Specyfikacja systemu wypożyczalni maszyn rolniczych

Autor: Bartłomiej Liber

Projekt również dostępny w repozytorium github: https://github.com/bertck/projekt1_PAI

Wstęp / Cel projektu

System wspiera spółdzielnię rolników, która dzieli między sobą wspólnie zakupione maszyny. Aplikacja umożliwia przeglądanie maszyn, zarządzanie kontami użytkowników i rezerwowanie sprzętu na całe dni zgodnie z zasadą „kto pierwszy, ten lepszy".

Zakres funkcjonalności

- Rejestracja i logowanie użytkowników.
- Podział na zwykłych użytkowników oraz administratorów.
- Przeglądanie listy maszyn.
- Dodawanie, modyfikacja i usuwanie maszyn (administrator).
- Przeglądanie rezerwacji maszyny.
- Tworzenie, modyfikacja i usuwanie rezerwacji (tylko własnych, z wyjątkiem administratora).
- Przegląd i administracja kontami użytkowników (administrator).

Architektura systemu

Aplikacja wykorzystuje architekturę warstwową:

1. **Warstwa prezentacji** – szablony Pug renderowane po stronie serwera (SSR).
2. **Warstwa logiki biznesowej** – trasy Express, walidacja danych, reguły dostępu.
3. **Warstwa dostępu do danych** – ORM Sequelize komunikujący się z relacyjną bazą SQLite.

Sesje HTTP przechowują identyfikator użytkownika; dane użytkownika są dostępne w `res.locals.currentUser`.

Technologie i narzędzia

- Node.js LTS
- Express 5
- Sequelize 6 + sterownik sqlite3
- Pug (silnik widoków)
- Bcryptjs – haszowanie haseł
- dotenv – konfiguracja środowiskowa

Model bazy danych

Tabela `users`

Pole	Typ	Ograniczenia	Opis
`id`	INTEGER	PK, autoinkrementacja	Identyfikator użytkownika
`email`	STRING	unikalne, not null, format e-mail	Adres e-mail
`username`	STRING(3-20)	unikalne, not null	Nazwa użytkownika
`passwordHash`	STRING	not null	Hasło w postaci skrótu
`role`	ENUM('user','admin')	not null, domyślnie `user`	Rola w systemie

Tabela `machines`

Pole	Typ	Ograniczenia	Opis
`id`	INTEGER	PK, autoinkrementacja	Identyfikator maszyny
`name`	STRING(5-20)	not null	Nazwa maszyny
`description`	TEXT	max 100 znaków	Opcjonalny opis

Tabela `reservations`

Pole	Typ	Ograniczenia	Opis
`id`	INTEGER	PK, autoinkrementacja	Identyfikator rezerwacji
`startDate`	DATEONLY	not null	Data rozpoczęcia rezerwacji
`endDate`	DATEONLY	not null, >= `startDate`	Data zakończenia rezerwacji
`userId`	INTEGER	FK -> `users.id`, CASCADE on delete	Rezerwujący użytkownik
`machineId`	INTEGER	FK -> `machines.id`, CASCADE on delete	Rezerwowana maszyna

Dodatkowe reguły biznesowe:

- Rezerwacje można składać maksymalnie z trzymiesięcznym wyprzedzeniem.
- Zakres dat nie może nachodzić na istniejącą rezerwację tej samej maszyny.

Interfejs REST

Uwierzytelnianie

Metoda	Endpoint	Dostęp	Opis
GET	`/auth/login`	publiczny	Formularz logowania
POST	`/auth/login`	publiczny	Logowanie użytkownika
GET	`/auth/logout`	zalogowani	Wylogowanie
GET	`/auth/register`	publiczny	Formularz rejestracji
POST	`/auth/register`	publiczny	Założenie konta

Maszyny

Metoda	Endpoint	Dostęp	Opis
GET	`/machines`	zalogowani	Lista maszyn
GET	`/machines/new`	administrator	Formularz dodawania maszyny
POST	`/machines`	administrator	Utworzenie maszyny
GET	`/machines/:id`	zalogowani	Szczegóły maszyny
GET	`/machines/:id/edit`	administrator	Formularz edycji
PUT	`/machines/:id`	administrator	Aktualizacja danych maszyny
DELETE	`/machines/:id`	administrator	Usunięcie maszyny

Rezerwacje

Metoda	Endpoint	Dostęp	Opis
GET	`/reservations`	zalogowani	Lista rezerwacji; `machineId` w query filtruje po maszynie
GET	`/reservations/new`	zalogowani	Formularz tworzenia rezerwacji
POST	`/reservations`	zalogowani	Utworzenie rezerwacji dla bieżącego użytkownika
GET	`/reservations/:id`	zalogowani	Szczegóły rezerwacji
GET	`/reservations/:id/edit`	właściciel / admin	Formularz edycji
PUT	`/reservations/:id`	właściciel / admin	Aktualizacja rezerwacji
DELETE	`/reservations/:id`	właściciel / admin	Usunięcie rezerwacji

Użytkownicy

Metoda	Endpoint	Dostęp	Opis
GET	`/users`	administrator	Lista użytkowników
PUT	`/users/:id`	administrator	Zmiana nazwy lub roli użytkownika
DELETE	`/users/:id`	administrator	Usunięcie konta (nie można usunąć ostatniego admina)

Instrukcja uruchomienia

1. **Klonowanie repozytorium**

   ```
   git clone <adres_repozytorium>
   cd projekt1_PAI
   ```

2. **Instalacja zależności**

   ```
   npm install
   ```

3. **Konfiguracja środowiska** – opcjonalny plik `.env`:

   - `SERVER_PORT` – port HTTP (domyślnie 3000)
   - `SESSION_SECRET` – tajny klucz sesji
   - `DB_FILE_PATH` – ścieżka do pliku bazy danych
   - `ADMIN_EMAIL`, `ADMIN_USERNAME`, `ADMIN_PASSWORD` – dane początkowego administratora

4. **Uruchomienie aplikacji**

   ```
   npm start
   ```

   Serwer będzie dostępny pod adresem `http://localhost:3000`.

Kluczowe moduły

- `src/app.js` – konfiguracja Express, rejestracja tras i middleware.
- `src/models/` – definicje modeli Sequelize (`User`, `Machine`, `Reservation`).
- `src/routes/` – implementacja logiki biznesowej i interfejsu REST.
- `src/middlewares/` – kontrola dostępu i wczytywanie danych użytkownika.
- `src/db/` – konfiguracja i inicjalizacja bazy danych.
- `src/views/` – szablony Pug renderujące odpowiedzi HTML.

Bezpieczeństwo

- Logowanie oparte na sesjach HTTP; identyfikator użytkownika przechowywany jest w sesji.
- Hasła są przechowywane w postaci skrótów `bcrypt`.
- Middleware `ensureAuthenticated` wymaga zalogowania do większości zasobów.
- Middleware `ensureAdmin` ogranicza operacje administracyjne do roli `admin`.