UNIVERZITA HRADEC KRÁLOVÉ
FAKULTA INFORMATIKY A MANAGEMENTU

Aplikace pro sportovní centrum „ActiveLife“

Seminární práce z předmětu Databázové systémy 2

Členové pracovního týmu DataForge:

Jakub Doležal, Jakub Kyzr, Václav Havelka

V Hradci Králové
dne 20. dubna 2025

Obsah

1 Úvod

Cílem projektu je vytvoření moderní databázové aplikace pro sportovní centrum "ActiveLife", která zajistí efektivní správu rezervací sportovišť a aktivit. Nový systém nahradí dosavadní, převážně manuální řešení, které již nedostačuje rostoucím požadavkům centra. Sportovní centrum nabízí široké spektrum aktivit a dosavadní IT řešení založené na jednoduché evidenci je neflexibilní a pomalé. Nová aplikace má za cíl zefektivnit provoz, usnadnit správu informací a zlepšit uživatelskou zkušenost.

2 Zadání

Aplikace bude nasazena v prostředí sportovního centra "ActiveLife", které potřebuje nahradit stávající manuální systém rezervací moderním databázovým řešením. Požadavky na nový systém zahrnují:

- **Evidence a správa dat:** Záznamy o uživatelích (návštěvnících), sportovištích, aktivitách, rezervacích v časových slotech (včetně storna a úprav) a zaměstnancích (včetně směn).
- **Vstupy:** Uživatelské formuláře pro registraci, přihlášení, vytváření a úpravu rezervací, zadávání aktivit a cen.
- **Výstupy:** Reporty o využití sportovišť, seznam rezervací pro uživatele, statistiky pro plánování a marketing.
- **Uživatelské role:** Běžný uživatel (správa vlastních rezervací), Zaměstnanec (správa rezervací, provozní informace), Správce centra (plná správa systému, cen, reportů).
- **Technické požadavky:** Relační databáze (PostgreSQL), intuitivní a responzivní uživatelské rozhraní (Next.js, React, shadcn/ui, Tailwind CSS), zabezpečení (šifrovaná komunikace, NextAuth.js), verzování (Git), kontejnerizace (Docker).

3 Uživatelská dokumentace

3.1 Základní popis používané aplikace

Tato aplikace slouží ke komplexní správě rezervací sportovního centra "ActiveLife". Umožňuje uživatelům prohlížet dostupná sportoviště a aktivity, vytvářet a spravovat své rezervace online. Zaměstnancům a správcům centra poskytuje nástroje pro efektivní řízení provozu, správu kapacit, cen a generování reportů. Cílem je zjednodušit rezervační proces a poskytnout přehledné informace všem zúčastněným stranám.

3.2 Instalace

Aplikace je navržena pro spuštění pomocí Dockeru a Docker Compose.

1. Naklonujte repozitář projektu.
2. V kořenovém adresáři projektu spusťte příkaz `docker-compose up -d`. Tím se spustí potřebné služby (aplikační server, databáze) v kontejnerech.

3. Databáze se inicializuje automaticky včetně migrací a základních dat (seed).
4. Pro lokální vývoj bez Dockeru je potřeba mít nainstalovaný Node.js (včetně PNPM) a PostgreSQL. Závislosti se instalují pomocí `pnpm install`. Databáze se nastaví pomocí Prisma migrací (`pnpm prisma migrate dev`) a seedování (`pnpm prisma db seed`).
5. Aplikace bude dostupná na adrese `http://localhost:3000` (nebo dle konfigurace).

3.3 Přístupová oprávnění

Aplikace využívá systém rolí pro řízení přístupu k funkcím. Existují tři základní role:

- **Běžný uživatel:** Může se registrovat, přihlásit, spravovat svůj profil, prohlížet sportoviště/aktivity a vytvářet/spravovat vlastní rezervace.
- **Zaměstnanec:** Má práva běžného uživatele a navíc může spravovat všechny rezervace, zadávat rezervace manuálně a vidět přehledy směn.
- **Správce centra:** Má nejvyšší oprávnění, včetně všech práv zaměstnance, a navíc může spravovat uživatele, role, sportoviště, aktivity, ceny a generovat systémové reporty.

Pro testování jsou k dispozici následující ukázkové účty:

- Uživatel: `petr.svoboda@example.com`, Heslo: `user123`
- Zaměstnanec: `zamestnanec@activelife.cz`, Heslo: `zam123`
- Správce: `admin@activelife.cz`, Heslo: `admin123`

3.4 Použití aplikace

Aplikace je rozdělena do několika hlavních modulů dle uživatelských rolí:

- **Veřejná část:** Úvodní obrazovka, katalog sportovišť a aktivit, registrační a přihlašovací formulář.
- **Modul pro běžné uživatele:** Osobní účet, přehled vlastních rezervací, rezervační systém (kalendář, výběr slotů), historie.
- **Modul pro zaměstnance:** Správa denních rezervací, přehled obsazenosti, manuální zadávání rezervací, správa směn.
- **Modul pro správce centra:** Správa uživatelů a rolí, správa sportovišť a aktivit, cenová politika, systémová nastavení, generování reportů, plánování směn.

Základní kroky pro rezervaci:

1. Přihlaste se nebo se zaregistrujte.
2. Přejděte do sekce rezervací nebo katalogu sportovišť/aktivit.
3. Vyberte požadované sportoviště/aktivitu, datum a časový slot.
4. Potvrďte rezervaci.

4 Programová dokumentace

4.1 Datová část

Datová část aplikace je postavena na relační databázi PostgreSQL a spravována pomocí Prisma ORM. Model zahrnuje entity pro uživatele, role, zaměstnance, sportoviště, aktivity, časové sloty, rezervace, směny a další.

4.1.1 Analýza

Datový model byl navržen na základě analýzy požadavků sportovního centra „ActiveLife“. Cílem bylo vytvořit flexibilní a škálovatelnou strukturu pro správu rezervací.

Klíčové entity a vztahy Entitně-vztahový diagram (ERD) znázorňuje následující klíčové entity a jejich vztahy:

- **Uživatelé a role:** User-Reservation (1:N), User-Role (N:1)
- **Sportoviště:** Facility-TimeSlot (1:N), Facility-Activity (M:N)
- **Rezervace:** TimeSlot-Reservation (1:N)
- **Zaměstnanci:** Employee-EmployeeShift (1:N)

Uživatelské rozhraní Aplikace je navržena jako responzivní webová aplikace s moduly pro různé role uživatelů.

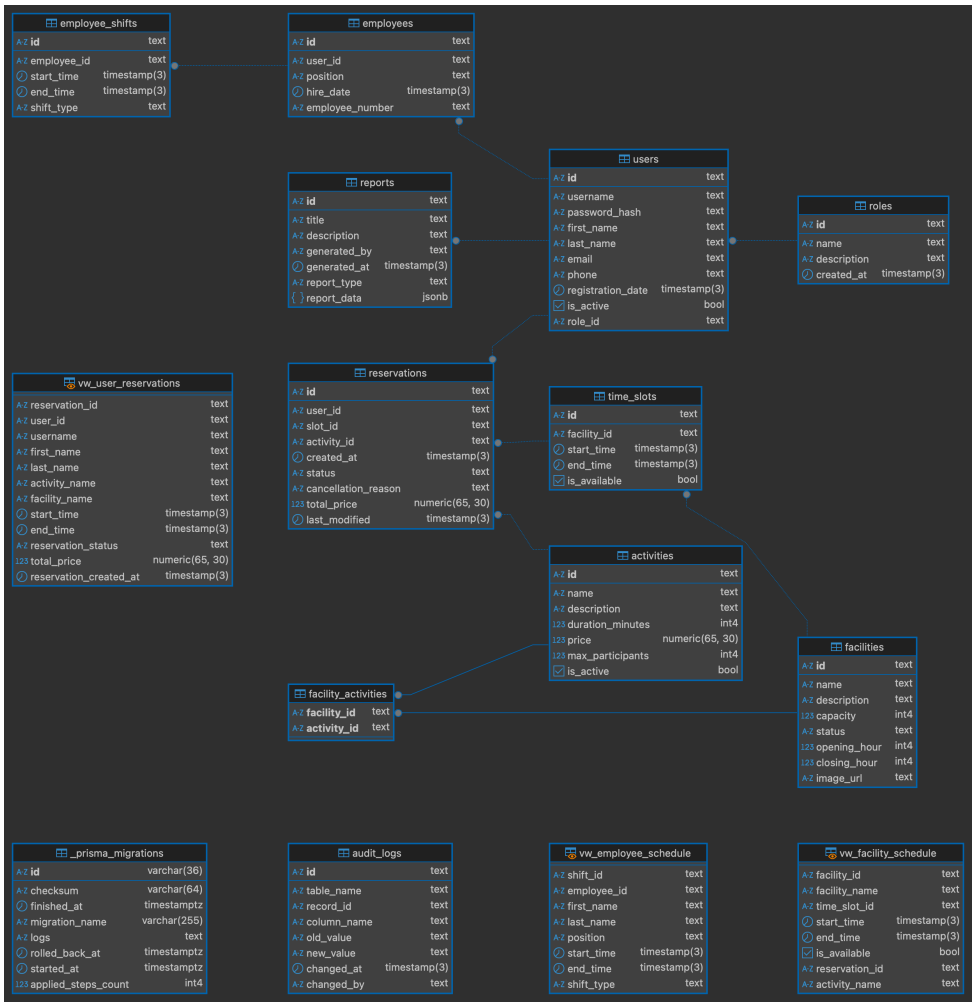
4.1.2 Fyzický model dat

Fyzický model dat je definován v souboru `prisma/schema.prisma` a v migračních SQL skriptech. Tento model zahrnuje definice všech tabulek, jejich polí, datových typů, relací (cizích klíčů) a indexů. Prisma zajišťuje mapování mezi tímto schématem a skutečnou strukturou databáze PostgreSQL.

Klíčové tabulky zahrnují `User`, `Role`, `Facility`, `Activity`, `TimeSlot`, `Reservation`, `EmployeeShift` a také spojovací tabulky pro M:N vztahy. Například propojení sportovišť a aktivit je realizováno pomocí tabulky `facility_activities`.

```
CREATE TABLE facility_activities (
  facility_id text NOT NULL,
  activity_id text NOT NULL,
  CONSTRAINT facility_activities_pkey PRIMARY KEY (facility_id, activity_id),
  CONSTRAINT facility_activities_activity_id_fkey FOREIGN KEY (activity_id)
      REFERENCES activities(id) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT facility_activities_facility_id_fkey FOREIGN KEY (facility_id)
      REFERENCES facilities(id) ON DELETE CASCADE ON UPDATE CASCADE
);
```

Datový slovník s popisem jednotlivých tabulek a sloupců je implicitně obsažen v `schema.prisma`.



Obrázek 1: Entitně-vztahový diagram (ERD) hlavních entit systému

4.1.3 Číselníky

Projekt využívá následující číselníky:

- **Role:** Tabulka `Role` definuje možné uživatelské role v systému:
  - `'ADMIN'` – Administrátor s plným přístupem
  - `'EMPLOYEE'` – Zaměstnanec s provozním přístupem
  - `'USER'` – Běžný uživatel se základním přístupem

  Zdroj: Definováno v `prisma/seed/seed-users.ts`
- **Status sportoviště:** Sloupec `status` v tabulce `Facility`:
  - `'ACTIVE'` – Sportoviště je aktivní a dostupné
  - `'MAINTENANCE'` – Sportoviště je v údržbě
  - `'CLOSED'` – Sportoviště je uzavřeno

  Zdroj: Definováno v `prisma/seed/seed-sport-facilities.ts`
- **Status rezervace:** Sloupec `status` v tabulce `Reservation`:
  - `'confirmed'` – Potvrzená rezervace
  - `'pending'` – Čekající na potvrzení
  - `'cancelled'` – Zrušená rezervace

  Zdroj: Definováno v `prisma/seed/seed-reservations.ts`
- **Typ směny:** Sloupec `shift_type` v tabulce `EmployeeShift`:
  - `'Ranní'` – Ranní směna (typicky 8:00-16:00)
  - `'Odpolední'` – Odpolední směna (typicky 14:00-22:00)

  Zdroj: Definováno v `prisma/seed/seed-employee-shifts.ts`

Všechny číselníky jsou implementovány jako textové sloupce v databázi a jejich hodnoty jsou validovány na aplikační úrovni pomocí TypeScript typů a Zod schémat.

4.1.4 Pohledy

V databázi jsou definovány následující pohledy pro zjednodušení dotazů a reportingu (definované v `prisma/migrations/20250420125507_add_initial_views/migration.sql`):

- `vw_user_reservations`: Zobrazuje detailní informace o rezervacích včetně údajů o uživateli, aktivitě a sportovišti.
- `vw_facility_schedule`: Poskytuje přehled rozvrhu sportovišť včetně obsazenosti a plánovaných aktivit.
- `vw_employee_schedule`: Zobrazuje rozpis směn zaměstnanců s jejich osobními údaji.

```
CREATE VIEW vw_user_reservations AS
SELECT
    r.id as reservation_id,
    u.id as user_id,
    u.username,
    u.first_name,
    u.last_name,
    a.name as activity_name,
    f.name as facility_name,
    ts.start_time,
    ts.end_time,
    r.status as reservation_status,
    r.total_price,
    r.created_at as reservation_created_at
FROM reservations r
JOIN users u ON r.user_id = u.id
JOIN activities a ON r.activity_id = a.id
JOIN time_slots ts ON r.slot_id = ts.id
JOIN facilities f ON ts.facility_id = f.id;



CREATE VIEW vw_facility_schedule AS
SELECT
    f.id as facility_id,
    f.name as facility_name,
    ts.id as time_slot_id,
    ts.start_time,
    ts.end_time,
    ts.is_available,
    r.id as reservation_id,
    a.name as activity_name
FROM facilities f
JOIN time_slots ts ON f.id = ts.facility_id
LEFT JOIN reservations r ON ts.id = r.slot_id AND r.status != 'cancelled'
LEFT JOIN activities a ON r.activity_id = a.id;



CREATE VIEW vw_employee_schedule AS
SELECT
    es.id as shift_id,
    e.id as employee_id,
    u.first_name,
    u.last_name,
    e.position,
    es.start_time,
    es.end_time,
    es.shift_type
FROM employee_shifts es
JOIN employees e ON es.employee_id = e.id
JOIN users u ON e.user_id = u.id;
```

4.1.5 Funkce

Projekt využívá následující databázové funkce (definované v `prisma/migrations/20250420131104_add_db_functions/migration.sql`):

- `calculate_facility_revenue`: Kalkuluje celkové příjmy pro dané sportoviště v zadaném období z potvrzených rezervací.
- `check_user_active_reservations`: Vrací počet aktivních (budoucích potvrzených nebo čekajících) rezervací pro daného uživatele.
- `get_facility_availability_summary`: Poskytuje textový souhrn dostupných vs. celkových časových slotů pro dané sportoviště a den.

```
CREATE OR REPLACE FUNCTION calculate_facility_revenue(
    p_facility_id TEXT,
    p_start_date DATE,
    p_end_date DATE
)
RETURNS DECIMAL AS $$
DECLARE
    total_revenue DECIMAL;
BEGIN
    SELECT COALESCE(SUM(r.total_price), 0.00)
    INTO total_revenue
    FROM reservations r
    JOIN time_slots ts ON r.slot_id = ts.id
    WHERE ts.facility_id = p_facility_id
      AND r.status = 'confirmed'
      AND ts.start_time >= p_start_date::timestamp
      AND ts.start_time < (p_end_date + interval '1 day')::timestamp;

    RETURN total_revenue;
END;
$$ LANGUAGE plpgsql;
```

```
CREATE OR REPLACE FUNCTION check_user_active_reservations(
    p_user_id TEXT
)
RETURNS INTEGER AS $$
DECLARE
    active_count INTEGER;
BEGIN
    SELECT COUNT(*)
```

```
    INTO active_count
    FROM reservations r
    JOIN time_slots ts ON r.slot_id = ts.id
    WHERE r.user_id = p_user_id
      AND r.status IN ('confirmed', 'pending')
      AND ts.start_time >= NOW();

    RETURN active_count;
END;
$$ LANGUAGE plpgsql;



CREATE OR REPLACE FUNCTION get_facility_availability_summary(
    p_facility_id TEXT,
    p_check_date DATE
)
RETURNS TEXT AS $$
DECLARE
    facility_name TEXT;
    total_slots INTEGER;
    available_slots INTEGER;
    summary TEXT;
BEGIN
    SELECT name INTO facility_name FROM facilities WHERE id = p_facility_id;

    IF NOT FOUND THEN
        RETURN 'Sportoviste nenalezeno.';
    END IF;

    SELECT COUNT(*)
    INTO total_slots
    FROM time_slots ts
    WHERE ts.facility_id = p_facility_id
      AND ts.start_time >= p_check_date::timestamp
      AND ts.start_time < (p_check_date + interval '1 day')::timestamp;

    SELECT COUNT(*)
    INTO available_slots
    FROM time_slots ts
    WHERE ts.facility_id = p_facility_id
      AND ts.start_time >= p_check_date::timestamp
      AND ts.start_time < (p_check_date + interval '1 day')::timestamp
      AND ts.is_available = TRUE;

    summary := available_slots::TEXT || '/' || total_slots::TEXT ||
               ' slotu volnych dne ' || to_char(p_check_date, 'DD.MM.YYYY');

    RETURN summary;
END;
$$ LANGUAGE plpgsql;
```

Tyto funkce jsou volány z aplikace pomocí `prisma.$queryRaw`.

4.1.6 Uložené procedury

Projekt využívá následující uložené procedury (definované v `prisma/migrations/20250420132146_add_stored_procedures/migration.sql`):

- `cancel_reservation(p_reservation_id UUID, p_cancellation_reason TEXT)`: Aktualizuje status rezervace na `'CANCELLED'` a zaznamená důvod zrušení.
- `deactivate_user(p_user_id UUID)`: Nastaví příznak `is_active` uživatele na `false`. (Použito triggerem `trg_user_deactivation`).
- `assign_employee_shift(p_employee_id UUID, p_start_time TIMESTAMP WITH TIME ZONE, p_end_time TIMESTAMP WITH TIME ZONE, p_shift_type TEXT)`: Vloží novou pracovní směnu pro zaměstnance.

Tyto procedury jsou volány z aplikace pomocí `prisma.$executeRaw` nebo `prisma.$executeRawUnsafe`.

```
CREATE OR REPLACE PROCEDURE cancel_reservation(
    p_reservation_id TEXT,
    p_cancellation_reason TEXT
)
LANGUAGE plpgsql
AS $$
BEGIN
    UPDATE reservations
    SET status = 'cancelled',
        cancellation_reason = p_cancellation_reason,
        last_modified = NOW()
    WHERE reservation_id = p_reservation_id;
END;
$$;
```

```
CREATE OR REPLACE PROCEDURE deactivate_user(
    p_user_id TEXT
)
LANGUAGE plpgsql
AS $$
BEGIN
    UPDATE users
    SET is_active = false
    WHERE id = p_user_id;
END;
$$;
```

```
CREATE OR REPLACE PROCEDURE assign_employee_shift(
    p_employee_id TEXT,
    p_start_time TIMESTAMP WITH TIME ZONE,
    p_end_time TIMESTAMP WITH TIME ZONE,
    p_shift_type TEXT
)
LANGUAGE plpgsql
AS $$
BEGIN
    INSERT INTO employee_shifts (shift_id, employee_id, start_time, end_time,
        shift_type)
    VALUES (gen_random_uuid()::text, p_employee_id, p_start_time, p_end_time,
        p_shift_type);
END;
$$;
```

4.1.7 Sekvence

Projekt nevyužívá explicitně definované sekvence pro generování číselných ID. Místo toho jsou primární klíče implementovány jako UUID, které jsou generovány následujícími způsoby:

- Většina tabulek používá UUID generované pomocí Prisma `@default(uuid())` nebo přímo v PostgreSQL pomocí `gen_random_uuid()`.
- Tabulka `audit_logs` používá `@default(dbgenerated("gen_random_uuid()"))` pro přímé generování UUID v databázi.
- V seed datech jsou UUID generována pomocí `randomUUID()` z Node.js crypto modulu.

Toto řešení bylo zvoleno pro zajištění globální unikátnosti identifikátorů napříč celou aplikací a pro usnadnění případné distribuce nebo replikace dat v budoucnu.

4.2 Aplikace

Aplikační část je postavena na frameworku Next.js s využitím App Routeru. Frontend využívá React, TypeScript a komponentovou knihovnu shadcn/ui s Tailwind CSS pro stylování. Backendová logika je řešena pomocí API routes v Next.js a interaguje s databází přes Prisma ORM.

4.2.1 Použité prostředí

- **Framework:** Next.js (v14+ s App Router)
- **Jazyk:** TypeScript

- **Frontend:** React (v18+), shadcn/ui, Tailwind CSS, Zustand (pro state management), React Hook Form (pro formuláře), Sonner (pro notifikace)
- **Backend (API Routes):** Node.js (runtime Next.js)
- **Databáze:** PostgreSQL (v15+)
- **ORM:** Prisma (v5+)
- **Autentizace/Autorizace:** NextAuth.js (v5+)
- **Package Manager:** PNPM
- **Kontejnerizace:** Docker, Docker Compose
- **Verzování:** Git
- **Linting/Formátování:** ESLint, Prettier

4.2.2 Řízení uživatelských účtů

Správa uživatelských účtů a autentizace je řešena pomocí knihovny NextAuth.js. Při registraci se ukládají údaje uživatele do tabulky `User`, včetně hashe hesla (používá se bcrypt). Každý uživatel má přiřazenou roli (odkaz na tabulku `Role`), která určuje jeho oprávnění (Role-Based Access Control - RBAC). Middleware v Next.js (`src/middleware.ts`) a kontroly na úrovni API routes a serverových komponent ověřují autentizaci a autorizaci uživatele pro přístup k chráněným zdrojům a funkcím na základě jeho role.

4.2.3 Moduly

Aplikace je strukturována do modulů, které odpovídají hlavním funkčním oblastem a uživatelským rolím:

- **Autentizace (`src/app/(auth)`):** Registrace, přihlášení, odhlášení a správa autentizace uživatelů.
- **Správa účtu (`src/app/app/account`):** Úprava profilu a nastavení uživatelského účtu.
- **Zaměstnanecký modul (`src/app/app/employee`):** Správa rezervací a směn pro zaměstnance.
- **Správa sportovišť (`src/app/app/facilities`):** Zobrazení seznamu, detailu, dostupnosti, vytváření a editace sportovišť.
- **Rezervace (`src/app/app/reservations`):** Vytváření a správa rezervací, přehled vlastních a všech rezervací.
- **Správa uživatelů (`src/app/app/users`):** Správa uživatelských účtů a jejich oprávnění.

- **API Routes (`src/app/api`):** Backend logika zahrnující:
  - `/api/activities`: Správa sportovních aktivit
  - `/api/auth`: Autentizační endpointy
  - `/api/facilities`: CRUD operace pro sportoviště
  - `/api/reservations`: Správa rezervací
  - `/api/time-slots`: Správa časových slotů
  - `/api/user`: Správa uživatelského profilu
- **Komponenty (`src/components`):** Znovu použitelné UI komponenty:
  - `auth/`: Komponenty pro autentizaci
  - `facilities/`: Komponenty pro správu sportovišť
  - `profile/`: Komponenty pro uživatelský profil
  - `reservations/`: Komponenty pro rezervace
  - `ui/`: Obecné UI komponenty postavené na shadcn/ui
  - `users/`: Komponenty pro správu uživatelů
- **Knihovny a utility (`src/lib`):** Sdílené funkce a konfigurace:
  - Databázový klient (Prisma)
  - TypeScript typy a rozhraní
  - Utility funkce
  - Konfigurace NextAuth.js
- **Middleware (`src/middleware.ts`):** Implementace autentizace a autorizace na úrovni routování.

4.2.4 Formuláře

Aplikace využívá React Hook Form s Zod schématy pro validaci formulářů. Všechny formuláře jsou implementovány s důrazem na uživatelskou přívětivost a validaci dat. Klíčové formuláře zahrnují:

- **Autentizační formuláře (`src/components/ui/auth-form.tsx`):**
  - Přihlašovací formulář: Email (validace formátu), heslo (min. 6 znaků)
  - Registrační formulář: Uživatelské jméno (3-20 znaků), jméno, příjmení, email, heslo, telefon (volitelný)
- **Formulář pro správu sportovišť (`src/components/facilities/facility-form.tsx`):**
  - Název (min. 3 znaky)

  - Popis (volitelný)
  - Kapacita (kladné číslo)
  - Status (aktivní/v údržbě/uzavřeno)
  - Otevírací a zavírací hodina (0-23, validace vzájemného vztahu)
  - URL obrázku (volitelné, validace formátu URL)
- **Formulář pro manuální rezervace (`src/components/reservations/ManualReservationDial tsx`):**
  - Vyhledávání/výběr uživatele s možností vytvoření nového
  - Výběr sportoviště a aktivity (dynamicky načítané)
  - Výběr data a časového slotu
  - Interní poznámky (volitelné)
  - Komplexní validace dostupnosti a oprávnění
- **Formuláře pro správu uživatelů:**
  - Editace uživatele (`src/components/users/edit-user-form.tsx`)
  - Přidání uživatele (`src/components/users/add-user-form.tsx`)
- **Formuláře pro správu profilu:**
  - Editace profilu (`src/components/profile/edit-profile-form.tsx`)
  - Změna hesla (`src/components/profile/change-password-form.tsx`)

Všechny formuláře implementují:

- Validaci na straně klienta pomocí Zod schémat
- Okamžitou zpětnou vazbu při chybách
- Ošetření stavů načítání a chyb
- Notifikace o úspěchu/neúspěchu pomocí knihovny Sonner
- Responzivní design pomocí shadcn/ui komponent
- Přístupnost (ARIA atributy, klávesová ovladatelnost)

4.2.5 Orientace ve zdrojovém kódu

Zdrojový kód projektu je organizován ve složce `src` a následuje konvence Next.js App Routeru:

- **`src/app/`:** Hlavní složka aplikace.
  - `(auth)/`: Route group pro autentizační stránky (login, register).

  - `api/`: Backend API routes. Každá podsložka obvykle odpovídá entitě nebo funkční oblasti (např. `api/facilities`, `api/reservations`). Obsahuje `route.ts` soubory s handlery pro HTTP metody (GET, POST, PUT, DELETE).
  - `app/`: Route group pro chráněné části aplikace dostupné po přihlášení.
    * `layout.tsx`: Hlavní layout pro přihlášené uživatele.
    * `page.tsx`: Výchozí stránka po přihlášení (dashboard).
    * Podsložky odpovídají hlavním sekcím aplikace (`facilities`, `reservations`, `account`, `users`). Stránky jsou definovány v `page.tsx`, specifické layouty v `layout.tsx`. Dynamické segmenty (např. `[id]`) slouží pro detailní stránky.

- `src/components/`: Znovu použitelné React komponenty.
  - `ui/`: Komponenty z shadcn/ui (generované CLI).
  - `auth/`, `facilities/`, `profile/`, etc.: Aplikačně specifické komponenty.

- `src/lib/`: Sdílené knihovny, utility a typy.
  - `db.ts`: Inicializace a export Prisma klienta.
  - `types.ts`: Definice TypeScript typů a rozhraní.
  - `utils.ts`: Pomocné funkce.
  - `auth.ts`: Konfigurace NextAuth.js.

- `src/hooks/`: Vlastní React hooky.

- `src/providers/`: React context providers (např. pro session, theme).

- `prisma/`: Soubory související s Prisma ORM.
  - `schema.prisma`: Definice datového modelu.
  - `migrations/`: Složka s SQL migračními soubory.
  - `seed.ts`: Skript pro naplnění databáze počátečními daty.

- `public/`: Statické soubory (obrázky, fonty).

- **Kořenový adresář:** Konfigurační soubory (`next.config.ts`, `tsconfig.json`, `package.json`, `compose.yml`, `.env`, etc.).

Kód je formátován pomocí Prettier (v3.5.3) a kvalita je kontrolována pomocí ESLint (v9) s následující konfigurací:

- **ESLint (`eslint.config.mjs`):**
  - Využívá nový formát flat config
  - Rozšiřuje konfigurace `next/core-web-vitals` a `next/typescript`
  - Integrován s Next.js pro kontrolu typů a výkonu

- **Prettier (`.prettierrc`):**
  - Maximální délka řádku: 80 znaků

  - Odsazení: 2 mezery (taby)
  - Jednoduché uvozovky pro řetězce
  - Středníky na konci příkazů
  - Bez čárek na konci posledního prvku
  - Pluginy pro formátování Tailwind CSS a package.json

- **Ignorované soubory (`.prettierignore`):**
  - Závislosti (`node\_modules/`, `.pnp`, `.pnp.js`, `pnpm-lock.yaml`)
  - Vývojová prostředí (`.idea/`, `.vscode/`)
  - Buildové výstupy (`.next/`, `out/`, `build/`)
  - Prostředí (`.env*`)
  - TypeScript deklarace (`next-env.d.ts`)

- **NPM skripty:**
  - `pnpm lint`: Spuštění ESLint kontroly
  - `pnpm build`: Zahrnuje typovou kontrolu TypeScriptu

Tato konfigurace zajišťuje konzistentní formátování kódu napříč projektem a pomáhá předcházet běžným chybám a problémům s kvalitou kódu.

5 Závěr

Databázová aplikace pro správu rezervací sportovního centra ActiveLife byla úspěšně implementována s využitím moderních technologií a postupů. Projekt splnil všechny stanovené požadavky a přináší následující klíčové funkcionality:

5.1 Implementované funkce

- **Správa uživatelů a rolí** – Komplexní systém uživatelských rolí (administrátor, zaměstnanec, běžný uživatel) s odpovídajícími oprávněními
- **Rezervační systém** – Intuitivní proces rezervace sportovišť s podporou časových slotů a různých aktivit
- **Správa sportovišť** – Evidence a správa sportovních zařízení včetně jejich kapacit a provozní doby
- **Reportování** – Generování reportů pro management s využitím vlastních databázových funkcí a pohledů
- **Audit změn** – Automatické sledování změn v systému pomocí triggerů a audit logu

5.2 Technické aspekty

- **Databázová vrstva**
  - Implementace 11 hlavních databázových tabulek
  - 3 pohledy pro optimalizaci častých dotazů
  - 3 vlastní funkce pro výpočet revenue, kontrolu rezervací a dostupnosti
  - 3 uložené procedury pro správu rezervací a uživatelů
  - 2 triggery pro audit změn a automatickou správu rezervací
  - Optimalizované indexy pro rychlé vyhledávání
- **Aplikační vrstva**
  - Moderní webová aplikace postavená na Next.js 14 s App Routerem
  - Responzivní UI využívající shadcn/ui komponenty
  - Robustní validace formulářů pomocí Zod a React Hook Form
  - Bezpečná autentizace a autorizace s NextAuth.js
  - Typově bezpečné API endpointy s Prisma ORM

5.3 Dosažené cíle

Projekt úspěšně dosáhl následujících cílů:

- Nahrazení manuálního systému evidence moderním digitálním řešením
- Zefektivnění procesu rezervací a správy sportovišť
- Implementace všech požadovaných databázových objektů a funkcí
- Vytvoření intuitivního uživatelského rozhraní
- Zajištění bezpečnosti a auditovatelnosti systému

5.4 Budoucí rozšíření

Systém byl navržen s ohledem na možná budoucí rozšíření, mezi která patří:

- Implementace platebního systému pro online platby
- Rozšíření reportovacích funkcí
- Mobilní aplikace pro snadnější přístup
- Integrace s externími kalendářovými systémy

Projekt demonstruje praktické využití pokročilých databázových konceptů v reálné aplikaci a poskytuje solidní základ pro další rozvoj systému. Díky použití moderních technologií a důrazu na kvalitu kódu je aplikace připravena na budoucí rozšíření a dlouhodobou údržbu.

A Přílohy

A.1 Inicializace databáze

Databáze je plně verzována a inicializována pomocí následujících nástrojů:

- **Prisma Migrate** (`pnpm prisma migrate dev`) – Spravuje schéma databáze a jeho změny:
  - Vytváří všechny tabulky, pohledy, funkce, procedury a triggery
  - Migrační soubory jsou verzovány v `prisma/migrations/`
  - Zajišťuje bezpečnou aplikaci změn schématu v určeném pořadí
- **Prisma Seed** (`pnpm prisma db seed`) – Inicializuje testovací data:
  - Vytváří výchozí role a uživatelské účty
  - Naplňuje databázi ukázkovými sportovišti a aktivitami
  - Generuje testovací rezervace a směny
  - Seed skripty jsou verzovány v `prisma/seed/`

Díky tomuto přístupu není potřeba udržovat samostatné zálohy databáze, protože:

- Struktura databáze je plně definována v migracích
- Testovací data jsou reprodukovatelná pomocí seed skriptů
- Vše je verzováno v Gitu spolu se zdrojovým kódem
- Produkční data by měla být zálohována na úrovni infrastruktury

A.2 Zdrojové kódy aplikace

Kompletní zdrojové kódy aplikace jsou dostupné v GitHub repozitáři na adrese https://github.com/Kedlub/dbs-dataforge.

Projekt je vyvíjen týmem DataForge a je verzován pomocí Gitu. Pro lokální vývoj je k dispozici dokumentace v `README.md`, která obsahuje instrukce pro instalaci a spuštění vývojového prostředí.