-cloud en pure on-prem scenario's.
- Federatie met klant-identity systemen.
- Dit legt een consistente basis voor alle toekomstige apps, waaronder de Drive App, en maakt Fobis Platform **open, veilig en goed beheersbaar**.

3.1.8 Praatplaat en Stakeholdercommunicatie

Fobis Platform – Praatplaat & Stakeholdercommunicatie

Dit document helpt om de **kern van Fobis Platform** en de **SaaS-ready, on-prem capable** strategie helder uit te leggen aan management, collega's en partners.

Je kunt het gebruiken als basis voor een slide deck (bijv. 6–8 slides) en als spiekbrief bij gesprekken.

---

1. KERNBOODSCHAP IN ÉÉN ZIN

Fobis Platform maakt van ons huidige on-prem Fobis een modern, open en SaaS-ready platform, dat óók on-prem en hybride inzetbaar blijft, en waarmee we kunnen doorgroeien naar een industry-standaard in de foodproductie.

---

2. SLIDE-INDELING (VOORSTEL)

Slide 1 – Titel & context
- Titel: *"Fobis Platform – van product naar platform"*
- Kernpunten:
- Fobis vandaag: on-prem product bij foodproducenten.
- Ambitie: Fobis Platform als basis voor moderne apps (zoals de Drive App) en partneroplossingen.

Slide 2 – Huidige situatie & uitdagingen
- Bullets:
- Fobis draait **on-prem** bij klanten; elke omgeving is nét anders.
- Auth, autorisatie en licensering zijn historisch gegroeid en niet centraal gestandaardiseerd.
- Logging vooral lokaal; **moeilijk centraal overzicht** en proactieve support.
- Beheersbaarheid van al die verschillende installaties is nu óók complex.

Slide 3 – Visie: SaaS-ready, on-prem capable
- Visualiseer:
- Fobis Core (bestaand systeem) + Fobis Platform Services + Apps (Drive App, etc.).
- Kernboodschappen:
- We ontwerpen als **SaaS-platform**, maar **on-prem/hybride blijft volledig ondersteund**.
- Scheiding tussen **control plane** (auth, licensing, logging) en **data plane** (shopfloor, machines, Fobis Core).

Slide 4 – Kerncomponenten van het platform
- Noem de belangrijkste bouwstenen:
- Auth & Identity (OIDC/OAuth2).
- Authorization (claims-based, rollen & policies).
- Licensing-service (modules/features per tenant).
- Logging & observability (Serilog + centrale aggregatie).
- Integratielaag (API's, events) richting Fobis Core en externe systemen.

Slide 5 – Deploymentmodellen: on-prem, klant-cloud, RBK SaaS
- Tabel of plaatje met 3 blokken:
- **On-prem**: alles bij klant, optionele connectie naar RBK.
- **Klant-cloud/hybride**: platform in klant-cloud, Fobis Core on-prem of in diezelfde cloud.
- **RBK SaaS**: platform in RBK-cloud, klanten als tenants.
- Boodschap:
- Logische architectuur is **overal dezelfde**; alleen de fysieke plek verschilt.

Slide 6 – Beheersbaarheid: angst versus realiteit
- Herken de angst:
- "SaaS is moeilijk te beheren" – multi-tenant, 24/7, security, etc.
- Nuancering:
- De huidige situatie (vele on-prem installs) is óók complex:
- Versieverspreiding, maatwerk, weinig centraal zicht.
- Met een platform en standaard deploymentmodellen wordt beheer op termijn **juist eenvoudiger**:

- Centraal ingerichte auth/licensing/logging.
- Gestandaardiseerde monitoring en tooling.

Slide 7 – Businessambitie: Fobis als industry-platform
- Boodschap:
- Niet alleen “we koppelen met Exact/Odoo”, maar: “zij koppelen met **Fobis Platform**”.
- Fobis wordt een **aanspreekpunt in de keten** voor productie- en logistieke data.
- Voorbeeld: Drive App als eerste referentie; meer apps en partneroplossingen volgen.

Slide 8 – Volgende stappen
- Kort:
- Uitwerken van platformkaders (visie, architectuur, auth, licensing, logging, deployment).
- Drive App als eerste app op het platform.
- Samen met 4DotNet (Henk en Bob) implementatiepad scherp maken.

---

3. KERNZINNEN OM TE GEBRUIKEN IN GESPREKKEN

Een aantal formuleringen die je kunt gebruiken:

- **Over visie**
- “We bouwen een platform dat nu al waarde levert voor on-prem klanten, maar ook klaar is voor SaaS.”
- “SaaS-ready, on-prem capable: beide werelden, zelfde platform.”
- **Over beheersbaarheid**
- “Een goed ontworpen SaaS-ready architectuur maakt beheer voorspelbaarder, niet ingewikkelder.”
- “In plaats van 50 verschillende installaties, willen we 3 duidelijke deploymentmodellen.”
- **Over security en standaarden**
- “We stappen over op open standaarden: OpenID Connect, OAuth2, gestructureerde logging.”
- “Authenticatie, autorisatie en licensering worden centrale capabilities in plaats van ad-hocoplossingen.”
- **Over commerciële ambitie**
- “Fobis Platform moet een naam worden waar anderen mee wíllen koppelen.”
- “Fobis Next is niet één product, maar een platform plus een ecosysteem eromheen.”

---

4. HOE DEZE STUKKEN BIJ ELKAAR HOREN

Je hebt nu de volgende documenten in `Verkenning`:

- **Visie en Framing** – waarom Fobis Platform en wat is de ambitie.
- **Doelarchitectuur** – hoe Fobis Core, platformservices en apps samenhangen.
- **Authenticatie en Autorisatie** – OIDC/OAuth2 en claims-based access.
- **Licensing Service Architectuur** – centrale licensering voor SaaS én on-prem.
- **Logging en Observability** – Serilog + centrale aggregatie en proactieve support.
- **Deploymentmodellen** – on-prem, klant-cloud, RBK SaaS.
- **Praatplaat & Stakeholdercommunicatie** – dit document.

Samen vormen ze de **kaders** waarmee je begin januari met Henk en Bob direct de diepte in kunt.

---

5. SUGGESTIES VOOR GEBRUIK

- Gebruik de **praatplaat-slides** voor:
- Management-/MT-overleg.
- Afstemming met sales/consultants.
- Kick-off met 4DotNet.
- Gebruik de **detaildocumenten** voor:

• Architectuursessies.
• Technische beslissingen (IdP-keuze, loggingstack, etc.).

3.2 Apps

3.2.1 Drive-app

Visie, scope en backlog

DRIVE APP – VISIE, SCOPE EN EERSTE BACKLOG

1. Context en doel

- **Context**
- Fobis wordt vandaag on-prem ingezet bij foodproducenten (vleesverwerkers, bakkerijen, etc.).
- Fobis Platform gaat Fobis ontsluiten voor moderne, cloud-ready toepassingen.
- De **Drive App** is de eerste referentie-app op Fobis Platform.
- **Doel Drive App**
- Chauffeurs een eenvoudige, robuuste app geven om **ritten te zien** en **leveringen af te melden**.
- Real-time of near real-time terugkoppeling via het Fobis Platform naar Fobis **of andere plannings-/ERP-systemen** (zoals bijvoorbeeld Exact Globe of Odoo), en van daaruit naar de klant.
- User experience minimaal op niveau van (of beter dan) bestaande oplossingen zoals Distririjden.

---

2. Visie

De Drive App is:

- De **primair gebruikte tool voor chauffeurs** tijdens hun ritten.
- Een **showcase** van Fobis Platform:
- Gebruikt de centrale auth/autorisatie/licensing.
- Werkt via de API- en integratielaag naar Fobis Core.
- Laat zien hoe logging/observability en proactieve support werken.
- Een app die:
- **Eenvoudig** te gebruiken is (minimale afleiding, grote knoppen, weinig tekst).
- **Robuust** is in omgevingen met slechte of wisselende verbinding.
- Makkelijk uit te rollen en te beheren is (updates, configuratie, licenties).

---

3. Belangrijkste gebruikersrollen

- **Chauffeur**
- Dagelijkse gebruiker van de app.
- Heeft een persoonlijk account of een device-/ritgebonden login (nog te beslissen).
- **Planner**
- Werkt in Fobis (Core / planningmodule).
- Wil real-time inzicht in rit- en afleverstatus.
- **Service / Support (RBK en klant)**
- Wil kunnen zien wat er misgaat als een chauffeur problemen meldt.
- Gebruikt logging/monitoring om issues snel te analyseren.

---

4. Hoofdgebruikersflows (niveau 1)

1. **Inloggen als chauffeur**

2. Chauffeur start de app.
3. Auth via gekozen IdP (of vereenvoudigde login, afhankelijk van klantscenario).
4. App haalt de ritten van de chauffeur op voor vandaag (en eventueel de komende dagen).
5. **Overzicht ritten van vandaag**
6. Chauffeur ziet een lijst met ritten (met basisinformatie: ritnummer, starttijd, aantal stops, status).
7. Kan een rit selecteren om de stops/leveringen te zien.
8. **Overzicht stops/leveringen binnen een rit**
9. Lijst met stops in ritvolgorde.
10. Per stop: klantnaam, adres, tijdvenster (indien relevant), aantal colli/pallets/gewicht.
11. Status per stop (niet gestart / onderweg / afgeleverd / niet-afgeleverd).
12. **Levering afmelden (geslaagd)**
13. Chauffeur kiest een stop.
14. App toont details + mogelijkheid “Afgeleverd”.
15. Na bevestiging:
    - Status in app → Afgeleverd.
    - Event/aanroep naar Fobis Platform → Fobis Core (order/delivery-update).
16. **Levering afmelden (niet geslaagd)**
17. Chauffeur kiest een stop.
18. App toont standaardredenen (bijv. “ontvanger niet aanwezig”, “adres onbekend”, “schade”, …).
19. Chauffeur kiest een reden (en eventueel toelichting/foto).
20. Status in app → Niet-afgeleverd + reden.
21. **(Optioneel) Foto/handtekening vastleggen**
22. Bij geslaagde levering:
    - Handtekening van ontvanger vastleggen.
    - Eventueel foto van bon/papieren.
23. Bij niet-geslaagde levering:
    - Foto van situatie (bijv. gesloten deur, schade).
24. **Synchronisatie en offline gebruik**
25. App kan in beperkte mate offline werken:
    - Data lokaal cachen (ritten, stops).
    - Afmeldacties opslaan in een lokale queue.
26. Bij verbinding:
    - Queue wordt weggeschreven naar Fobis Platform.

5. MVP-scope (voorstel)

In scope voor MVP:

- Inloggen chauffeur (basis, via platform-auth).
- Overzicht ritten van vandaag (1 chauffeur = 1 set ritten).
- Overzicht stops/leveringen per rit (basisinformatie).
- Afmelden levering:
- Status: afgeleverd / niet-afgeleverd.
- Bij niet-afgeleverd: kiezen uit standaardredenen.
- Basis offline gedrag:
- Ritgegevens cache voor de dag.
- Afmeldingen bufferen bij tijdelijke netwerkissues.
- Logging van kernacties:
- Login, rit-selectie, afmeldacties, fouten.

Buiten scope MVP (voor later):

- Handtekeningen en foto's.
- Route-optimalisatie / navigatie-integratie (Google Maps / Waze).
- Push-notificaties (bijvoorbeeld ritwijzigingen).
- Geavanceerde configuratie per klant (bijv. eigen redenenlijsten beheren via portal).

---

6. Eerste user stories (voorbeeld)

6.1 Authenticatie & basisgebruik

- Als **chauffeur** wil ik kunnen **inloggen** met mijn bedrijfsaccount of chauffeurcode, zodat ik toegang heb tot mijn ritten.
- Als **chauffeur** wil ik na inloggen **direct mijn ritten van vandaag zien**, zodat ik snel kan starten.
- Als **chauffeur** wil ik per rit **alle stops met basisinformatie** zien, zodat ik weet waar en wat ik moet leveren.

6.2 Afmelden leveringen

- Als **chauffeur** wil ik een stop als **'afgeleverd' kunnen markeren**, zodat de planning realtime weet dat de levering gelukt is.
- Als **chauffeur** wil ik een stop als **'niet-afgeleverd' met een reden kunnen markeren**, zodat voor iedereen duidelijk is waarom de levering niet gelukt is.
- Als **planner** wil ik in Fobis **per rit en per stop de actuele afleverstatus zien**, zodat ik klanten kan informeren en snel kan reageren op problemen.

6.3 Offline gedrag

- Als **chauffeur** wil ik mijn ritten **blijven zien en kunnen afmelden** ook als ik tijdelijk geen verbinding heb, zodat ik door kan werken.
- Als **planner** wil ik dat de app **afmeldingen synchroniseert zodra er weer verbinding is**, zodat de data alsnog in Fobis terechtkomt.

6.4 Logging & support

- Als **RBK-supportmedewerker** wil ik **logentries per rit en per chauffeur kunnen terugvinden**, zodat ik snel kan zien wat er misging bij een incident.
- Als **klantbeheerder** wil ik **overzicht hebben van het gebruik van de Drive App** (bijv. aantal ritten, mislukte leveringen), zodat ik procesverbeteringen kan doen.

---

7. Non-functionele eisen (NFR's)

- **Performance**
- App moet vlot reageren op gangbare Android-toestellen (en eventueel iOS, afhankelijk van keuze).

• Laden van ritten/stops: binnen enkele seconden.

• **Robuustheid**

• App moet om kunnen gaan met wisselende netwerkverbindingen (4G/5G, platteland).

• Acties mogen niet “kwijtraken” bij tijdelijke storingen (queue/persistent storage).

• **Beveiliging**

• Beveiligde communicatie (HTTPS/TLS).

• Minimale data op device; gevoelige data versleuteld opslaan.

• Sessietimeout en mogelijkheid om op afstand toegang te blokkeren (bij verlies device).

• **Usability**

• Grote knoppen, duidelijke statusiconen, minimale tekst.

• UI afgestemd op gebruik in vrachtwagens/busjes (licht/donker, contrast).

• **Beheer**

• App-updates zoveel mogelijk via standaard app stores of MDM van klant.

• Configuratie (bijv. omgeving, loggingniveau) via Fobis Platform, niet hardcoded.

---

8. Technische uitgangspunten (richtinggevend)

• **Frontendtechnologie**

• Te kiezen met 4DotNet (bijv. React Native, Flutter, .NET MAUI).

• Belangrijk: goede offline-ondersteuning en integratie met platform-auth.

• **Back-end integratie**

• Drive App praat uitsluitend met Fobis Platform API’s.

• Fobis Platform vertaalt dit naar Fobis Core (geen directe toegang vanaf device naar Core).

• **Security**

• OAuth2 / OIDC flows voor mobile (PKCE, etc.).

• Tokens bevatten `tenant_id`, `user_id`, rol (`chauffeur`) en eventueel licentieclaims.

• **Licensing**

• Toegang tot Drive App is gekoppeld aan licentie voor product/module “Drive”.

---

9. Openstaande vragen / beslispunten

Deze punten kunnen in januari met Henk en Bob verder worden uitgewerkt:

• Loginmodel:

• Persoonlijke accounts per chauffeur, of device-/ritgebonden accounts?

• Hoe om te gaan met uitzendkrachten of tijdelijke chauffeurs?

• Offline-scope:

• Hoe ver moet offline functionaliteit gaan in MVP (alleen afmelden, of ook ritten kunnen laden op voorhand)?

• Platformkeuzes:

• Definitieve keuze voor mobiele tech-stack.

• Keuze voor concrete IdP in de eerste klantcases.

• Devicebeleid:

• Eigen devices van chauffeurs (BYOD) vs. bedrijfsdevices.

---

10. Inspiratie vanuit DistriRijden – post-MVP epics

Op basis van inspiratie uit oplossingen zoals DistriRijden (`https://distridata.nl`) kunnen we een aantal **post-MVP epics** definiëren:

- **Digitale ondertekening van pakbonnen**
- Bij geslaagde levering kan de ontvanger op het device tekenen.
- Handtekening en relevante metadata (tijd, locatie, chauffeur) worden opgeslagen in Fobis.
- Vervangt papieren bonnen voor bewijs van aflevering.
- **Retouren en emballage-afhandeling**
- Chauffeur kan retouren (fouten, verkeerde levering, beschadigde goederen) registreren tijdens de stop.
- Registratie van emballage (pallets, kratten, fusten) – meegegeven en retour genomen.
- Data wordt direct in Fobis verwerkt voor voorraad- en emballageadministratie.
- **Realtime statusupdates en tracking**
- Naast status van stops ook (optioneel) basis trackinginformatie:
- Laatst bekende status/tijdstempel per stop.
- Optioneel: locatiegegevens indien toegestaan/zinvol.
- Planner en (later) klantenportaal kunnen live mee kijken in voortgang.
- **Uitgebreide foto-dossiers**
- Foto's bij geslaagde leveringen (schade, specifieke situaties).
- Foto's bij niet-geslaagde leveringen (gesloten deur, onveilige situatie).
- Foto's bij retouren en emballage (beschadiging, aantallen).
- Deze beelden worden in Fobis gekoppeld aan rit/stop/delivery en zijn terug te vinden in rapportages.

Deze epics sluiten aan bij de voordelen die `https://distridata.nl` beschrijft (papierloos werken, minder fouten, betere bewijsvoering) en vormen natuurlijke vervolgstappen na de MVP.

---

11. Samenvatting

- De Drive App is de eerste **referentie-app** op Fobis Platform en richt zich op het **afmelden van leveringen door chauffeurs**.
- Dit document geeft:
- Visie, scope en MVP-afbakening.
- Hoofdgebruikersflows en eerste user stories.
- Non-functionele eisen, openstaande beslispunten en post-MVP epics (geïnspireerd door DistriRijden).
- Het vormt daarmee een goede basis voor een gezamenlijke **inception met 4DotNet** om het ontwerp en de implementatie te starten.

Integratie met ERP en planningssystemen

DRIVE APP – INTEGRATIE MET ERP- EN PLANNINGSSYSTEMEN VIA FOBIS PLATFORM

1. Doel

Dit document beschrijft hoe we de Drive App **architectonisch loskoppelen** van Fobis Core en van specifieke ERP-/planningssystemen (zoals Exact Globe en Odoo), door:

- Een **canoniek model** voor ritten/stops in Fobis Platform te gebruiken.
- Integratie met bronsystemen te regelen via **adapters** in de platformlaag.
- De Drive App uitsluitend tegen **platform-API's** te laten praten.

Doel: vandaag starten met Fobis als bron, maar het ontwerp zo maken dat **ieder plannings-/ERP-systeem kan aanhaken** zonder de app zelf te wijzigen.

---

2. Hoog-over architectuur

Logische keten:

1. **Planning/ERP-systeem**
2. Fobis Core, Exact Globe, Odoo, of een ander systeem dat ritten en stops plant.
3. **Fobis Platform – Integratielaag**
4. Canoniek rit-/stopmodel.
5. Adapters per bronsysteem (Fobis-adapter, Exact-adapter, Odoo-adapter, …).
6. API's en events richting apps.
7. **Drive App**
8. Praat alleen met platform-API's (`/api/drive/...`).
9. Kent geen details van Fobis, Exact, Odoo.

Belangrijk principe: **de Drive App weet niets van het onderliggende planningssysteem** – alle mapping gebeurt in de platformlaag.

---

3. Canoniek model in Fobis Platform

We definiëren in Fobis Platform een **eigen, systeem-onafhankelijk datamodel** voor ritten en stops, bijvoorbeeld:

- `Trip` (rit)
- `tripId` (platform-id)
- `externalId` (id in bronsysteem)
- `tenantId`
- `driverId`
- `plannedDate`
- `status` (planned, in_progress, completed, canceled)
- …
- `Stop`
- `stopId` (platform-id)
- `externalId` (id in bronsysteem)
- `tripId`
- `sequenceNumber`
- `customerName`
- `address`
- `timeWindow`
- `status` (not_started, en_route, delivered, not_delivered)
- `reasonCode` (bij niet-afgeleverd)
- …

Alle platform-API's gebruiken dit **canonieke model**.
Adapters zijn verantwoordelijk voor de vertaling tussen:

- Canoniek model ↔ Fobis Core.
- Canoniek model ↔ Exact Globe (bijv. verkooporders/ritplanning).
- Canoniek model ↔ Odoo (delivery orders, routes, enz.).

---

4. Integratiepatronen tussen platform en bronsystemen

We onderscheiden twee hoofdpatronen:

4.1 Pull (platform haalt ritten op)

- Integratieservice in Fobis Platform:
- Vraagt periodiek of op event-basis de geplande ritten op bij het bronsysteem:
- Via API's (REST/GraphQL) indien beschikbaar.
- Via database of bestand (alleen als noodzakelijk, bij legacy).
- Mapt de data naar het canonieke `Trip` / `Stop` -model.
- Slaat ritten op in de platform-database.
- Voordeel:
- Platform is leidend voor de app.
- App hoeft niet te wachten op live-connectie met het bronsysteem.

4.2 Push (bronsysteem stuurt ritten naar platform)

- Bronsysteem publiceert ritten/stops:
- Via webhooks, berichtenqueue (bijv. RabbitMQ, Azure Service Bus) of andere integratiemechanismen.
- Naar een endpoint of queue in Fobis Platform.
- Data wordt gemapt naar canoniek model en opgeslagen.

In beide gevallen zijn **adapters** verantwoordelijk voor de vertaling van en naar het canonieke model.

---

5. API's tussen Drive App en Fobis Platform

De Drive App gebruikt een set **stabiele platform-API's**, bijvoorbeeld:

- `GET /api/drive/trips?date=...`
  Haalt ritten voor ingelogde chauffeur op.
- `GET /api/drive/trips/{tripId}/stops`
  Haalt stops voor een rit op.
- `POST /api/drive/stops/{stopId}/complete`
  Meldt stop als afgeleverd.
- `POST /api/drive/stops/{stopId}/fail`
  Meldt stop als niet-afgeleverd (incl. reden, optionele foto/handtekening).

Deze API's werken **uitsluitend met het canonieke model**.
De platformservices zorgen ervoor dat:

- Statusupdates teruggeschreven worden naar het juiste bronsysteem via de juiste adapter.
- Eventueel meerdere bronsystemen per tenant ondersteund kunnen worden (bij complexe landschappen).

---

6. Adapters per plannings-/ERP-systeem

Voor elk ondersteund bronsysteem komt er een **aparte adapter**, bijvoorbeeld:

- `FobisAdapter`
- Kent de Fobis Core-datamodellen en API's.
- Mapt Fobis-ritten ↔ canoniek `Trip` / `Stop`.
- Schrijft afmeldingen/retours terug in Fobis.
- `ExactGlobeAdapter`
- Integreert met de wijze waarop Exact ritten/leveringen vertegenwoordigt (bijv. verkooporders + routes).
- Mapt Exact-data ↔ canoniek model.
- `OdooAdapter`
- Integreert met Odoo's delivery/route-modules.
- Mapt Odoo-data ↔ canoniek model.

Adapters volgen hetzelfde **interfacecontract**, bijvoorbeeld:

- `GetTripsForDriver(tenantId, driverId, date)`
- `UpdateStopStatus(tenantId, tripExternalId, stopExternalId, newStatus, reasonCode, extraData)`

Daardoor kan per tenant **geconfigureerd** worden welk adaptertype actief is, zonder de Drive App te wijzigen.

---

7. Tenantconfiguratie – kiezen van het bronsysteem

Per tenant (klant) wordt in Fobis Platform vastgelegd:

- Welk **bronsysteemtype** in gebruik is:
- `FOBIS_CORE`, `EXACT_GLOBE`, `ODOO`, `CUSTOM`, ...
- De benodigde connectiedetails:
- API-URL's, credentials, queue-namen, etc.

De integratieservice kiest op basis hiervan automatisch de juiste adapter voor:

- Ritten ophalen.
- Afmeldingen terugschrijven.

Zo kunnen:

- Sommige klanten puur op Fobis draaien.
- Andere klanten primair op Exact of Odoo plannen, met Fobis als "datahub".
- Later ook andere systemen worden aangesloten zonder impact op de app.

---

8. Event- en dataflow bij afmelden

Voorbeeldflow bij "stop afmelden" in Drive App:

1. Chauffeur meldt in Drive App stop `S123` als afgeleverd.
2. App stuurt `POST /api/drive/stops/{stopId}/complete` naar Fobis Platform.
3. Platform:
4. Valideert auth/licensing.
5. Zoekt bij `stopId` de bijbehorende `tenantId`, `tripId`, `externalIds`.
6. Schrijft statuswijziging in het canonieke model (interne database).
7. Roept de juiste adapter aan voor die tenant.
8. Adapter:
9. Vertaalt de statuswijziging naar een call/event richting het bronsysteem (Fobis/Exact/Odoo).
10. Handelt bronspecifieke logica af (bijv. updates van orderstatus, retourcodes).
11. Platform logt de actie (inclusief correlation-id) en geeft de app een bevestiging.

De app verandert **niet** als het bronsysteem verandert; alleen de adapter en tenantconfiguratie veranderen.

---

9. Architectonische voordelen

- **Losse koppeling**:
- Drive App kent alleen Fobis Platform-API's en het canonieke model.
- Bronsystemen zitten achter adapters.
- **Uitbreidbaarheid**:
- Nieuwe bronsystemen kunnen worden toegevoegd door nieuwe adapters te schrijven.
- **Herbruikbaarheid**:
- Andere apps kunnen hetzelfde canonieke model en dezelfde integratielaag hergebruiken.
- **Multi-systeem scenario's**:
- Mogelijk om, per tenant of zelfs per proces, verschillende bronsystemen te gebruiken.

---

10. Samenvatting

- Architectonisch bereiken we de gewenste onafhankelijkheid door:

- Een **canoniek rit-/stopmodel** in Fobis Platform.
- **Adapters** per plannings-/ERP-systeem.
- Een **Drive App** die uitsluitend met platform-API's praat.
- In de praktijk starten we met Fobis Core als bronsysteem, maar het ontwerp maakt het mogelijk om later **Exact, Odoo of andere systemen** aan te haken zonder de Drive App zelf aan te passen.

3.3 Organisatie

3.3.1 Organisatie en Ontwikkelteam(s)

Fobis Platform – Organisatie en Ontwikkelteam(s)

1. DOEL VAN DIT DOCUMENT

Dit document verkent de **organisatorische structuur** en **ontwikkelteam(s)** die nodig zijn voor het Fobis Platform. Het gaat om:

- Hoe ontwikkelteam(s) zijn georganiseerd (structuur, rollen, verantwoordelijkheden).
- Relatie tussen platform-ontwikkeling en app-ontwikkeling.
- Samenwerking met partners (zoals 4DotNet).
- Skills en expertise die nodig zijn.
- Werkwijze en processen (agile, scrum, etc.).

---

2. HUIDIGE SITUATIE

[Te vullen: beschrijving van de huidige organisatie rond Fobis-ontwikkeling]

---

3. VERKENNING: ORGANISATIEMODELLEN

3.1 Platform-team vs. App-teams

[Te vullen: verkenning van verschillende organisatiemodellen]

- Centraal platform-team dat de basis bouwt.
- App-teams die bovenop het platform werken.
- Hybride model met gedeelde verantwoordelijkheden.

3.2 Rollen en verantwoordelijkheden

[Te vullen: verkenning van benodigde rollen]

- Platform architect.
- Backend developers.
- Frontend developers.
- DevOps engineers.
- QA/test engineers.
- Product owners.

3.3 Samenwerking met partners

[Te vullen: verkenning van samenwerkingsmodellen met externe partners]

- Hoe werken we samen met 4DotNet?
- Welke delen ontwikkelen we zelf, welke delen extern?
- Hoe beheren we code ownership en IP?

---

4. SKILLS EN EXPERTISE

[Te vullen: verkenning van benodigde skills]

- Technische skills (C#, .NET, cloud, etc.).
- Domain knowledge (Fobis, foodproductie).
- Platform-ontwikkeling vs. applicatie-ontwikkeling.

---

5. WERKWIJZE EN PROCESSEN

[Te vullen: verkenning van ontwikkelprocessen]

- Agile/scrum aanpak.
- Code reviews en kwaliteitsprocessen.
- Release management.
- Documentatie en kennisbeheer.

---

6. UITDAGINGEN EN AANDACHTSPUNTEN

[Te vullen: verkenning van uitdagingen]

- Schaalbaarheid van het team.
- Kennisoverdracht en onboarding.
- Balans tussen platform en apps.
- Continuïteit en bus factor.

---

7. VERVOLGSTAPPEN

[Te vullen: wat zijn de volgende stappen in deze verkenning?]

3.3.2 Serviceteam

Fobis Platform – Serviceteam

1. DOEL VAN DIT DOCUMENT

Dit document verkent de **organisatie en rol van het serviceteam** voor Fobis Platform. Het gaat om:

- Verantwoordelijkheden van het serviceteam.
- Ondersteuning van verschillende deploymentmodellen (on-prem, hybride, SaaS).
- Relatie tussen serviceteam en ontwikkelteam(s).
- Skills en expertise die nodig zijn.
- Processen voor support, monitoring en incident management.

---

2. HUIDIGE SITUATIE

[Te vullen: beschrijving van de huidige serviceteam-organisatie rond Fobis]

---

3. VERKENNING: SERVICETEAM STRUCTUUR

3.1 Verantwoordelijkheden

[Te vullen: verkenning van serviceteam verantwoordelijkheden]

- Platform monitoring en observability.
- Incident response en troubleshooting.
- Klantondersteuning en support.
- Deployment en configuratiemanagement.
- Security en compliance.

3.2 Deploymentmodel-specifieke aspecten

[Te vullen: verkenning per deploymentmodel]

- **On-prem**: ondersteuning van klantinstallaties.
- **Klant-cloud**: samenwerking met klant-IT.
- **RBK SaaS**: 24/7 platformbeheer.

3.3 Relatie met ontwikkelteam(s)

[Te vullen: verkenning van samenwerking]

- Hoe werkt serviceteam samen met ontwikkelteam?
- Escalatieprocessen.
- Kennisoverdracht en documentatie.

---

4. SKILLS EN EXPERTISE

[Te vullen: verkenning van benodigde skills]

- Platform- en cloud-expertise.
- Troubleshooting en debugging.
- Klantcommunicatie.
- Monitoring en observability tools.
- Security en compliance kennis.

---

5. PROCESSEN EN PROCEDURES

[Te vullen: verkenning van serviceteam processen]

- Incident management.
- Change management.
- Monitoring en alerting.
- Runbooks en documentatie.
- SLA's en service levels.

---

6. TOOLS EN TOOLING

[Te vullen: verkenning van benodigde tools]

- Monitoring en observability platforms.
- Ticketing en incident management.
- Documentatie en kennisbeheer.
- Deployment en configuratiemanagement tools.

---

7. UITDAGINGEN EN AANDACHTSPUNTEN

[Te vullen: verkenning van uitdagingen]

- Schaalbaarheid bij groeiend aantal klanten.
- Complexiteit van verschillende deploymentmodellen.
- 24/7 beschikbaarheid voor SaaS.
- Kennisoverdracht en continuïteit.

---

8. VERVOLGSTAPPEN

[Te vullen: wat zijn de volgende stappen in deze verkenning?]

3.3.3 Projectmanagement

Fobis Platform – Projectmanagement

1. DOEL VAN DIT DOCUMENT

Dit document verkent de **projectmanagement-aanpak** voor Fobis Platform. Het gaat om:

- Projectmanagement structuur en rollen.
- Planning en roadmap management.
- Prioritering van features en initiatieven.
- Stakeholder management.
- Risicomanagement.
- Budget en resource planning.

---

2. HUIDIGE SITUATIE

[Te vullen: beschrijving van de huidige projectmanagement-aanpak rond Fobis]

---

3. VERKENNING: PROJECTMANAGEMENT STRUCTUUR

3.1 Rollen en verantwoordelijkheden

[Te vullen: verkenning van projectmanagement rollen]

- Product owner(s).
- Project manager(s).
- Program manager (voor platform als geheel).
- Stakeholder representatives.

3.2 Planning en roadmap

[Te vullen: verkenning van planningprocessen]

- Roadmap planning (kwartaal, jaar).
- Sprint/release planning.
- Feature prioritering.
- Dependency management tussen platform en apps.

3.3 Stakeholder management

[Te vullen: verkenning van stakeholder management]

- Wie zijn de belangrijkste stakeholders?
- Hoe communiceren we met verschillende stakeholders?
- Besluitvormingsprocessen.

---

4. METHODOLOGIE EN AANPAK

[Te vullen: verkenning van projectmanagement methodologie]

- Agile/scrum voor ontwikkeling.
- Waterfall voor strategische planning?
- Hybride aanpak?
- Portfolio management.

---

5. PRIORITERING EN BESLUITVORMING

[Te vullen: verkenning van prioriteringsprocessen]

- Hoe prioriteren we platform vs. app features?
- Besluitvormingscriteria.
- Trade-offs en afwegingen.

---

6. RISICOMANAGEMENT

[Te vullen: verkenning van risicomanagement]

- Technische risico's.
- Organisatorische risico's.
- Markt- en business risico's.
- Mitigatiestrategieën.

---

7. BUDGET EN RESOURCES

[Te vullen: verkenning van budget en resource planning]

- Budgetallocatie (platform vs. apps).
- Resource planning.
- Externe partners en contractors.

---

8. UITDAGINGEN EN AANDACHTSPUNTEN

[Te vullen: verkenning van uitdagingen]

- Balans tussen platform en apps.
- Lange-termijn visie vs. korte-termijn behoeften.
- Stakeholder alignment.
- Resource constraints.

---

9. VERVOLGSTAPPEN

[Te vullen: wat zijn de volgende stappen in deze verkenning?]

3.3.4 Marketing organisatie

Fobis Platform – Marketing organisatie

1. DOEL VAN DIT DOCUMENT

Dit document verkent de **marketing organisatie** rond Fobis Platform. Het gaat om:

- Marketing structuur en rollen.
- Marketing strategie en positionering.
- Communicatie naar verschillende doelgroepen.
- Marketing activiteiten en kanalen.
- Relatie tussen marketing en product development.
- Metrics en success criteria.

---

2. HUIDIGE SITUATIE

[Te vullen: beschrijving van de huidige marketing organisatie rond Fobis]

---

3. VERKENNING: MARKETING STRUCTUUR

3.1 Rollen en verantwoordelijkheden

[Te vullen: verkenning van marketing rollen]

- Marketing manager.
- Product marketing.
- Content marketing.
- Digital marketing.
- Events en partnerships.

3.2 Marketing strategie

[Te vullen: verkenning van marketing strategie]

- Positionering van Fobis Platform.
- Merkverhaal en messaging (zie ook: FOBIS Next - Merkverhaal).
- Doelgroepen en personas.
- Unique selling points.

3.3 Communicatie naar doelgroepen

[Te vullen: verkenning van doelgroepcommunicatie]

- Bestaande Fobis-klanten.
- Potentiële nieuwe klanten.
- Partners (zoals 4DotNet).
- Interne stakeholders.

---

4. MARKETING ACTIVITEITEN EN KANALEN

[Te vullen: verkenning van marketing activiteiten]

- Website en online presence.
- Content marketing (blog, whitepapers, case studies).
- Events en webinars.
- Social media.
- Partner marketing.
- Sales enablement.

5. RELATIE MET PRODUCT DEVELOPMENT

[Te vullen: verkenning van samenwerking]

- Hoe werkt marketing samen met product development?
- Feature announcements en releases.
- Customer feedback loops.
- Go-to-market strategieën.

6. MARKETING VOOR VERSCHILLENDE DEPLOYMENTMODELLEN

[Te vullen: verkenning per deploymentmodel]

- Marketing voor on-prem klanten.
- Marketing voor hybride/cloud klanten.
- Marketing voor SaaS-aanbod.
- Verschillende messaging per model?

7. METRICS EN SUCCESS CRITERIA

[Te vullen: verkenning van marketing metrics]

- Awareness metrics.
- Lead generation.
- Conversion metrics.
- Customer acquisition cost.
- Brand perception.

8. UITDAGINGEN EN AANDACHTSPUNTEN

[Te vullen: verkenning van uitdagingen]

- Positionering van platform vs. product.
- Communicatie van technische complexiteit.
- Budget en resource constraints.
- Alignment met product roadmap.

9. VERVOLGSTAPPEN

[Te vullen: wat zijn de volgende stappen in deze verkenning?]

3.3.5 Licensering en Verdienmodel

Fobis Platform – Licensering en Verdienmodel

1. DOEL VAN DIT DOCUMENT

Dit document verkent de **licensering** en **verdienmodel** voor Fobis Platform. Het gaat om:

- Licensemodellen voor platform en apps.
- Pricing strategieën.
- Verdienmodellen per deployment model (on-prem, hybride, SaaS).
- Relatie tussen licensing service (technisch) en business model.
- Revenue streams en business case.

---

2. HUIDIGE SITUATIE

[Te vullen: beschrijving van het huidige Fobis-licensemodel en verdienmodel]

---

3. VERKENNING: LICENSEMODELLEN

3.1 Platform licensing

[Te vullen: verkenning van platform licensing]

- Basis platform licentie.
- Per-tenant licensing.
- Feature-based licensing (modules).
- Usage-based licensing.

3.2 App licensing

[Te vullen: verkenning van app licensing]

- Drive App licensing.
- Toekomstige apps.
- Per-app vs. bundled licensing.
- Partner apps en revenue sharing.

3.3 Licensing per deployment model

[Te vullen: verkenning per deployment model]

- **On-prem**: traditionele licentieverkoop?
- **Klant-cloud**: cloud licensing model?
- **RBK SaaS**: subscription model?

---

4. PRICING STRATEGIEËN

[Te vullen: verkenning van pricing strategieën]

- One-time vs. recurring revenue.
- Subscription tiers.
- Usage-based pricing.
- Enterprise vs. SMB pricing.

4.1 Pricing per deploymentmodel

[Te vullen: verkenning van pricing per model]

- On-prem: eenmalige licentie + support?
- Klant-cloud: subscription + hosting?
- RBK SaaS: per-tenant subscription?

---

5. REVENUE STREAMS

[Te vullen: verkenning van revenue streams]

- Platform licensing.
- App licensing.
- Support en maintenance.
- Professional services.
- Hosting en infrastructure (voor SaaS).
- Partner revenue sharing.

---

6. BUSINESS CASE EN FINANCIËLE MODELLEN

[Te vullen: verkenning van business case]

- Kostenstructuur (ontwikkeling, operatie, support).
- Revenue projections.
- Break-even analyse.
- ROI voor verschillende deploymentmodellen.

---

7. TECHNISCHE VS. BUSINESS LICENSING

[Te vullen: verkenning van relatie tussen technisch en business]

- Hoe sluit de Licensing Service Architectuur (technisch) aan op business modellen?
- Feature flags en licensing enforcement.
- Reporting en analytics voor licensing.

---

8. CONCURRENTIE EN MARKTPOSITIE

[Te vullen: verkenning van concurrentie]

- Hoe verhouden we ons tot concurrenten?
- Marktprijzen en -modellen.
- Differentiatie via pricing?

---

9. UITDAGINGEN EN AANDACHTSPUNTEN

[Te vullen: verkenning van uitdagingen]

- Transitie van huidige naar nieuwe modellen.
- Klantacceptatie van nieuwe pricing.
- Complexiteit van verschillende modellen.
- Revenue recognition en accounting.

---

10. VERVOLGSTAPPEN

[Te vullen: wat zijn de volgende stappen in deze verkenning?]

4. Architectuur

4.1 Now

4.1.1 Fobis

Fobis is an ERP/MES application written in C# on top of .NET Core.

Documentation

- See **Documentation** folder for the available documentation.
- Run `ServeDocumentation.bat` for a web version of the documentation.
- To update the documentation use **MkDocs** (see the documentation in the main FOBIS repository).

Dependencies

The projects use DevExpress for the desktop and client applications on the user-facing side, ASP.NET on the back-end side and EntityFramework to communicate with the database.

Components

Fobis can be roughly divided into the following functional components:

- Desktop
- Services
- Devices

And a variety of tools/libraries.

How to deploy locally

PREREQUISITES

- An MSSQL database.
- Restore SQL/<$currentversion>/BasePoint.zip for a starting position.
- A license file.
- Can be requested through marketing, or generated using the `Desktop/RBK.Fobis.LicenseGenerator` tool.

SOFTWARE NEEDED

- Visual Studio or other IDE.
- DevExpress
- NuGet

CONFIGURATION NEEDED

For the services:
- Manage user secrets and configure ConnectionStrings to include your FobisConnection. Example:

`json usersecrets.json`

```
{
  "ConnectionStrings": {
    "FobisConnection": "Server=.\\LOCAL01;persist security info=True;user id=sa;password=p@ssw0rd;",
    "GeCoConnection": "Server=.\\BASEPOINT;persist security info=True;user id=sa;password=p@ssw0rd;",
    "DefaultConnection": "Server=.\\LOCAL01;Database=prdPA;persist security info=True;user id=sa;password=p@ssw0rd;",
    "LoggingConnection": "Server=.\\LOCAL01;Database=prdLog;persist security info=True;user id=sa;password=p@ssw0rd;",
```

```
    "FobisBIConnection": "Server=.\\LOCAL01;Database=FobisBI;persist security info=True;user id=sa;password=p@ssw0rd;",
    "SUConnection": "Server=.\\LOCAL01;Database=prdSU;persist security info=True;user id=sa;password=p@ssw0rd;",
    "IEConnection": "Server=.\\LOCAL01;Database=prdIE;persist security info=True;user id=sa;password=p@ssw0rd;"
  }
}
```

BUILD & RUN

This section is limited to the main components. Because we override HTTP.SYS settings, run the services as admin.

Manual:

- `dotnet run .\Services\RBK.Fobis.API.BOService\RBK.Fobis.API.BOService.csproj`
- `dotnet run .\Services\RBK.Fobis.API.UIService\RBK.Fobis.API.UIService.csproj`
- `dotnet run .\Desktop\RBK.Fobis.WindowsClient\RBK.Fobis.WindowsClient.csproj`

Through Visual Studio:

- Services:
- At the bare minimum, set the following as startup project:
- RBK.Fobis.UIService
- RBK.Fobis.BOService
- Client:
- Set as startup project:
- RBK.Fobis.WindowsClient

Important:
To run the applications you also need the `RBK.Fobis.LicenseService`, however that application has become version-independant and as such is part of the `FOBIS-Utilities` repository.
This application (together with the version-independant `RBK.Fobis.MailService`) will be deployed by the Deploy pipeline and will be packaged in the FOBIS installer.

Deploying on-premise

Done through the installer by the ServiceDesk department.

4.1.2 FOBIS Libraries

We'll be using this repository to store our own libraries that we want to share between applications from different repositories.

FOBIS Development vs FOBIS Libraries

Libraries within this repository should be version independent from the regular FOBIS applications which can be found in the FOBIS-Development repository.

IMPORTANT
Projects in the FOBIS Development repository can use the FOBIS libraries, not the other way around!

4.1.3 Repository structure

In the root you'll find general stuff like:
- Editor config
- NuGet config
- Git ignore
- Library Solution file

Next to the general stuff will be customer specific folders which will contain the source code for the libraries, build scripts, documentation, etc.

Naming rules

We want consistancy in the naming of our libraries and namespaces.

We'll apply the following naming scheme:
- Solution: RBK.FOBIS.Libraries
- Application: RBK.FOBIS.<Library>
- Namespace: RBK.FOBIS.<Library>.***

Examples:
- Application: RBK.FOBIS.Communication
- Namespace: RBK.FOBIS.Communication.Network

Library versioning

For libraries within this repository we'll be using a different versioning strategy then regular FOBIS. We'll be using Semantic Versioning for the libraries within this repository. The implication is that each library will have a different version number.

Most important reason to differentiate from regular FOBIS is to specifically imply that the version are (and should be) unrelated.

4.1.4 Build automation

For every library there should be build scripts available and a pipeline configured. We should be able to push changes, a pipeline should automatically trigger and a build should be made available.

Whats New

Just like with every build for regular FOBIS, a Whats New file should be deployed for libraries as well. We need to be able to communicate which changes are included in the build.

4.1.5 Local Development

Our libraries will be packed and published as a NuGet package and be available via our Azure DevOps feed. For local development however this poses a difficulty, you don't want to release new versions while developing. The solution is working with a local NuGet feed and publishing the libraries to that feed.

We've added `publish-to-local-feed.ps1` scripts to the different projects which automates this proces. In the root of this repository there is a `local-feed` folder where the packages will be hosted from. In our other repositories we've added an additional NuGet source to points to this local feed folder. Because not all computers are alike, it works with an environment variable called `FOBIS_Libs_LocalFeed` which should point to this folder. To clear the `local-feed` folder of previously build packages, you can use the `clear-local-feed.ps1` script which is found in the root of the repository.

Steps when developing a new addition:
1. Make changes
2. Increase version number and add modifier, for instance: `1.1.1-rc1`
3. Run the `publish-to-local-feed.ps1` from the folder of the project
4. Update your project or `Directory.Packages.props` and use the new local version
5. Now you can use the new version in your project!

4.1.6 FOBIS-Pipelines

This repository contains standardized Azure DevOps pipeline templates for building, testing, and deploying FOBIS services and NuGet packages.

Repository Structure

```
FOBIS-Pipelines/
├── buildAndTest/             # Build and test pipeline definitions
│   ├── jobs/                 # Individual job templates for building and testing
│   │   ├── build.yml         # Defines the build process
│   │   └── test.yml          # Defines the test process
│   └── stages/
│       └── buildAndTest.yml  # Combines build and test jobs into a stage
├── config/
│   └── gitVersion.yml        # GitVersion configuration for semantic versioning
├── release/                  # Release pipeline definitions
│   ├── jobs/
│   │   ├── createInstaller.yml   # Handles installer creation
│   │   ├── deploy.yml            # Handles service deployment
│   │   └── nuget.yml             # Handles NuGet package publishing
│   └── stages/
│       ├── createInstaller.yml   # Defines the installer creation stage
│       ├── deploy.yml            # Defines the deployment stage
│       └── nuget.yml             # Defines the NuGet publishing stage
├── nuget-package.yml         # Main pipeline for NuGet package workflows
├── service.yml               # Main pipeline for service application workflows
├── variables.yml             # Common pipeline variables
└── variables-environment.yml # Environment-specific variables
```

Main Pipelines

SERVICE PIPELINE (`service.yml`)

Used for building, testing, and deploying service applications. This pipeline:

1. Builds .NET applications with specified SDK version
2. Runs automated tests
3. Creates deployment artifacts
4. Optionally deploys to the specified environment (when `deploy` is true, triggered manually and `createInstaller` is set to "None")
5. Creates installers when:
6. `createInstaller` is set to "Deploy" or "Release", OR
7. `createInstaller` is "None" but the pipeline runs automatically on the main branch (not manual or pull request triggers)

NUGET PACKAGE PIPELINE (`nuget-package.yml`)

Used for building, testing, and publishing NuGet packages. This pipeline:

1. Builds .NET library projects with specified SDK version
2. Runs automated tests
3. Uses GitVersion for semantic versioning
4. Creates NuGet packages
5. Publishes packages to the specified NuGet feed (except for pull requests)

Parameters

COMMON PARAMETERS

Parameter	Description	Default	Required	Values
`dotNetSdkVersion`	Version of .NET SDK to use	9.x	No	-
`nugetConfigPath`	Path to NuGet configuration file	$(sourcesDirectory)/NuGet.config	No	-
`nugetVersion`	Version of NuGet tool to use	x (latest)	No	-
`outputPath`	Location for build outputs	$(Build.ArtifactStagingDirectory)/output	No	-
`projectFiles`	Project files to build (glob pattern)	-	Yes	-
`shortName`	Short project name	-	Yes	-
`testProjectFiles`	Test project files to run (glob pattern)	-	Yes	-

SERVICE-SPECIFIC PARAMETERS

Parameter	Description	Default	Required	Values
`createInstaller`	Controls installer creation	-	Yes	None, Deploy, Release
`deploy`	Whether to include deployment stage	-	Yes	-
`installerSourcesPath`	Path to installer source directory	-	Yes	-
`runtime`	.NET runtime target	win-x64	No	-

NUGET-SPECIFIC PARAMETERS

Parameter	Description	Default	Required	Values
`nugetFeedName`	Name of the NuGet feed	rbk-fobis	No	-

GitVersion Configuration

This repository uses GitVersion for automatic semantic versioning based on Git history. The configuration follows GitHub Flow with automatic version calculation based on commits and tags.

MANUAL VERSION CONTROL

You can manually control version increments using specific commit message tags:

- `+{ServiceName}-semver: major` - Increments the major version (e.g., 1.0.0 → 2.0.0)
- `+{ServiceName}-semver: minor` - Increments the minor version (e.g., 1.0.0 → 1.1.0)
- `+{ServiceName}-semver: patch` - Increments the patch version (e.g., 1.0.0 → 1.0.1)
- `+{ServiceName}-semver: none` - Prevents version increment for this commit

`{ServiceName}` should be equal to the `shortName` as defined in the pipeline variables. For example, if your service is named `RBK.Fobis.MyService`, the `shortName` would be `MyService`, and you would use `+MyService-semver: major` to increment the major version.

Examples:

```
git commit -m "Complete redesign of API +MyService-semver: major"
git commit -m "Add new endpoint for user profiles +MyService-semver: minor"
git commit -m "Fix sorting bug in dashboard +MyService-semver: patch"
```

These commit message tags will be detected by GitVersion during the build process and will affect how the next version number is calculated.

Variables

COMMON VARIABLES (`variables.yml`)

The following variables are available for use in all pipelines:

Variable Name	Value	Description
`poolName`	FOBIS-Development	The agent pool used for pipeline execution
`sourcesDirectory`	$(Pipeline.Workspace)/s/Sources	Base directory where source code is located

ENVIRONMENT-SPECIFIC VARIABLES (`variables-environment.yml`)

This template provides environment-specific variables based on the deployment target:

Parameter	Type	Description
`environment`	string	Target environment (deploy/release)
`shortName`	string	Project name

Variable Name	Value (deploy environment)	Value (release environment)	Description
`networkPath`	`\\vs-rbk-fs01\Groups\FobisPS\Fobis.NET\Builds\{shortName}\`	`\\vs-rbk-fs01\Algemeen\FobisPS\Releases\{shortName}\`	Network path for deployment artifacts

Usage

To use these pipelines in your project:

1. Reference the FOBIS-Pipelines repository in your Azure DevOps pipeline
2. Include the appropriate template (service.yml or nuget-package.yml)
3. Provide required parameters specific to your project

EXAMPLE FOR A SERVICE

```yaml
parameters:
  - name: 'deploy'
    displayName: 'Deploy files (gets overwritten if Installer location is set to Deploy)'
    type: boolean
    default: true
  - name: 'createInstaller'
    displayName: 'Installer location'
    type: string
    default: 'None'
    values:
    - None
    - Deploy
    - Release
  - name: 'pipelineRepositoryBranch'
    displayName: 'FOBIS-Pipelines Branch'
    type: string
    default: 'main'

trigger:
  branches:
    include:
      - main
  paths: # Edit these paths to match your project
    include:
      - RBK.Fobis.MyService
      - RBK.Fobis.MyService.Tests
      - RBK.Fobis.Services
      - RBK.Fobis.TestUtilities
```

```yaml
variables:
  # Edit these variables to match your project
  - name: 'fullName'
    value: 'RBK.Fobis.MyService'
  - name: 'shortName'
    value: 'MyService'
  - name: 'dotNetSdkVersion'
    value: '9.x'

  # You should not need to change these variables
  - name: 'installerDirectory'
    value: '$(sourcesDirectory)/$(fullName)/Installer'
  - name: 'projectFiles'
    value: '$(sourcesDirectory)/$(fullName)/$(fullName).csproj'
  - name: 'testProjectFiles'
    value: '$(sourcesDirectory)/$(fullName).Tests/$(fullName).Tests.csproj'
  - template: variables.yml@FOBIS-Pipelines

resources:
  repositories:
    - repository: FOBIS-Pipelines
      type: git
      name: FOBIS-Pipelines
      ref: 'refs/heads/${{ parameters.pipelineRepositoryBranch }}'

stages:
  - template: service.yml@FOBIS-Pipelines
    parameters:
      createInstaller: ${{ parameters.createInstaller }}
      deploy: ${{ parameters.deploy }}
      dotNetSdkVersion: '${{ variables.dotNetSdkVersion }}'
      installerSourcesPath: '${{ variables.installerDirectory }}'
      projectFiles: '${{ variables.projectFiles }}'
      shortName: '${{ variables.shortName }}'
      testProjectFiles: '${{ variables.testProjectFiles }}'
```

EXAMPLE FOR A NUGET PACKAGE

```yaml
parameters:
  - name: 'pipelineRepositoryBranch'
    displayName: 'FOBIS-Pipelines Branch'
    type: string
    default: 'main'

trigger:
  branches:
    include:
      - main
  paths: # Edit these paths to match your project
    include:
      - RBK.FOBIS.MyPackage
      - RBK.FOBIS.MyPackage.Tests

variables:
  # Edit these variables to match your project
  - name: 'fullName'
    value: 'RBK.FOBIS.MyPackage'
  - name: 'shortName'
    value: 'MyPackage'
  - name: 'dotNetSdkVersion'
    value: '9.x'

  # You should not need to change these variables
  - name: 'projectFiles'
    value: '$(sourcesDirectory)/$(fullName)/$(fullName).csproj'
  - name: 'testProjectFiles'
    value: '$(sourcesDirectory)/$(fullName).Tests/$(fullName).Tests.csproj'
  - template: variables.yml@FOBIS-Pipelines

resources:
  repositories:
    - repository: FOBIS-Pipelines
      type: git
      name: FOBIS-Pipelines
      ref: 'refs/heads/${{ parameters.pipelineRepositoryBranch }}'

stages:
  - template: nuget-package.yml@FOBIS-Pipelines
    parameters:
      dotNetSdkVersion: '${{ variables.dotNetSdkVersion }}'
      projectFiles: '${{ variables.projectFiles }}'
      shortName: '${{ variables.shortName }}'
      testProjectFiles: '${{ variables.testProjectFiles }}'
```

4.1.7 FOBIS Utilities

We'll be using this repository to store our own utility applications that are not bound to a particular version of FOBIS.

Examples of applications that classify as utilities are:
- Standalone applications (like FOBIS Launcher, FOBIS Licensing and FOBIS Mailing)
- Test tools

FOBIS Development vs FOBIS Utilities

Applications within this repository should be version independent from the regular FOBIS applications which can be found in the FOBIS-Development repository.

4.1.8 Repository structure

In the root you'll find general stuff like:
- Editor config
- NuGet config
- Git ignore
- Utilities Solution file

Next to the general stuff will be application specific folders which will contain the source code, build scripts, documentation, etc.

Naming rules

We want consistancy in the naming of our tooling and namespaces.

We'll apply the following naming scheme:
- Solution: RBK.FOBIS.Utilities
- Application: RBK.FOBIS.<Utility>
- Namespace: RBK.FOBIS.<Utility>.***

Examples:
- Application: RBK.FOBIS.Launcher
- Namespace: RBK.FOBIS.Launcher.Models

Utility versioning

For utilities within this repository we'll be using a different versioning strategy then regular FOBIS. We'll be using Semantic Versioning for the applications within this repository. The implication is that each application will have a different version number.

Most important reason to differentiate from regular FOBIS is to specifically imply that the version are (and should be) unrelated.

4.1.9 Build automation

For every utility application there should be build scripts available and a pipeline configured. We should be able to push changes, a pipeline should automatically trigger and a build should be made available.

Whats New

Just like with every build for regular FOBIS, a Whats New file should be deployed for utilities as well. We need to be able to communicate which changes are included in the build.

4.1.10 FOBIS Verkoop App

Setup

DEVELOPMENT

1. Clone the repository
2. Open the solution in visual studio
3. It should show a warning to install the blazor wasm library or something similarly named. If it doesn't, manually install the *ASP.NET and web development* workload by going to `Tools>Get Tools and Features` in Visual Studio.
4. Setup the connection string,

   To select a database add the relevant connection string to the user secrets by right-clicking the `RBK.Fobis.VerkoopApp.Service` project in visual-studio. The program selects ConnectionStrings:VerkoopAppService as the connection string via `webBuilder.Configuration.GetConnectionString("VerkoopAppService");`, see below for an example of said connection string in a secrets.json file. If sqlite is desired set DB:UseSqlite to true and use the connection string `SQLite_VerkoopAppService` instead of the sqlserver variant: `VerkoopAppService` Also make sure you set the JWT Issuer, Audience and Key to the appropriate options. The JWT Key needs to have a length of >= 40 characters and should be randomly generated for production use.

```
{
    {
    "ConnectionStrings": {
        "VerkoopAppService": "Data Source=PC-21013;Initial Catalog=DEV_VerkoopApp;Integrated Security=True;Connect Timeout=30;Encrypt=True;Trust Server Certificate=True;Application
Intent=ReadWrite;Multi Subnet Failover=False"
    },
    "Jwt": {
        "Issuer": "https://localhost:7279/",
        "Audience": "https://localhost:7279/",
        "Key": "WqSe33yTrSwQki3PleFDLccVTmcUJXeyAuKEJ1xMthtgKBhPLOb47guOxL59paFoQau4vqJuninUqsOdZEQqilkSZ9pGD8SW5PVGxXeWeRYhiygd8rH8MYeOurkwTT93OR",
        "ExpirySeconds":  604800
    },
}
```

5. Run the app via the `https` profile, the https-swagger profile will work too but requires you to manually navigate to the index to use the app.
6. Copy the `MasterData.json` file from `docs\test-data\MasterData.json` to the `FOBIS_EXPORT` two directories above where the executable is located (probably RBK.Fobis.VerkoopApp.Service\bin\FOBIS_EXPORT)
7. Copy the `StockData.json`, `UserData.json` & `CustomerData.json` to the `FOBIS_EXPORT` directory
8. Login to the app with username `rbkvka` and password `rbk`.

   For testing it is recommended to use google chrome because the serviceworker console is more easily accessible It als has better developer tools for background-sync and allows the app to run normally when requesting via http to localhost.

   For a slightly faster dev-website install runtime relinking using the following command:

```
dotnet workload install wasm-tools
```

   When creating a new js or css file don't forget to add it to the bundle in the main Program.cs file

Structure

```
@startuml "General architecture diagram"

package Serverside {
    agent FOBIS as fb
    folder "shared\nfilesystem" as fs
    agent Webserver as ws
    database "Database" as db
}
package Clientside {
    agent Serviceworker as sw
    database "IndexedDB & CacheStorage" as idb
    idb <-right-> sw
    agent "Browser tab" as bt
    bt <--> sw
}

sw <--> ws : HTTPS
ws <-down-> db
ws <-left-> fs
fb <-right-> fs
```

```
@enduml
```

This repo consists of two projects a client part which is a Blazor WebAssembly app and a server part which is a asp.net core app. The server provides all api endpoints and serves the client app. The client app makes use of the api on the serverside via a serviceworker which does caching and storage for offline-use.

Editor extensions

To properly preview these markdown files no extensions should be required. A plantuml vscode extension is reccommended though so they can be regenerated at will from the sourcecode. These should be in the Recommended list of the plugin sidebar and sometimes vscode will ask you to install them. For future reference they where at the time of writing: jebbs.plantuml, editorconfig.editorconfig, davidanson.vscode-markdownlint & pdconsec.vscode-print. `pdconsec.vscode-print` is useful for generating pdf files for distribution elsewhere.

If you want to export all diagrams in the project: In VSCode

1. Open the FOBIS-Verkoop-App folder and press F1.
2. Look for *>PlantUML: Export Workspace Diagrams*
3. Press enter on it, if it asks for a format choose png, if it asks for a location use `docs/diagrams`. Both these options should be chosen automatically though.

At the time of writing I'm using plantuml-1.2024.5 locally but the server should work too.

4.1.11 FOBIS Customs

We'll be using this repository to store tailor-made customer software.

FOBIS Development vs FOBIS Customs

Applications within this repository should be version independent from the regular FOBIS applications which can be found in the FOBIS-Development repository.

IMPORTANT
It should be possible to update a customer specific application without updating the regular FOBIS software and vice-versa.

4.1.12 Repository structure

In the root you'll find general stuff like:
- Editor config
- NuGet config
- Git ignore

Next to the general stuff will be customer specific folders which will contain the source code for the applications, solution files, build scripts, installers, etc.

Naming rules

While the applications are mostly unrelated (except possibly if there are multiple custom applications for one customer), we do want consistancy in the naming of our applications and namespaces.

We'll apply the following naming scheme:
- Solution: RBK.FOBIS.<Customer>
- Application: RBK.FOBIS.<Customer>.<Application>
- Namespace: RBK.FOBIS.<Customer>.<Application>.***

Examples:
- Solution: RBK.FOBIS.Compaxo
- Application: RBK.FOBIS.Compaxo.HammenSorteerLijn
- Namespace: RBK.FOBIS.Compaxo.HammenSorteerLijn.Models

Application versioning

For applications within this repository we'll be using a different versioning strategy then regular FOBIS. We'll be using Semantic Versioning for the applications within this repository. The implication is that each application will have a different version number.

Most important reason to differentiate from regular FOBIS is to specifically imply that the version are (and should be) unrelated.

4.1.13 Build automation

For every customer there should be build scripts available and a pipeline configured. We should be able to push changes, a pipeline should automatically trigger and a build should be made available.

Whats New

Just like with every build for regular FOBIS, a Whats New file should be deployed for customer specific applications as well. We need to be able to communicate which changes are included in the build.

5. Marketing

5.1 FOBIS Next – Merkverhaal

5.1.1 1. Kernboodschap

FOBIS Next is het moderne, gedistribueerde serviceframework voor de volgende generatie van FOBIS-oplossingen.
Het helpt organisaties om hun bestaande FOBIS-landschap gecontroleerd te moderniseren, stap voor stap te migreren en nieuwe digitale diensten sneller te ontwikkelen.

5.1.2 2. Elevator pitch (kort)

FOBIS Next brengt de kracht en betrouwbaarheid van FOBIS naar een toekomstbestendige, gedistribueerde architectuur.
Met een modulair serviceframework, duidelijke integratiepunten en een sterke focus op beheerbaarheid maakt FOBIS Next het mogelijk om nieuwe functionaliteit veilig naast bestaande systemen te introduceren en geleidelijk door te groeien naar een moderne cloud- en servicegerichte omgeving.

5.1.3 3. Uitgebreide beschrijving

FOBIS is al jarenlang een vertrouwd fundament onder kritische processen. Tegelijkertijd veranderen de eisen aan IT-landschappen: microservices, cloud-native, schaalbaarheid, veiligheid, continue levering en snelle innovatie zijn de nieuwe standaard.

FOBIS Next is ontwikkeld als antwoord op die beweging.
Het is geen compleet nieuw systeem naast FOBIS, maar een **distributed service framework** dat:

- **FOBIS centraal houdt** als bron van waarheid, waar dat nodig is.
- **Nieuwe diensten en componenten** op een moderne manier ontsluit en met FOBIS verbindt.
- **Gefaseerde migratie** mogelijk maakt, zodat organisaties niet in één keer hoeven te “big-bangen”.

Zo ontstaat een **brug tussen het bewezen bestaande en het innovatieve nieuwe** – met behoud van stabiliteit én ruimte voor vernieuwing.

5.1.4 4. Positionering

- **Voor wie**
- Organisaties die al op FOBIS draaien en hun landschap willen moderniseren.
- IT- en architectuurteams die naar een gedistribueerde, servicegerichte architectuur willen groeien zonder risico’s op continuïteit.
- **Probleem**
- Monolithische applicaties zijn lastig te veranderen, moeilijk schaalbaar en vormen een rem op innovatie.
- Volledige vervanging (replatforming) is risicovol, duur en organisatorisch complex.
- **Oplossing (FOBIS Next)**
- Biedt een consistent serviceframework rondom FOBIS voor nieuwe en bestaande functionaliteit.
- Maakt het mogelijk om stap voor stap services los te trekken, te moderniseren en gecontroleerd uit te rollen.
- Ondersteunt distributed deployment, integratie en monitoring.
- **Belofte**
- **“Moderniseer in je eigen tempo, zonder de zekerheid van FOBIS op te geven.”**

5.1.5 5. Belangrijkste merkwaarden

- **Betrouwbaar** – sluit aan op het bewezen FOBIS-fundament; stabiliteit en voorspelbaarheid staan voorop.
- **Toekomstgericht** – ontworpen voor de volgende generatie architecturen en technologieën.
- **Controleerbaar** – gefaseerde migratie, duidelijke governance en inzicht in services en afhankelijkheden.
- **Samenwerkend** – legt verbinding tussen teams, systemen en domeinen; stimuleert gedeelde verantwoordelijkheid.

• **Pragmatisch** – geen “big bang”-visie, maar praktische stappen die aansluiten op de werkelijkheid van vandaag.

5.1.6 6. Verhaal achter de naam en het logo

• **Naam: “FOBIS Next”**

• “FOBIS” borgt de herkenbaarheid en continuïteit met het bestaande platform.

• “Next” vertelt het verhaal van vooruitgang, evolutie en de volgende stap: de volgende generatie van FOBIS-oplossingen.

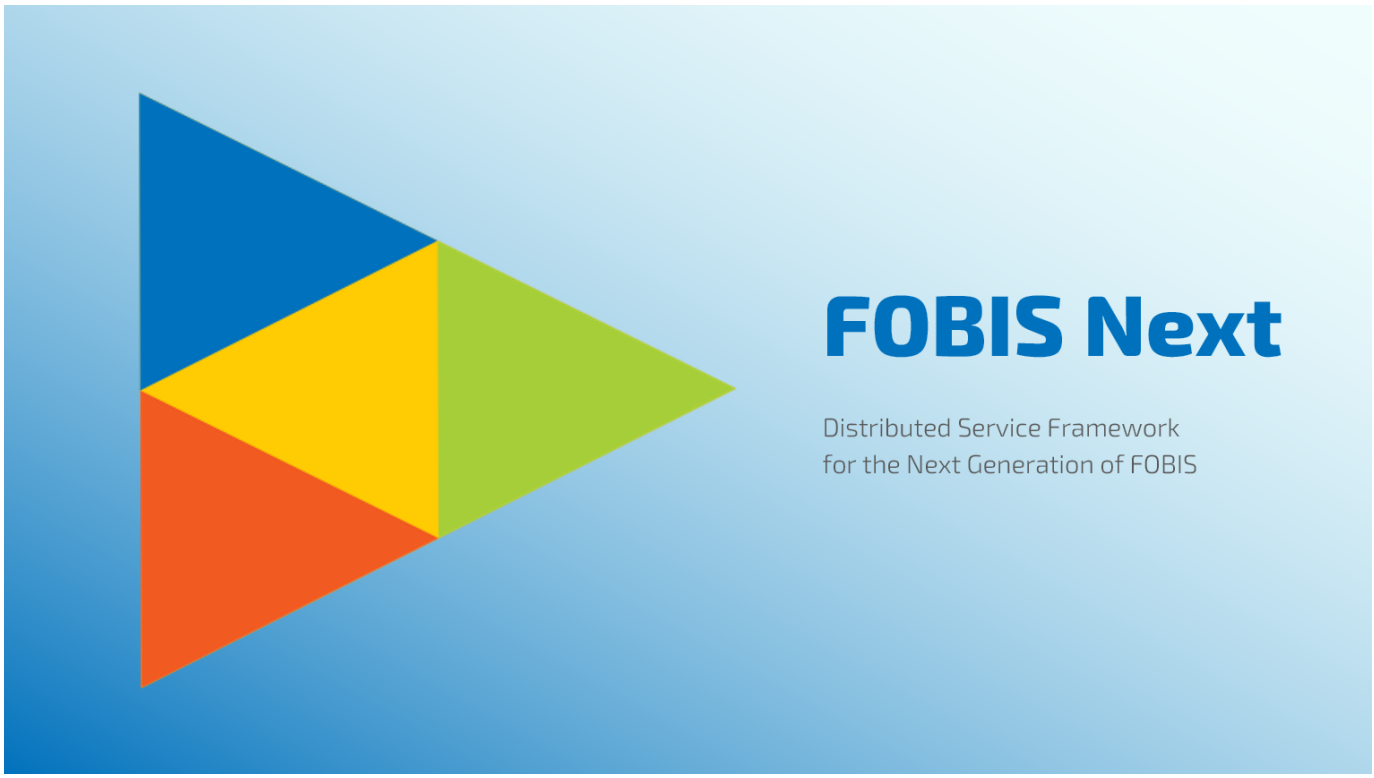
• **Logo**

• De **driehoek-vorm naar rechts** verwijst naar “play”, vooruitgang en beweging richting de toekomst.

• De **meerdere kleurvlakken** staan voor verschillende services/domeinen die samen één geheel vormen binnen FOBIS Next.

• De **heldere, moderne kleuren** geven een eigentijdse, technologische uitstraling zonder afstandelijk te worden.

Visueel ziet dat er bijvoorbeeld zo uit:



5.1.7 7. Tone of voice (voor teksten en presentaties)

- **Helder en concreet** – weinig jargon, maar wel technische diepgang waar nodig.
- **Vertrouwend en zeker** – benadruk stabiliteit en ervaring, geen hype.
- **Constructief en uitnodigend** – nodig mensen uit om mee te denken en stap voor stap mee te bewegen.
- **Oplossingsgericht** – leg de nadruk op wat FOBIS Next mogelijk maakt, niet alleen op wat er mis is met het oude.

5.1.8 8. Voorbeeldteksten

Website/landing page – korte intro

FOBIS Next is het distributed service framework voor de volgende generatie van FOBIS.
Het helpt organisaties om hun bestaande FOBIS-landschap veilig te moderniseren, nieuwe digitale diensten sneller te ontwikkelen en gecontroleerd te groeien naar een moderne, servicegerichte architectuur.

Pitchdeck – openingsslide

FOBIS Next
Distributed Service Framework
for the Next Generation of FOBIS

README – eerste alinea

FOBIS Next is een distributed service framework rondom het bestaande FOBIS-platform.
Het framework maakt het mogelijk om nieuwe services en componenten op een moderne, schaalbare manier te ontwikkelen en te integreren, terwijl de betrouwbaarheid van FOBIS behouden blijft. Dit repository bevat de technische basis, richtlijnen en referentie-implementaties voor FOBIS Next.

5.1.9 9. Stijlgids (visuele richtlijnen)

Deze stijlgids is bedoeld als praktisch referentiepunt voor iedereen die met FOBIS Next-materiaal werkt (slides, documenten, web, demo's).

9.1 Kleurenpalet

NB: Hex-waarden zijn indicatief; stem ze in de praktijk af op het bronmateriaal.

- **Primair FOBIS Next-blauw** (voor logo, koppen, knoppen)
- Voorstel: `#0077C8`
- **Accent geel** (vlak in logo, highlights)
- Voorstel: `#FFC928`
- **Accent groen** (vlak in logo, “Next”/toekomst)
- Voorstel: `#B3CF2F`
- **Accent oranje** (vlak in logo, energie/actie)
- Voorstel: `#F46A1A`
- **Achtergrond lichtblauw (hero/banner)**
- Voorstel: van `#E7F4FF` (licht) naar `#2E84D6` (donker) in een zachte diagonale gradient.
- **Neutrale kleuren (tekst)**
- Primair tekst: bijna zwart `#222222`
- Secundair tekst: grijs `#666666`
- Lichte lijnen/ondertitels: grijs `#AAAAAA`

Gebruik:

- Gebruik het **FOBIS Next-blauw** voor logo, belangrijke knoppen en H1/H2-koppen.
- Gebruik **geel/groen/oranje** spaarzaam als accent in grafieken, iconen of callouts.
- Gebruik de **lichtblauwe gradient** alleen voor hero-vlakken of grote bannerachtergronden, niet als standaard paginabackground in apps.

9.2 Vormen en beeldtaal

- **Driehoek / play-vorm**
- De basisvorm is een **rechthoekige driehoek naar rechts**: staat voor “play”, vooruit en next.
- In het hoofdlogo worden **drie kleinere driehoeken** (blauw, geel, oranje) gecombineerd tot één grote pijl, aangevuld met een groene driehoek als “vooruitwijzer”.
- **Stijl van vlakken**
- Hoofdvormen zijn **vlak en geometrisch**, zonder rondingen of gradient in de vlakken zelf.
- Randen zijn bij voorkeur **zonder outlines**; de kleurvlakken raken elkaar direct.
- **Gebruik in visualisaties**
- Gebruik driehoeken en pijlvormen om **flows, services en datastromen** te visualiseren.
- Combineer de logo-vorm subtiel in architectuurdiagrammen als visuele hint (“dit is een FOBIS Next service”).
- Vermijd drukke iconensets; houd illustraties eenvoudig en schematisch.

9.3 Lettertypes

In de huidige slide-set worden **Exo 2 ExtraBold** en **Exo 2 Light** gebruikt; die vormen samen de basis van de FOBIS Next-typografie.

• **Primair lettertype (koppen)**
• **Exo 2 ExtraBold** voor H1/H2-koppen en belangrijke titels (bijv. “FOBIS Next”).
• In HTML/CSS kun je bijvoorbeeld een stack gebruiken als:
• `font-family: "Exo 2", "Segoe UI", system-ui, -apple-system, BlinkMacSystemFont, sans-serif;`
• **Lichaamstekst**
• **Exo 2 Light** of **Exo 2 Regular** voor lopende tekst.
• Regelafstand minimaal 1.3–1.5 voor goede leesbaarheid in lange teksten.
• **Monospace (code en technische fragments)**
• Voor codeblokken en configuratievoorbeelden: **Consolas**, **Fira Code** of **Source Code Pro**.

9.4 Typografie-richtlijnen

• **Titels**: kort en beschrijvend; maximaal één kernboodschap per slide/pagina.
• **Gebruik van kapitalen**:
• “FOBIS” altijd in kapitalen.
• “Next” met hoofdletter N: “FOBIS Next”.
• **Opmaak**:
• Gebruik **vet** voor sleutelwoorden en belangrijke claims.
• Gebruik *cursief* voor nuances, voorbeeldzinnen en meta-commentaar (zoals hier).
• Vermijd onderstrepen behalve voor links.

9.5 Voorbeelden van toepassing

• **Slides**
• Titel in FOBIS Next-blauw op lichte (liefst witte of heel lichtblauwe) achtergrond.
• Logo op één vaste plek (bijv. linksboven), niet herhaald in elk element.
• Maximaal drie kleuren in één slide: blauw + één accent + neutraal grijs.
• **Website / documentatie (MkDocs)**
• Gebruik de primaire kleur voor navigatie en headings.
• Gebruik accentkleuren in callouts (bijv. “Belangrijk”, “Let op”, “Voorbeeld”).
• Houd pagina’s rustig: veel witruimte, korte paragrafen, duidelijke kopjes.