SDD - Dokument o návrhu softwaru

Obsah

[1 Úvod 1](#_Toc198857337)

[2 Architektura systému 1](#_Toc198857338)

[2.1 Diagram architektury 2](#_Toc198857339)

[3 Databázový návrh 2](#_Toc198857340)

[3.1 Entity a vztahy 2](#_Toc198857341)

[3.2 Přehled entit 2](#_Toc198857342)

[3.3 Enums 3](#_Toc198857343)

[4 Moduly systému 4](#_Toc198857344)

[4.1 User modul 4](#_Toc198857345)

[4.2 Branch modul 4](#_Toc198857346)

[4.3 Owner modul 5](#_Toc198857347)

[4.4 Employee modul 5](#_Toc198857348)

[4.5 Role & Přístupový modul 6](#_Toc198857349)

[5 Bezpečnostní návrh 6](#_Toc198857350)

[5.1 Autentizace 7](#_Toc198857351)

[5.2 Autorizace (RBAC) 7](#_Toc198857352)

[5.3 Ochrana dat a komunikace 7](#_Toc198857353)

[5.4 Prevence zneužití 7](#_Toc198857354)

[6 Deployment návrh 8](#_Toc198857355)

# 1 Úvod

**Název aplikace:** CompanyTrack

**Verze dokumentu:** 1.0

**Autor:** Matyáš Bláha

**Datum vytvoření:** 10.4.2025

**Účel dokumentu:**

Tento dokument popisuje návrh softwaru aplikace *CompanyTrack*, určené pro správu firem, jejich poboček, zaměstnanců a ekonomických údajů. Obsahuje architektonický návrh, strukturu databáze, hlavní moduly systému, bezpečnostní mechanismy a deployment řešení.

# 2 Architektura systému

Aplikace CompanyTrack je postavena jako **webová aplikace s client-server architekturou**. Frontend i backend jsou vytvořeny pomocí JavaScriptového ekosystému s důrazem na modularitu a škálovatelnost.

Backend je postavený na vrstvené archutektiře (controllers, services, repositories, dtos)

* **Frontend**: React / Next.js
* **Backend API**: Nest.js
* **Databáze**: PostgreSQL (hostovaná přes Supabase)
* **Autentizace**: Supabase Auth (JWT)
* **Data-fetching**: Axios + Orval (OpenAPI generovaný klient)
* **Stavová správa**: React Query

## 2.1 Diagram architektury

Obsah obrázku text, snímek obrazovky, Písmo, číslo

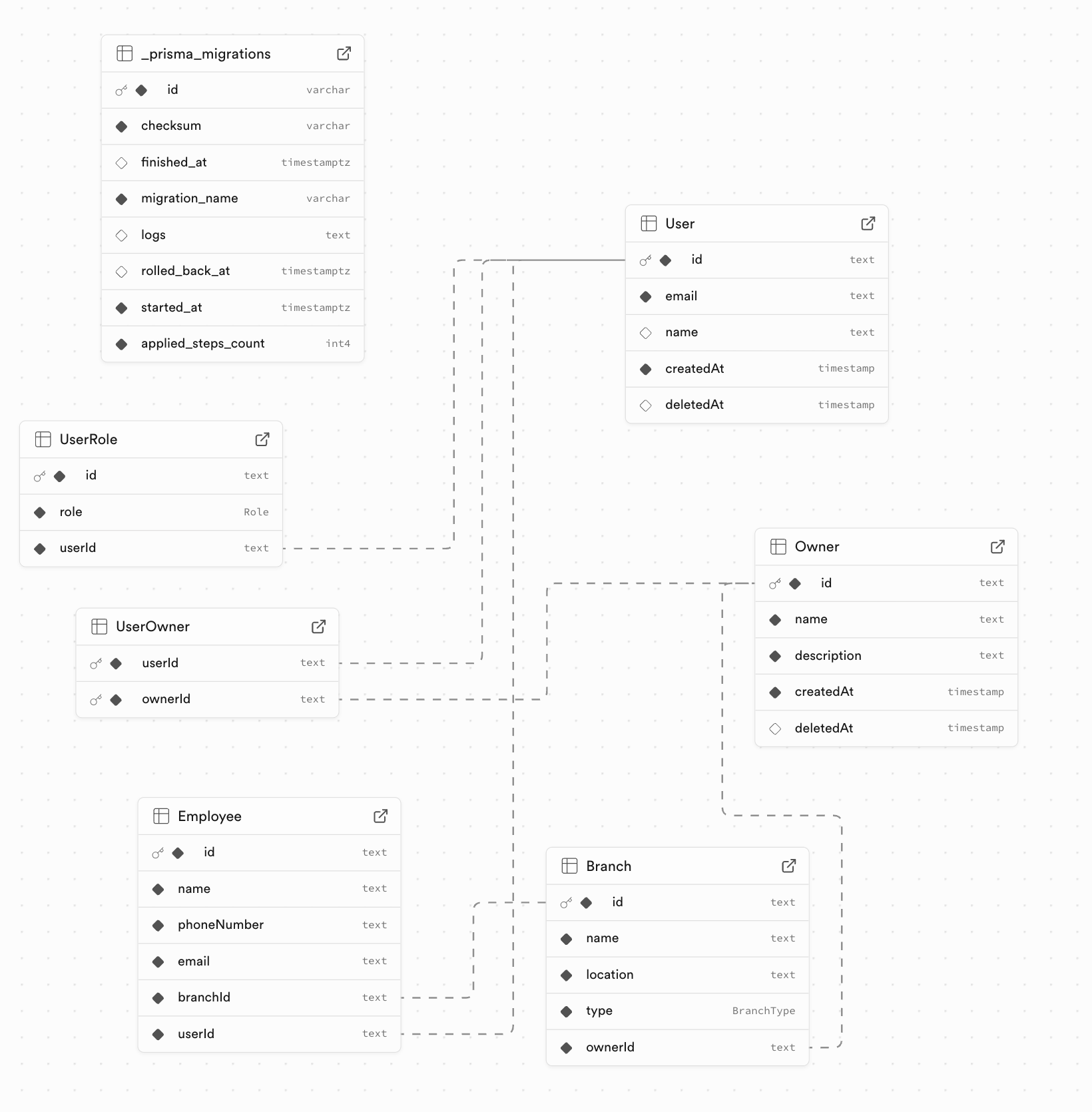
Obsah generovaný pomocí AI může být nesprávný.

# 3 Databázový návrh

## 3.1 Entity a vztahy

User (1) ───< (N) UserOwner >─── (1) Owner  
User (1) ──── (0..1) UserRole  
User (1) ──── (N) Employee  
Owner (1) ──── (N) Branch  
Branch (1) ──── (N) Employee  
Employee (1) ──── (1) User

3.2 Přehled entit



## 3.3 Enums

Role: SUPERADMIN, ACCOUNTANT, MANAGER, USER, HR

BranchType: OFFICE, MARKET, WAREHOUSE

# 4 Moduly systému

## 4.1 User modul

**Zodpovědnosti:**

* Správa uživatelů (napojení na Supabase uživatele)
* Získání detailů o uživateli, jeho roli, napojení na entitu Employee
* Práce s JWT autentizací

**Backend:**

* user.controller.ts: REST endpointy (getMe, getById, getAll)
* user.service.ts: logika pro získání uživatele, případně synchronizace se Supabase
* user.repository.ts: přístup k entitě User v databázi
* user.dto.ts: typování výstupních struktur

**Frontend:**

* Generované hooky přes Orval (např. useUserMe, useUserDetail)
* React komponenty pro zobrazení údajů
* Přihlášení a získání JWT cookie

## 4.2 Branch modul

**Zodpovědnosti:**

* CRUD operace pro pobočky (Branch)
* Napojení na majitele (Owner)
* Typování přes BranchDto

**Backend:**

* branch.controller.ts: /branches, /branches/:id
* branch.service.ts: getAll, getById
* branch.repository.ts: používá prisma.branch.findMany a findUnique
* branch.dto.ts: DTO obsahuje i navázaný OwnerDto

**Frontend:**

* Hooky typu useBranches, useBranchById
* Zobrazení v tabulce, možnost výběru typu pobočky (BranchType)

## 4.3 Owner modul

**Zodpovědnosti:**

* Správa firem/majitelů
* Napojení na UserOwner relaci
* Vztah k více pobočkám

**Backend:**

* owner.controller.ts: CRUD API
* owner.service.ts, owner.repository.ts
* DTO OwnerDto – používáno v BranchDto

**Frontend:**

* Hooky pro získání firem
* Komponenty pro výběr vlastníka při zakládání pobočky

## 4.4 Employee modul

**Zodpovědnosti:**

* Zaměstnanci v pobočkách
* Napojení na User
* Přidání/odebrání zaměstnance

**Backend:**

* employee.controller.ts: REST endpointy
* employee.service.ts: business logika
* employee.repository.ts: dotazy přes prisma.employee
* DTO EmployeeDto

**Frontend:**

* Hooky typu useEmployeesByBranch
* Formulář pro přidání zaměstnance do pobočky

## 4.5 Role & Přístupový modul

**Zodpovědnosti:**

* Správa uživatelských rolí
* RBAC (role-based access control)

**Backend:**

* Guardy v NestJS (JwtAuthGuard, RolesGuard)
* Rozlišení oprávnění dle UserRole

**Frontend:**

* Skrývání komponent dle role
* Redirecty podle oprávnění

# 5 Bezpečnostní návrh

Aplikace CompanyTrack klade důraz na bezpečnost uživatelských dat, přístupových práv a komunikace. Bezpečnostní návrh zahrnuje tyto oblasti:

## 5.1 Autentizace

* **Supabase Auth** zajišťuje přihlášení pomocí e-mailu a hesla.
* E-mailová adresa uživatele musí být ověřena (double opt-in).
* Po přihlášení je uživateli vydán **JWT token**, který obsahuje jeho identifikaci a roli.
* Token je uložen do **HttpOnly + Secure** cookie, aby nebyl přístupný JavaScriptem a byl chráněn před XSS útoky.

## 5.2 Autorizace (RBAC)

* Aplikace využívá **Role-Based Access Control**.
* Každý uživatel má přiřazenou roli (např. SUPERADMIN, MANAGER, HR…).
* Backend používá **NestJS Guardy** (JwtAuthGuard, RolesGuard), které ověřují:
  + že požadavek obsahuje platný JWT token
  + že uživatel má oprávnění ke konkrétní akci
* Frontend skrývá komponenty a blokuje přístupy na základě role.

## 5.3 Ochrana dat a komunikace

* Veškerá komunikace probíhá přes **HTTPS**.
* Hesla nejsou nikdy ukládána do databáze ručně – Supabase Auth používá bezpečné hashování.
* Citlivá pole jako hesla, tokeny nebo interní ID nejsou vystavována frontendové aplikaci.
* Přístup k datům je limitován jak na úrovni frontendových hooků, tak backend endpointů.

## 5.4 Prevence zneužití

* Systém má definované **kvóty a limity** (např. max. 2 pobočky a 10 zaměstnanců v trial verzi).
* Uživatelé bez ověření nemají přístup do systému.
* Neautorizované přístupy jsou logovány (budoucí rozšíření: Sentry nebo Vercel Analytics).

# 6 Deployment návrh

**Cílem** je zajistit škálovatelné, bezpečné a snadno spravovatelné nasazení webové aplikace CompanyTrack.

**Frontend (Next.js)**

* **Hosting:** [Vercel](https://vercel.com/) – plně integrovaný s Next.js
* **CI/CD:** automatické buildy z main větve (GitHub)
* **Environment variables:** nastavené ve Vercel dashboardu (NEXT\_PUBLIC\_API\_URL, atd.)

**Backend (Nest.js API)**

* **Možnosti:**
  + **Option 1 (Serverless):** Nasazení do Vercel jako serverless funkce
  + **Option 2 (Dedicated):** Nasazení na **Railway**, **Render** nebo **Fly.io** jako long-running container
* **Build pipeline:** GitHub Actions (lint, test, build, deploy)

**Databáze (PostgreSQL)**

* **Hostováno přes Supabase**
  + Bezpečný přístup přes role-based policy
  + Data jsou dostupná i přes Supabase Studio pro správu
* **ORM:** Prisma – generovaný klient (v apps/api/generated)

**Autentizace & Autorizace**

* **Supabase Auth (JWT)**
  + JWT token se ukládá do httpOnly cookie
* **NestJS Guardy:** role-based ochrana backend endpointů

**Monitoring (volitelně)**

* Sentry, LogRocket nebo Vercel Analytics (v budoucnu)