**Oké, hier is de bijgewerkte versie van 02 - Technisch Architectuur Document (TAD) voor NEXA met statusindicatoren per sectie/onderdeel waar relevant:**

**--- START OF FILE 02 - Technisch Architectuur Document (TAD) (Bijgewerkt met Status) ---**

**02 - Technisch Architectuur Document (TAD) voor NEXA  
Status: IN PROGRESS - (Basis DB Setup OK, API Runtime Beslissing Genomen)  
Inhoudsopgave:**

1. **Architectuuroverzicht**
2. **Componenten en Technologiekeuzes**
3. **Dataflow Beschrijvingen (TODO)**
4. **Dataopslag Architectuur**
5. **API Architectuur (TODO - Definitie)**
6. **Security Architectuur**
7. **Deployment Architectuur (TODO - Implementatie)**

**1. Architectuuroverzicht  
Status: [✓ Conceptueel Gedefinieerd]  
NEXA volgt een moderne webapplicatie-architectuur, gecentreerd rond Supabase als Backend-as-a-Service (BaaS) platform en Neo4j als gespecialiseerde graph database. De architectuur bestaat uit de volgende hoofdlagen:**

1. **Frontend Laag (React): De gebruikersinterface, draaiend in de browser van de gebruiker. Verantwoordelijk voor weergave, gebruikersinteractie en communicatie met de API Laag.**
2. **API Laag (Supabase Edge Functions - Deno): De backend logica, gehost als serverless functions binnen Supabase (draaiend op Deno runtime). Verantwoordelijk voor het afhandelen van requests van de frontend, orkestreren van de dataflow, authenticatie/autorisatie en communicatie met de datalagen en externe services.**
3. **Dataopslag Laag:**
   * **Supabase Postgres DB: Relationele database voor persistente opslag van gebruikersdata, authenticatie-info, applicatieconfiguraties, metadata snapshots (geparsed), analyseresultaten en logs. Maakt gebruik van Row Level Security (RLS) voor data-scheiding. [Status: ✓ Basis Schema & RLS Lokaal OK]**
   * **Neo4j Graph DB: Gespecialiseerde graph database (bij voorkeur cloud-hosted via Neo4j AuraDB) voor opslag van de knowledge graph (metadata componenten en relaties per snapshot). Wordt gebruikt voor complexe dependency tracing en impactanalyse queries. [Status: Model Gedefinieerd, Constraints/Indexen nog niet toegepast]**
4. **Externe Services Laag:**
   * **Salesforce Platform: Bron van de metadata, benaderd via OAuth 2.0 en de Metadata/Tooling API's.**
   * **AI Engine (LLM - OpenAI/Claude): Externe service voor het genereren van natuurlijke taal uitleg en remediatiesuggesties, aangeroepen vanuit de API Laag.**

**Hoog-niveau Diagram (Beschrijvend):  
Status: [✓ Conceptueel Gedefinieerd]**

* **De Gebruiker interageert via de browser met de React Frontend.**
* **De React Frontend stuurt HTTPS Requests naar de API Laag (Supabase Edge Functions - Deno) binnen het Supabase Platform.**
* **Binnen het Supabase Platform:**
  + **De API Laag communiceert via SQL/RPC met de Supabase Postgres DB.**
  + **De API Laag communiceert via HTTPS/Bolt met de Neo4j Graph DB (AuraDB), die zich als externe service buiten Supabase bevindt.**
  + **De API Laag doet API Calls naar de AI Engine (LLM) (externe service).**
  + **De API Laag doet OAuth/API Calls naar het Salesforce Platform (externe service).**
  + **De Supabase Postgres DB wordt beveiligd met Row Level Security Policies. [Status: ✓ Basis Geïmplementeerd]**
  + **De API Laag voert Authenticatie Checks uit (via Deno functie code).**
* **De Externe Services omvatten het Salesforce Platform, Neo4j Graph DB, en de AI Engine (LLM).**

**Kernprincipes:  
Status: [✓ Gedefinieerd]**

* **Supabase-Centric: Maximale benutting van Supabase features (Auth, DB, Functions, Storage) om complexiteit te minimaliseren.**
* **Specialized Datastores: Gebruik van Postgres voor gestructureerde data en Neo4j voor graph-specifieke queries.**
* **Serverless: API laag draait op schaalbare serverless functions (Deno runtime).**
* **API-Driven: Communicatie tussen frontend en backend verloopt via een goed gedefinieerde API.**
* **Snapshot-Based: Analyses zijn gebaseerd op point-in-time metadata snapshots.**

**2.1 Frontend (React)**

* **Status: [✓ Geïnitialiseerd & Basis OK]**
* **Framework: React (v18+) [Status: ✓ Gebruikt]**
* **Taal: TypeScript [Status: ✓ Gebruikt]**
* **UI Bibliotheek: Tailwind CSS + Shadcn/UI (Geïmpliceerd) [Status: ✓ Geïnstalleerd & Geconfigureerd]**
* **State Management: Zustand [Status: ✓ Geïnstalleerd, Basis Store OK]**
* **Routing: React Router [Status: ✓ Geïnstalleerd, Basis Routes OK]**
* **Data Fetching: React Query (TanStack Query) [Status: ✓ Geïnstalleerd, Gebruikt voor Mock Data] i.c.m. Supabase JS Client [Status: ✓ Geïnstalleerd]**
* **Graph Visualisatie: Cytoscape.js (Aanbevolen) [Status: [ ] Niet Geïmplementeerd]**
* **Hosting: Vercel (Aanbevolen) [Status: [ ] Niet Ingericht]**

**2.2 API Laag (Supabase Edge Functions)**

* **Status: [✓ Technologie Gedefinieerd, Implementatie TODO]**
* **Runtime: Deno (Runtime geleverd door Supabase) [Status: ✓ Beslissing Genomen]**
* **Taal: TypeScript [Status: ✓ Gekozen]**
* **Framework: Standaard Deno http/server module (of vergelijkbare Deno-native libraries) [Status: ✓ Beslissing Genomen]**
* **Supabase Interactie: Supabase JS Client (@supabase/supabase-js) [Status: ✓ Geïnstalleerd (Frontend), Compatibel]**
* **Neo4j Interactie: Officiële Neo4j JavaScript Driver (neo4j-driver) [Status: [ ] Nog niet geïnstalleerd/getest]**
* **Salesforce Interactie: JSforce (Compatibiliteit Deno TBD) [Status: [ ] Nog niet geïnstalleerd/getest]**
* **LLM Interactie: Officiële SDK's of fetch [Status: [ ] Nog niet geïmplementeerd]**
* **Build Proces: Deno kan TS direct uitvoeren; bundelen via Deno/Supabase CLI. [Status: Proces Gedefinieerd]**

**2.3 Dataopslag Laag**

* **Supabase Postgres DB:**
  + **Status: [✓ Lokaal Geïmplementeerd]**
  + **Versie: Laatste stabiele versie. [Status: ✓ Actief (Lokaal)]**
  + **Schema Beheer: Supabase Migrations. [Status: ✓ In gebruik]**
  + **Toegang: Via Supabase JS Client & Edge Functions. [Status: Concept OK]**
  + **Beveiliging: Row Level Security (RLS) policies. [Status: ✓ Basis geïmplementeerd]**
* **Neo4j Graph DB:**
  + **Status: [✓ Lokaal Gestart, Model Defined]**
  + **Hosting: Neo4j AuraDB (Aanbevolen) / Docker [Status: ✓ Lokaal via Docker]**
  + **Versie: Laatste stabiele versie. [Status: ✓ Actief (Lokaal)]**
  + **Query Taal: Cypher. [Status: ✓ Constraints/Indexen gedefinieerd]**
  + **Toegang: Via Neo4j JavaScript Driver. [Status: [ ] Nog niet geïmplementeerd]**
  + **Beveiliging: Standaard Docker/AuraDB features. [Status: Basis OK (Lokaal)]**

**2.4 Externe Services Laag**

* **Status: [✓ Gedefinieerd, Integratie TODO]**
* **Salesforce Platform:**
  + **Authenticatie: OAuth 2.0 PKCE Flow. [Status: Concept OK]**
  + **API's: Metadata API, Tooling API, etc. [Status: Concept OK]**
* **AI Engine (LLM):**
  + **Provider: OpenAI/Anthropic (Configureerbaar). [Status: Concept OK]**
  + **Integratie: SDK's / REST API. [Status: Concept OK]**
  + **Beveiliging: API keys via Supabase secrets. [Status: Concept OK]**

**2.5 Ontwikkeling & Tooling**

* **Status: [✓ Ingesteld & Gebruikt]**
* **Versiebeheer: Git (GitHub) [Status: ✓ Actief]**
* **Code Editor: VS Code [Status: ✓ Actief]**
* **Package Manager: PNPM [Status: ✓ Actief]**
* **Lokale Ontwikkeling:**
  + **Supabase CLI [Status: ✓ Actief]**
  + **Docker Compose (voor Neo4j) [Status: ✓ Actief]**
* **CI/CD: GitHub Actions (Aanbevolen) [Status: [ ] Niet Ingericht]**

**3. Dataflow Beschrijvingen  
Status: [✓ Conceptueel Gedefinieerd, Implementatie TODO]  
Deze sectie beschrijft de sequenties van interacties tussen componenten voor kernfunctionaliteiten. *(Let op: Verwijzingen naar de API Laag impliceren nu de Deno runtime.)***

**3.1 Dataflow: Metadata Pipeline (Fetch, Parse, Store, Build Graph)  
Dit proces wordt getriggerd door de gebruiker vanuit de frontend om Salesforce metadata te importeren en de knowledge graph te bouwen/updaten.**

1. **Trigger (Frontend):**
   * **Gebruiker selecteert een gekoppelde Salesforce Org en klikt op "Fetch Metadata" (of vergelijkbaar).**
   * **De React Frontend stuurt een authenticated HTTPS POST request naar een specifieke API endpoint op een Supabase Edge Function (bijv. /functions/v1/pipeline-start). De request bevat het ID van de gekoppelde org. [Status: Frontend UI element bestaat (conceptueel), API call TODO]**
2. **Start Pipeline (API Function - Deno):**
   * **De Edge Function valideert de request en de autorisatie van de gebruiker (via Supabase Auth context). [Status: Auth validatie TODO]**
   * **Maakt een nieuw metadata\_snapshot record aan in de Supabase Postgres DB met status 'PENDING' of 'FETCHING', gekoppeld aan de org en gebruiker. Retourneert het snapshot\_id naar de frontend. [Status: DB interactie TODO]**
3. **Fetch Metadata (API Function - Deno):**
   * **De Edge Function haalt de benodigde Salesforce access token (veilig opgeslagen/ontsleuteld) op voor de geselecteerde org. [Status: Token beheer/decryptie TODO]**
   * **Gebruikt een Deno-compatibele library (zoals JSforce, indien compatibel, of directe fetch calls) om verbinding te maken met de Salesforce Platform Metadata API. [Status: SF API interactie TODO]**
   * **Vraagt de benodigde metadata types op (FR-PIPE-02).**
   * **Status Update: Werkt de status van de metadata\_snapshot bij. [Status: DB interactie TODO]**
4. **Parse Metadata (API Function - Deno):**
   * **Na ontvangst van de metadata, parsed de Edge Function deze data (XML/JSON). [Status: Parsing logica TODO]**
   * **Zet ruwe metadata om naar de gedefinieerde structuur (FR-PIPE-03).**
   * **Status Update: Werkt de status bij. [Status: DB interactie TODO]**
5. **Store Metadata (API Function - Deno):**
   * **De Edge Function schrijft de gestructureerde metadata naar parsed\_components en parsed\_relationships in Supabase DB. [Status: DB interactie TODO]**
   * **Status Update: Werkt de status bij. [Status: DB interactie TODO]**
6. **Start Graph Population (API Function - Deno - mogelijk aparte trigger/functie):**
   * **Wordt getriggerd na succesvolle opslag. [Status: Trigger mechanisme TODO]**
   * **Status Update: Werkt de status bij. [Status: DB interactie TODO]**
7. **Populate Graph (API Function - Deno):**
   * **Leest gestructureerde metadata uit Supabase DB. [Status: DB interactie TODO]**
   * **Vertaalt data naar Cypher statements. [Status: Vertaling logica TODO]**
   * **Maakt verbinding met Neo4j (via driver - check Deno compat.). [Status: Neo4j interactie TODO]**
   * **Voert Cypher queries uit.**
8. **Voltooiing (API Function - Deno & Frontend):**
   * **Werkt snapshot status bij naar 'COMPLETED' of 'FAILED'. [Status: DB interactie TODO]**
   * **Frontend kan status pollen of via Realtime updates ontvangen. [Status: Frontend status update TODO]**

**3.2 Dataflow: AI Impact Analyse Vraag  
Dit proces beschrijft hoe een vraag van de gebruiker in de AI Assistant wordt afgehandeld.**

1. **Vraag Stellen (Frontend):**
   * **Gebruiker typt vraag in AI Assistant UI.**
   * **React Frontend stuurt authenticated POST request naar een API endpoint (bijv. /functions/v1/ai-ask). [Status: Frontend UI element bestaat (conceptueel), API call TODO]**
2. **Vraag Verwerking (API Function - Deno):**
   * **Valideert input en autorisatie. [Status: Auth validatie TODO]**
   * **Interpreteert gebruikersvraag (FR-AI-02). [Status: NLU/Intentieherkenning TODO]**
3. **Graph Query (API Function - Deno):**
   * **Genereert Cypher query(s). [Status: Cypher generatie TODO]**
   * **Voert query(s) uit op Neo4j. [Status: Neo4j interactie TODO]**
4. **Context Voorbereiding (API Function - Deno):**
   * **Verzamelt Neo4j resultaten. [Status: TODO]**
   * **Bereidt context/prompt voor LLM voor (FR-AI-03). [Status: Prompt engineering TODO]**
5. **LLM Aanroep (API Function - Deno):**
   * **Stuurt prompt naar AI Engine (OpenAI/Claude) via API/SDK. [Status: LLM API interactie TODO]**
   * **Wacht op antwoord.**
6. **Antwoord Verwerking & Opslag (API Function - Deno):**
   * **Ontvangt antwoord van LLM.**
   * **Verwerkt antwoord (optioneel). [Status: TODO]**
   * **Slaat interactie op in ai\_interactions tabel (FR-AI-07). [Status: DB interactie TODO]**
   * **Stuurt antwoord terug naar frontend.**
7. **Antwoord Tonen (Frontend):**
   * **React Frontend ontvangt antwoord.**
   * **Toont antwoord in UI (FR-AI-04). [Status: Frontend UI update TODO]**
   * **Toont feedback knoppen (FR-AI-06). [Status: Frontend UI update TODO]**
8. **(Optioneel) Feedback Verwerking:**
   * **Frontend stuurt feedback naar API endpoint (bijv. /functions/v1/ai-feedback). [Status: Frontend API call TODO]**
   * **Edge Function slaat feedback op in ai\_interactions (FR-AI-07). [Status: DB interactie TODO]**

**4. Dataopslag Architectuur  
Status: [✓ Supabase Lokaal OK, Neo4j Model Defined]  
Deze sectie beschrijft de structuur en modellen voor dataopslag in Supabase Postgres en Neo4j.**

**4.1 Supabase (PostgreSQL) Datamodel**

* **Status: [✓ Lokaal Geïmplementeerd & Geseed]  
  De Supabase database wordt gebruikt voor gestructureerde data, gebruikersbeheer, en het persisteren van de geparste metadata voordat deze naar Neo4j gaat.**
* **Kern Tabellen: [Status: ✓ Alle tabellen aangemaakt via migratie]**
  + **users (Beheerd door Supabase Auth)**
  + **organizations**
  + **organization\_members**
  + **salesforce\_connections**
  + **metadata\_snapshots**
  + **parsed\_components**
  + **parsed\_relationships**
  + **analysis\_results**
  + **ai\_interactions  
    *(Gedetailleerde kolomdefinities zoals in File 04 zijn hier niet herhaald, maar zijn wel geïmplementeerd).***
* **Seed Data: [Status: ✓ OK] Het lokale schema wordt gevuld met testdata (Test Organisatie, Lidmaatschap, Test Connecties) via apps/supabase/seed.sql, dat automatisch draait na supabase db reset. Dit vereist dat een specifieke testgebruiker vooraf handmatig wordt aangemaakt in Supabase Auth.**
* **Beveiliging: Row Level Security (RLS) policies zijn geïmplementeerd om ervoor te zorgen dat gebruikers alleen toegang hebben tot data die behoort tot de organizations waarvan zij lid zijn. [Status: ✓ Basis geïmplementeerd]**

**4.2 Neo4j Graph Model**

* **Status: [✓ Model Gedefinieerd, Setup TODO]  
  De Neo4j database bevat de knowledge graph, geoptimaliseerd voor het traceren van afhankelijkheden. Elke node en relatie bevat een snapshotId property om data van verschillende snapshots te scheiden.**
* **Node Labels (Voorbeelden): [Status: ✓ Gedefinieerd]**
  + **:MetadataComponent (Abstract)**
    - **Properties: snapshotId, apiName, label, type, sfId, createdAt**
  + **:CustomObject**
  + **:CustomField**
  + **:Flow**
  + **... (Andere types)**
* **Relationship Types (Voorbeelden): [Status: ✓ Gedefinieerd]**
  + **BELONGS\_TO**
  + **REFERENCES**
  + **USED\_IN (met properties)**
  + **TRIGGERS**
  + **IMPACTS**
* **Indexering: Indexen op :MetadataComponent(snapshotId) en :MetadataComponent(apiName) en :MetadataComponent(type) zijn gedefinieerd. [Status: ✓ Gedefinieerd, nog niet toegepast]**
* **Constraint: Een UNIQUE constraint op (:MetadataComponent {snapshotId, apiName}) is gedefinieerd. [Status: ✓ Gedefinieerd, nog niet toegepast]**
* **Model Evolutie: Dit model is een startpunt en kan evolueren.**

**5. API Architectuur  
Status: [✓ Conceptueel Gedefinieerd, Implementatie TODO]  
Deze sectie definieert de architectuur van de API-laag die draait op Supabase Edge Functions (Deno runtime) en dient als de interface tussen de frontend en de backend logica/data.**

**5.1 API Stijl en Format**

* **Status: [✓ Gedefinieerd]**
* **Stijl: RESTful API. We gebruiken standaard HTTP-methoden (GET, POST, PUT, DELETE) en resource-gebaseerde URL's (bijv. /functions/v1/<resource>).**
* **Format: JSON voor zowel request bodies als response bodies.**
* **Content Type: application/json.**

**5.2 Hosting en Framework**

* **Status: [✓ Technologie Gedefinieerd]**
* **Hosting: Supabase Edge Functions. Elke (of een groep gerelateerde) endpoint(s) wordt geïmplementeerd als een aparte Edge Function in een eigen map binnen apps/supabase/functions/.**
* **Framework: Deno runtime. Request handling gebeurt primair met de standaard Deno http/server module of Deno-compatibele routing libraries (indien nodig voor complexere functies). Geen Express.js.**

**5.3 Basispad en Versioning**

* **Status: [✓ Gedefinieerd]**
* **Basispad: Supabase Functions hebben een standaard basispad (bijv. /functions/v1/). We gebruiken de functiemapnaam als resource (bv. /functions/v1/connections). Verdere routing binnen een functie is mogelijk maar minder gebruikelijk dan bij Express.**
* **Versioning: API-versies kunnen worden beheerd via het URL-pad (bv. /functions/v1/, /functions/v2/) door functies in verschillende versiemappen te organiseren of via URL-structuur binnen de functie.**

**5.4 Authenticatie en Autorisatie**

* **Status: [✓ Conceptueel Gedefinieerd, Implementatie TODO]**
* **Authenticatie: Alle beschermde API-endpoints vereisen authenticatie via JWT (Supabase Auth). De frontend stuurt het token in de Authorization: Bearer <token> header.**
* **Token Validatie: Edge Function code valideert het token bij elke beschermde request met Supabase/Deno helpers.**
* **Autorisatie: Na authenticatie bepaalt de Edge Function code of de gebruiker gemachtigd is (op basis van user\_id, organization\_members, etc.), ondersteund door RLS op database niveau. [RLS Basis Status: ✓ Geïmplementeerd]**

**5.5 Endpoint Definitie Overzicht (Conceptueel)**

* **Status: [✓ Conceptueel Gedefinieerd, Implementatie TODO]  
  *(De conceptuele endpoints blijven hetzelfde, maar de implementatie zal in Deno zijn)***
* **Authenticatie (Grotendeels Supabase Client SDK + specifieke Functions):**
  + **Standaard Supabase endpoints ...**
  + **POST /functions/v1/sfdc-auth-initiate: Start OAuth flow.**
  + **GET /functions/v1/sfdc-auth-callback: Callback endpoint.**
* **Salesforce Connections:**
  + **GET /functions/v1/connections: Haalt lijst op. [Status: Implementatie Volgende Stap]**
  + **GET /functions/v1/connections?id={connectionId}: Haalt details op. (Alternatief: /functions/v1/connection-details met ID in body/header).**
  + **DELETE /functions/v1/connections?id={connectionId}: Verwijdert koppeling.**
* **Metadata Pipeline & Snapshots:**
  + **POST /functions/v1/pipeline-start: Start pipeline.**
  + **GET /functions/v1/snapshots?connectionId={connectionId}: Haalt lijst op.**
  + **GET /functions/v1/snapshot-status?id={snapshotId}: Haalt status op.**
  + **GET /functions/v1/snapshot-details?id={snapshotId}: Haalt details op.**
* **Graph & Analyse:**
  + **GET /functions/v1/graph-visualize?snapshotId={...}: Haalt data op voor visualisatie.**
  + **GET /functions/v1/graph-component?snapshotId={...}&apiName={...}: Haalt componentdetails op.**
  + **POST /functions/v1/analysis-start: Start analyse.**
  + **GET /functions/v1/analysis-results?snapshotId={...}: Haalt resultaten op.**
* **AI Assistant:**
  + **POST /functions/v1/ai-ask: Stuurt vraag naar AI.**
  + **POST /functions/v1/ai-feedback: Stuurt feedback.**

**5.6 Error Handling**

* **Status: [✓ Conceptueel Gedefinieerd, Implementatie TODO]**
* **API-fouten worden geretourneerd met passende HTTP status codes (4xx, 5xx).**
* **Response body bij fout bevat gestructureerd JSON-object ({ "error": "...", "details": "..." }). Implementatie in elke Edge Function.**

**5.7 CORS (Cross-Origin Resource Sharing)**

* **Status: [✓ Basis OK, Verfijning TODO]**
* **Supabase Edge Functions moeten correcte CORS headers retourneren. Dit wordt afgehandeld binnen de serve functie in index.ts. De Access-Control-Allow-Origin moet specifiek worden ingesteld voor staging/productie omgevingen (niet '\*').**

**6. Security Architectuur  
Status: [✓ Conceptueel Gedefinieerd, ✓ RLS Basis OK, Encryptie Kolommen OK]**Deze sectie beschrijft de beveiligingsmaatregelen die worden geïmplementeerd om de vertrouwelijkheid, integriteit en beschikbaarheid van het NEXA platform en de data van gebruikers te waarborgen.

**6.1 Authenticatie**

* **Status: [✓ Mechanisme Gekozen (Supabase Auth)]**
* Mechanisme: Supabase Auth wordt gebruikt voor gebruikersauthenticatie. Ondersteuning voor e-mail/wachtwoord en mogelijk OAuth providers (Google, GitHub) voor registratie en login (FR-AUTH-01, FR-AUTH-02).
* JWT: Na succesvolle login geeft Supabase Auth een JSON Web Token (JWT) uit. Dit token wordt door de frontend opgeslagen (bijv. in localStorage via Supabase JS Client) en meegestuurd in de Authorization: Bearer <token> header bij elke API-request naar de Supabase Edge Functions.
* Token Validatie: **Edge Function code** (binnen de Deno runtime) valideert de handtekening en geldigheid (vervaldatum) van het JWT bij elke beschermde request met behulp van Supabase helper libraries of standaard JWT libraries compatibel met Deno. De user\_id wordt uit het gevalideerde token geëxtraheerd voor autorisatiechecks. [Status: Implementatie TODO]
* Wachtwoordbeleid: Supabase Auth standaardinstellingen voor wachtwoordcomplexiteit en eventuele MFA (Multi-Factor Authentication) worden overwogen. [Status: Configuratie TODO]

**6.2 Autorisatie**

* **Status: [✓ RLS Basis OK, API Autorisatie TODO]**
* API Autorisatie: De API-laag (**Edge Functions code**) voert autorisatiechecks uit na succesvolle authenticatie. Deze checks verifiëren of de geauthenticeerde gebruiker (user\_id uit JWT) de gevraagde actie mag uitvoeren op de specifieke resource (bijv. toegang tot een snapshotId of connectionId). Dit gebeurt op basis van lidmaatschap van organizations en eventuele rollen (uit organization\_members tabel). [Status: Implementatie TODO]
* Database Autorisatie (RLS): Supabase Row Level Security (RLS) is geconfigureerd op kritieke tabellen (organizations, organization\_members, salesforce\_connections, metadata\_snapshots, parsed\_components, etc.). RLS policies zorgen ervoor dat database queries (ook die via de Supabase client library of RPC calls vanuit Edge Functions) alleen data retourneren die behoort tot de organisatie(s) van de geauthenticeerde gebruiker. Dit biedt een diepe verdedigingslaag. [Status: ✓ Basis Geïmplementeerd]

**6.3 Data Security**

* **Status: [✓ Concept Gedefinieerd, Encryptie Kolommen OK]**
* Encryptie in Transit: Alle communicatie ... verloopt via HTTPS/TLS...
* Encryptie at Rest (Supabase):
  + Supabase versleutelt data at rest standaard op infrastructuurniveau.
  + Specifieke Kolom Encryptie: Gevoelige data, met name access\_token\_encrypted en refresh\_token\_encrypted in de salesforce\_connections tabel, *moeten* expliciet applicatief versleuteld worden voordat ze naar de database worden geschreven. Dit gebeurt in de Edge Function(s) die tokens opslaan/bijwerken, mogelijk met pgsodium via database functies aangeroepen vanuit de Edge Function, of met een Deno-compatibele encryptie library (met sleutels veilig beheerd als Supabase secrets). [Status: ✓ Kolommen Gedefinieerd, Encryptie Logica TODO]
* Encryptie at Rest (Neo4j AuraDB): Neo4j AuraDB biedt standaard encryptie at rest.
* Data Masking/Minimization: Er wordt geen onnodig gevoelige Salesforce data opgeslagen...

**6.4 API Security**

* **Status: [Conceptueel Gedefinieerd, Implementatie TODO]**
* Input Validatie: Alle data die via API-requests binnenkomt ... wordt gevalideerd ... (bijv. met een Deno-compatibele library zoals Zod).
* Rate Limiting: Implementatie van rate limiting op API-endpoints (mogelijk via Supabase ingebouwde features of Deno middleware zoals oak indien een framework wordt gebruikt).
* Security Headers: Gebruik van standaard security headers ... op zowel de frontend hosting als de API responses (ingesteld in de Edge Function response).
* CORS: Strikte configuratie van Cross-Origin Resource Sharing (CORS) headers in de Edge Function responses om alleen requests van de toegestane frontend domeinen toe te staan (dynamisch instelbaar per omgeving). [Status: Basis headers in voorbeeld, verfijning nodig]

**6.5 Salesforce Connectie Security**

* **Status: [Conceptueel Gedefinieerd, Implementatie TODO]**
* OAuth 2.0 PKCE: De Proof Key for Code Exchange (PKCE) extensie wordt gebruikt tijdens de OAuth flow (geïmplementeerd in de relevante Edge Functions).
* Veilige Token Opslag: Refresh en access tokens worden versleuteld opgeslagen in de Supabase database (zie 6.3) en beheerd door backend Edge Functions. Tokens worden nooit direct naar de frontend gestuurd.
* Scoped Permissions: Tijdens de OAuth flow worden alleen de minimaal benodigde Salesforce permissies aangevraagd...

**6.6 Infrastructuur Security**

* **Status: [✓ Platform Keuzes Gemaakt]**
* Supabase: Vertrouwt op de beveiligingsmaatregelen van het Supabase platform... Toegang tot het Supabase project dashboard wordt beperkt en beveiligd met MFA.
* Neo4j AuraDB: Vertrouwt op de beveiligingsmaatregelen van Neo4j's cloud service... Database credentials worden veilig opgeslagen als secrets in de Supabase Functions environment. [Status: Lokaal: Docker gebruikt, Cloud: AuraDB aanbevolen]
* LLM Provider: API keys ... worden veilig opgeslagen als secrets in de Supabase Functions environment...

**6.7 Dependency Management**

* **Status: [Tooling Gekozen, Proces TODO]**
* Scanning: Regelmatig scannen van project dependencies (PNPM packages, Deno modules) op bekende kwetsbaarheden met tools zoals pnpm audit, Snyk, of GitHub Dependabot. Voor Deno: deno lint --unstable kan helpen, en externe scanners.
* Updates: Tijdig updaten van dependencies ...

**6.8 Logging en Monitoring**

* **Status: [Conceptueel Gedefinieerd, Implementatie TODO]**
* Audit Logging: Belangrijke security-gerelateerde events ... worden gelogd (bijv. via console.log in Edge Functions, wat verschijnt in Supabase Function logs, of door expliciet naar een logtabel te schrijven).
* Monitoring: Monitoring op ongebruikelijke activiteiten of error rates (via Supabase ingebouwde monitoring of externe tools).

**7. Deployment Architectuur  
Status: [✓ Conceptueel Gedefinieerd, ✓ Lokale Omgeving OK, Staging/Prod TODO]**Deze sectie beschrijft de strategie en processen voor het bouwen, testen en deployen van de NEXA applicatie naar de verschillende omgevingen.

**7.1 Omgevingen**Er worden minimaal drie omgevingen onderhouden:

1. **local (Ontwikkelomgeving):**
   * **Status: [✓ Operationeel]**
   * Doel: Individuele ontwikkeling en unit testing.
   * Setup: Draait lokaal op de machine van de ontwikkelaar.
   * Componenten:
     + Frontend (React): Draait lokaal via pnpm --filter frontend dev.
     + Supabase Stack: Draait lokaal via supabase start (incl. Postgres DB, Auth, Storage, Edge Functions Deno runtime). Database wordt geïnitialiseerd met migraties en seed.sql.
     + Neo4j: Draait lokaal via Docker Compose. (Constraints/Indexen nog handmatig toepassen na reset).
     + Salesforce: Ontwikkelaars gebruiken Developer Edition orgs of Sandboxes.
   * Secrets: Lokale .env bestanden.
2. **staging (Test-/Acceptatieomgeving):**
   * **Status: [Nog Niet Ingericht]**
   * Doel: Integratietesten, UAT.
   * Setup: Gehost in de cloud.
   * Componenten:
     + Frontend (React): Gehost op Vercel/Netlify.
     + Supabase Project: Apart Supabase project (betaalde tier waarschijnlijk nodig).
     + Neo4j Instance: Aparte Neo4j AuraDB instance.
     + Salesforce: Koppelingen naar Sandboxes.
   * Data: Testdata (mogelijk geanonimiseerd).
   * Deployment: Via CI/CD vanuit develop branch.
3. **production (Productieomgeving):**
   * **Status: [Nog Niet Ingericht]**
   * Doel: Live omgeving.
   * Setup: Gehost in de cloud.
   * Componenten:
     + Frontend (React): Gehost op Vercel/Netlify.
     + Supabase Project: Apart Supabase project (betaalde tier).
     + Neo4j Instance: Aparte Neo4j AuraDB instance (geschaald).
     + Salesforce: Koppelingen naar productie orgs.
   * Data: Live klantdata. Backups, monitoring essentieel.
   * Deployment: Via CI/CD vanuit main branch (met goedkeuring).

**7.2 CI/CD Pipeline (Continuous Integration / Continuous Deployment)**

* **Status: [Conceptueel Gedefinieerd, Implementatie TODO]**
* Tool: GitHub Actions (aanbevolen).
* Trigger: Pushes naar develop/main, Pull Requests.
* **Pipeline Stappen (Voorbeeld):**
  1. Checkout Code
  2. Setup Environment (Node.js voor PNPM, Deno voor functions)
  3. Install Dependencies (pnpm install)
  4. Lint & Format Check (pnpm lint, pnpm format:check)
  5. Unit Tests (pnpm test:frontend, supabase functions test? - Deno test commando's)
  6. Build Frontend (pnpm --filter frontend build)
  7. Build/Bundle Functions (**Check Deno/Supabase CLI voor bundling/dependency management indien nodig**)
  8. (Optioneel) Integration Tests
  9. Deploy Supabase Migrations (supabase link, supabase migration up via CLI in CI/CD context)
  10. Deploy Supabase Edge Functions (supabase functions deploy via CLI in CI/CD context)
  11. Deploy Frontend (Vercel/Netlify CLI/Action)
  12. (Optioneel) End-to-End Tests
  13. Notificatie
* Deployment naar Productie: Vergelijkbaar, vanuit main, mogelijk handmatige trigger.