Sprint 1, grupa B , raporty ze spotkań *Scrum Meetings*

Sprint Backlog

task1: Inicjalizacja środowiska – konfiguracja plików startowych projektu
task2: Utworzenie struktury frontendu (Vite + React + Tailwind + routing, App.tsx, main.tsx, konfiguracja routera)
task3: Przygotowanie struktury backendu (Express + routery, konfiguracja błędów)
task4: Implementacja bazy danych SQLite – konfiguracja typów i schematów (ORM)
task5: Konfiguracja GitHub – dodanie repozytorium i pliku README.md
task6: Opracowanie przepływu logowania i rejestracji – formularze i logika sesji
task7: Widok profilu użytkownika – dane + działające wylogowanie
task14: Formularz rezerwacji i logika widoku klienta
task16: Konfiguracja systemu do wysyłki wiadomości e-mail – potwierdzenia rezerwacji i pliki ICS
task17: Middleware Express Rate-Limit – ograniczenie (1 rezerwacja / 30 s na IP + e-mail)
task18: Widok „Moje rezerwacje" / Dashboard użytkownika
task19: Middleware autoryzacji API wg ról OWNER/WORKER + kontekstu companyId
task20: Panel ustawień firmy – interfejs właściciela
task32: Przygotowanie panelu administratora – układ i komponenty

Raport z pierwszego tygodnia realizacji sprintu 1

06.11.2025

Scrum Meeting 1 (27.10.2024)

Na pierwszym spotkaniu rozmawialiśmy o doprecyzowaniu pomysłu i tym, jak poukładać narzędzia pracy zespołowej. Ustaliliśmy, że idziemy w monorepo i omówiliśmy zasady pracy z Jira i GitHubem: jak ma wyglądać projekt i board w Jira (kolumny, workflow, etykiety), jak nazywamy zadania i gałęzie oraz jaki format commitów chcemy przyjąć. Ustaliliśmy też, że repozytorium na GitHubie będzie naszym centralnym miejscem pracy. W części technicznej uzgodniliśmy, że przygotujemy jedynie szkielet: frontend na Vite + React + Tailwind oraz backend na Expressie z ts-node, a w głównym package.json dodamy skrypty do uruchamiania obu serwerów. Uzgodniliśmy minimalne połączenie front–back (CORS i prosty health-check), żeby można było lokalnie włączyć środowisko i płynnie ruszyć z pracami w kolejnych tygodniach.

Scrum Meeting 2 (06.11.2025)

Na drugim spotkaniu skupiliśmy się na tym, jak rozwijać funkcjonalności na bazie przygotowanego szkieletu. Ustaliliśmy, że po stronie backendu potrzebujemy kluczowych middleware'ów (helmet, cors, sesje, parser JSON) oraz logowania żądań, a autoryzację opieramy na sesji: po zalogowaniu przechowujemy w sesji identyfikator, e-mail i rolę (USER/WORKER/OWNER/PLATFORM_ADMIN), żeby móc sprawnie weryfikować uprawnienia. Po stronie frontendu omówiliśmy pełny przepływ logowania i rejestracji: LoginPage z walidacjami, podglądem hasła, komunikatami i przekierowaniem zależnym od roli oraz RegisterPage z walidacją e-maila, wskaźnikiem siły hasła, potwierdzeniem hasła i akceptacją

regulaminu. Uzgodniliśmy też dodanie AuthContext do obsługi sesji i ProtectedRoute do kontroli dostępu oraz automatycznych przekierowań.

Zadania realizowane przez poszczególnych deweloperów:

Rafał Gola:

Realizowane zadanie	Tygodniowy czas pracy [h]	Status realizowanego zadania
Inicjalizacja środowiska – konfiguracja plików startowych projektu	4	ukończone
Konfiguracja GitHub, dodanie repozytorium i pliku	2	ukończone

Przemysław Habdas:

Realizowane zadanie	Tygodniowy czas pracy [h]	Status realizowanego zadania
Przygotowanie struktury backendu	3	Ukończone
Middleware autoryzacji API wg ról OWNER/WORKER + kontekstu companyId.	4	W realizacji
Middleware Express Rate-Limit (1 rezerwacja/30 s per IP + e-mail).	2	W realizacji

Kamil:

Realizowane zadanie	Tygodniowy czas pracy [h]	Status realizowanego zadania
Utworzenie struktury frontendu (Vite + React + Tailwind, `App.tsx`, `main.tsx`, konfiguracja routera)	5	Ukończone

Jakub:

Realizowane zadanie	Tygodniowy czas pracy [h]	Status realizowanego zadania
Opracowanie przepływu logowania i rejestracji – koncepcja widoków i logiki sesji`	2	Ukonczone
Widok profilu użytkownika z danymi i działającym wylogowaniem.	4	W realizacji

Daniel:

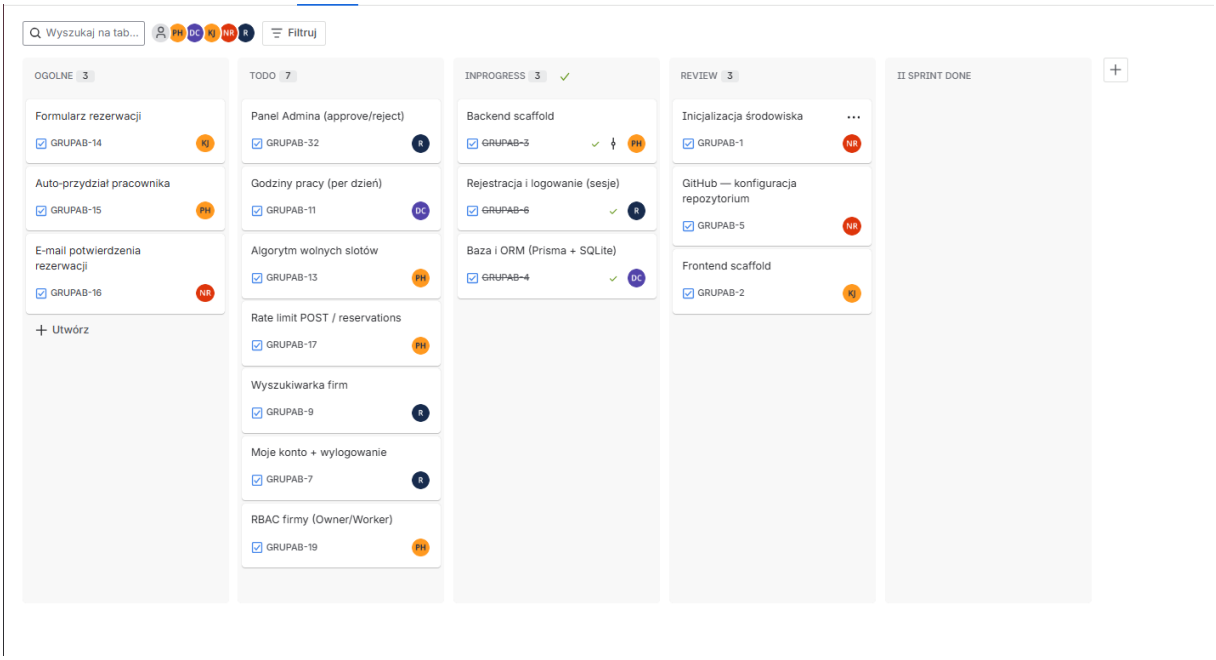
Realizowane zadanie	Tygodniowy czas pracy [h]	Status realizowanego zadania
Wstępna struktura plików pod przyszłą bazę i ORM	2	Ukonczone

Projekt zespołowy systemu informatycznego

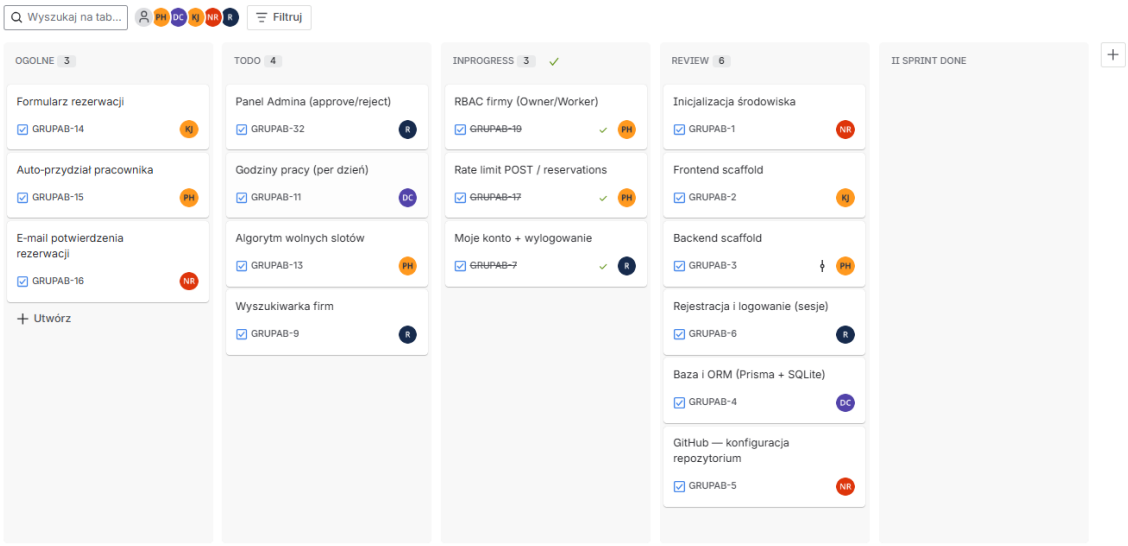
Cele osiągnięte w bieżącym tygodniu

- Ustaliliśmy strukturę projektu w architekturze monorepo (frontend + backend).
- Skonfigurowaliśmy środowisko deweloperskie i repozytorium GitHub.
- Uruchomiliśmy podstawowy szkielet aplikacji (Vite + React + Tailwind + Express).
- Ustaliliśmy zasady pracy zespołowej w Jira oraz format commitów.
- Połączyliśmy frontend z backendem testowym endpointem (health-check).
- Przygotowaliśmy wspólne pliki konfiguracyjne TypeScript.

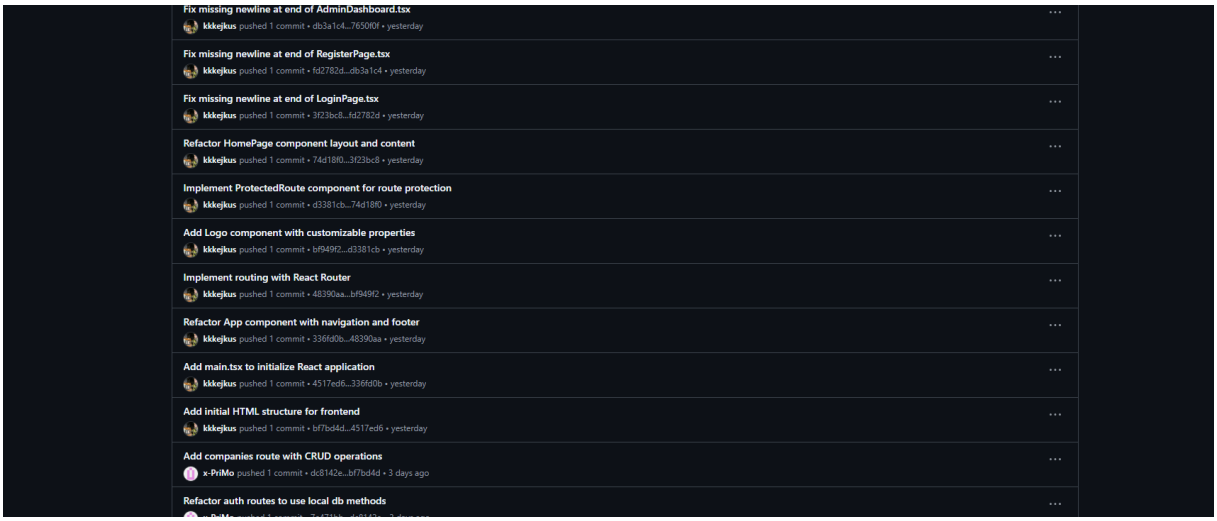
Stan tablicy Kanban na dzień 27.10.2024

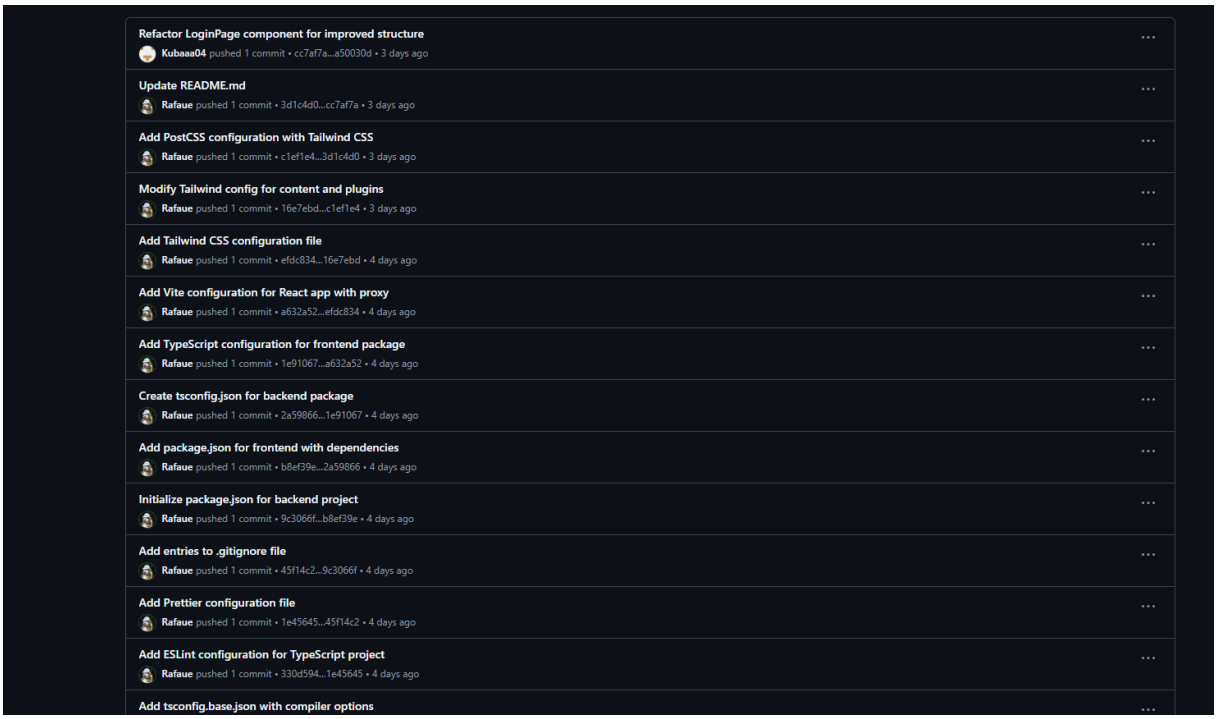


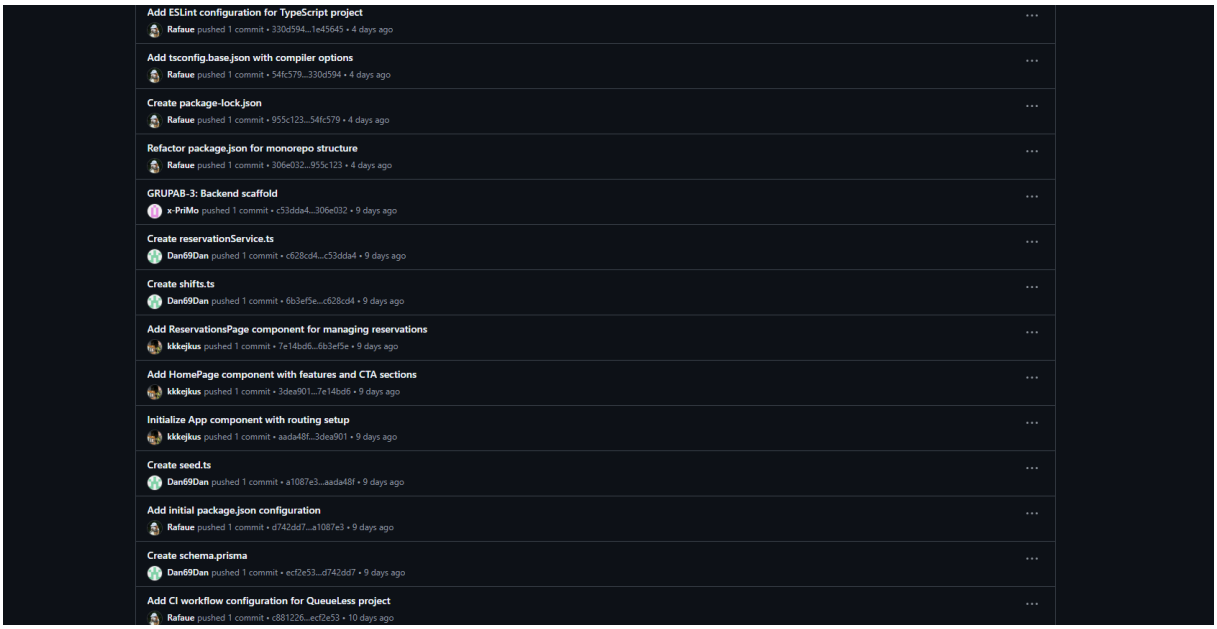
Stan tablicy Kanban na dzień 06.11.2025



Commity z Githuba – stan na dzień 06.11.2025

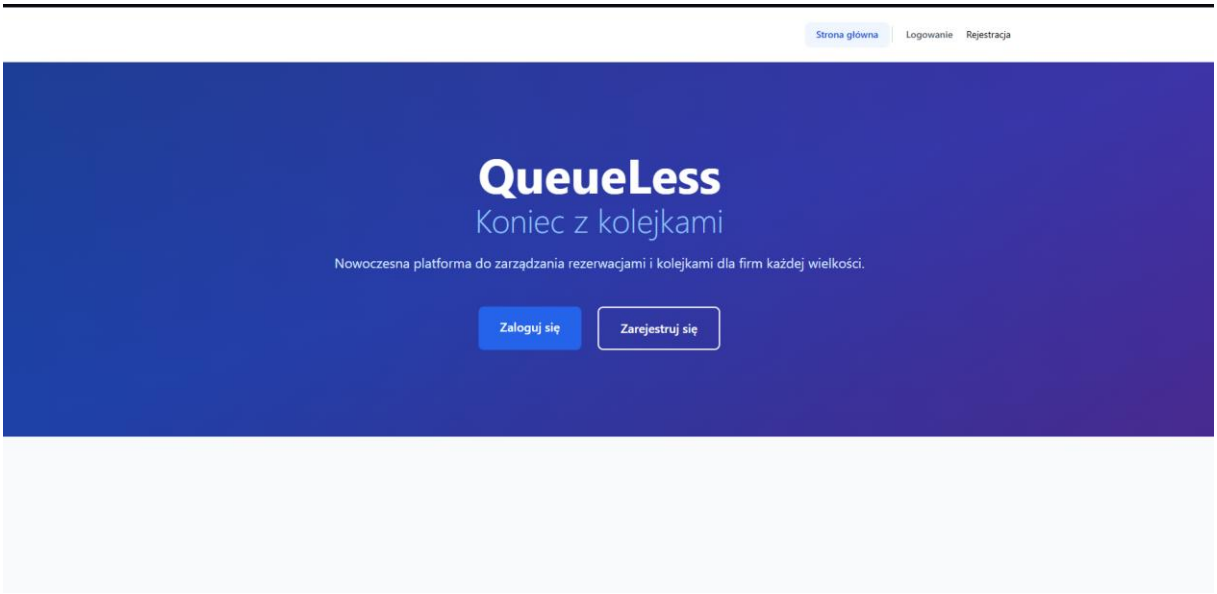




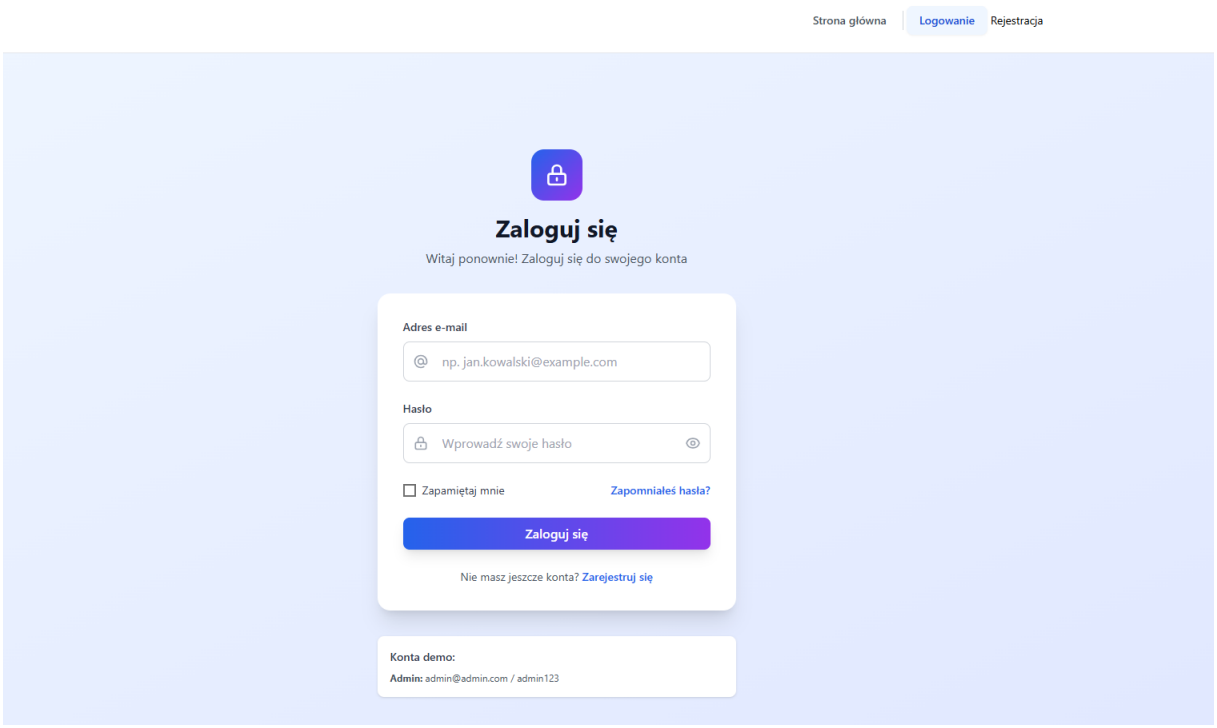


Implementowane widoki aplikacji

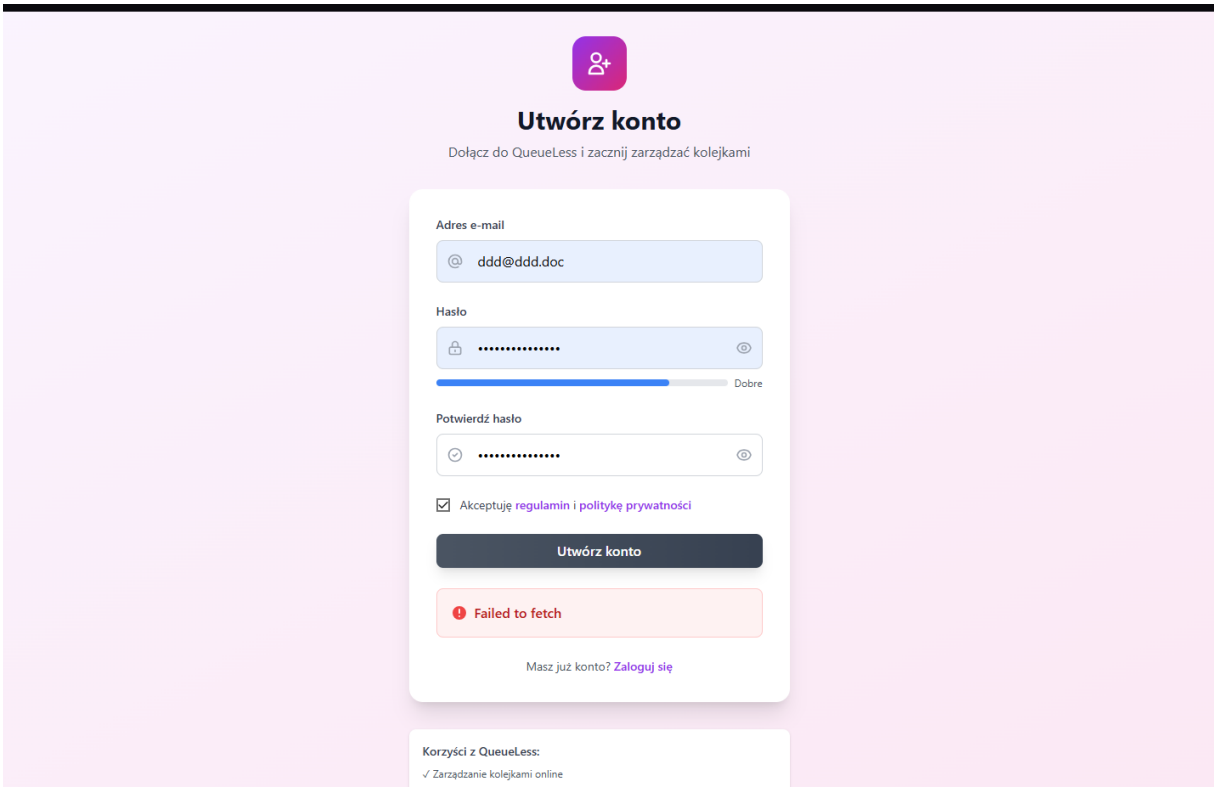
Ekran startowy:



Ekran logowania:



Ekran Rejestracji:

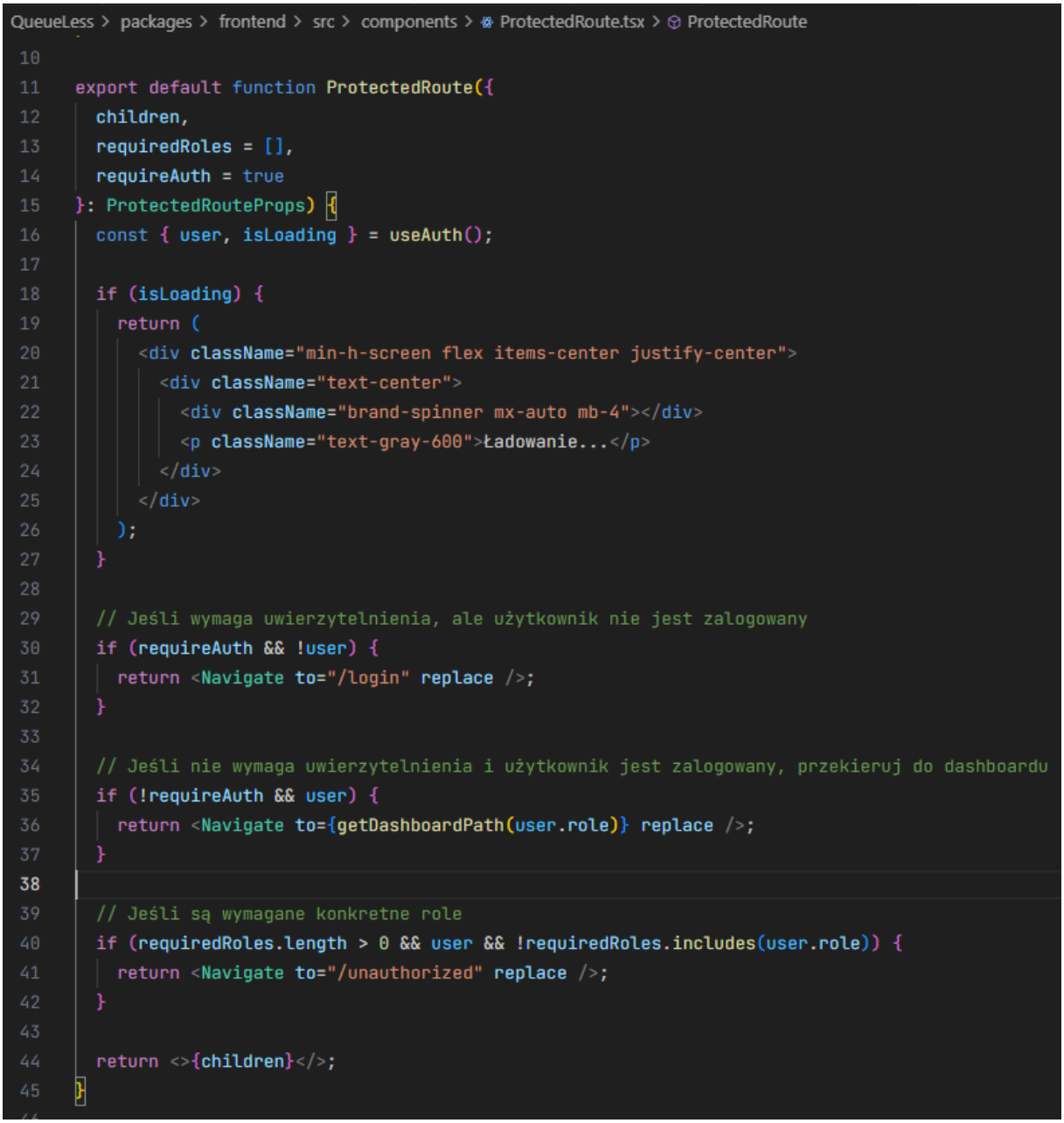


Kod źródłowy – istotne fragmenty opracowane w bieżącym tygodniu

```
QueueLess > packages > backend > src > routes > TS auth.ts > router.post('/register') callback
12
13  router.post('/register', async (req, res) => {
14    const parsed = registerSchema.safeParse(req.body);
15    if (!parsed.success) return res.status(400).json({ error: 'Invalid input' });
16
17    const { email, password } = parsed.data;
18
19    // Sprawdź czy użytkownik już istnieje
20    const existingUser = db.getUserByEmail(email);
21    if (existingUser) return res.status(409).json({ error: 'Email already registered' });
22
23    // Hashuj hasło i utwórz użytkownika
24    const hash = await bcrypt.hash(password, 10);
25    const user = db.createUser({
26      email,
27      password: hash,
28      role: 'USER',
29      isActive: true
30    });
31
```

Ten fragment kodu obsługuje rejestrację użytkownika w aplikacji. Sprawdza poprawność danych wejściowych, weryfikuje, czy podany e-mail nie jest już zarejestrowany, a następnie

hashuje hasło za pomocą bcrypts i tworzy nowego użytkownika z rolą „USER” i statusem aktywnym.



Ten fragment kodu odpowiada za ochronę tras w aplikacji — komponent ProtectedRoute sprawdza, czy użytkownik jest zalogowany i czy ma odpowiednią rolę dostępu. W zależności od wyniku przekierowuje go na stronę logowania, dashboard lub „unauthorized”, a jeśli wszystko jest poprawne, wyświetla żądany widok.

Projekt zespołowy systemu informatycznego

Raport z drugiego tygodnia realizacji sprintu 1

10.11.2025

Na meetingu 3 skoncentrowaliśmy się na tym, żeby logowanie działało po prostu bez stresu. Zaczęliśmy od przejrzenia ekranu i konsoli, bo część osób widziała komunikat „Not found". Potem sprawdziliśmy „ręcznie" połączenie z serwerem i szybko okazało się, że aplikacja woła pod inny adres niż ten, którego oczekuje backend. Ustaliliśmy prostą, konkretną poprawkę: przepisać ścieżkę w ustawieniach frontu tak, żeby trafiała we właściwe miejsce, zrestartować oba serwery i sprawdzić jeszcze raz cały przepływ. Zrobiliśmy krótki test po zmianie, żeby mieć pewność, że sesja się poprawnie zakłada, a użytkownik dostaje czytelną informację. Przy okazji uporządkowaliśmy detale na ekranie logowania: poprawne dane demo, prostsze komunikaty w błędnych przypadkach i sensowne podpowiedzi co wpisać. Ustaliliśmy też, że w najbliższych dniach ruszymy z pierwszymi „widocznymi" elementami — listą firm i szkicem panelu admina — tak, żeby każdy mógł zobaczyć postęp bez zaglądania w technikalia.

14.11.2025

Na meetingu 4 usiedliśmy do podziału ról i zaplanowaliśmy, jak ma wyglądać pierwsza, lekka wersja panelu administratora. Chcieliśmy, żeby była prosta w obsłudze: lista firm, widoczny status firmy (aktywna/wyłączona) i kilka przełączników, które od razu dają poczucie kontroli. Ustaliliśmy też „bezpieczniki" — żeby nikt nie mógł męczyć logowania w nieskończoność i żeby dane, które wpisujemy, były od razu sensownie sprawdzane. Zdecydowaliśmy, że przygotujemy podstawowy zestaw danych startowych (admin, jedna przykładowa firma, pracownik), dzięki czemu demo będzie od pierwszego dnia wyglądać „jak działa". Doprecyzowaliśmy, co dla nas znaczy „zrobione" w tej części (czyli jasna definicja gotowości), a na koniec ułożyliśmy prosty plan kontroli jakości: czy da się bez problemu wejść, czy sesja trzyma, czy komunikaty są zrozumiałe i czy aplikacja zachowuje się grzecznie w typowych błędach. Na bazie tego zrobiliśmy krótki plan następnego kroku: widok listy firm na froncie, prosty profil firmy z przyciskiem „Zarezerwuj", oraz pierwszy ekran „kolejki dnia" dla pracownika, gdzie można w prosty sposób zacząć i zakończyć wizytę. Dla użytkownika końcowego chcemy dorzucić przyjazne wiadomości (np. „sprawdź e-mail lub hasło"), kilka drobnych udogodnień na telefonie i szybkie potwierdzenie rezerwacji e-mailem z linkiem do kalendarza, żeby całość czuła się „gotowa do użycia".

Zadania realizowane przez poszczególnych deweloperów:

Rafał Gola:

Realizowane zadanie	Tygodniowy czas pracy [h]	Status realizowanego zadania
Konfiguracja GitHub oraz utworzenie repozytorium i pipeline CI (T5)	2	Ukończone
Konfiguracja systemu do wysyłki wiadomości e-mail – potwierdzenia rezerwacji i ICS (T16)	3	W realizacji

Przemysław Habdas:

Realizowane zadanie	Tygodniowy czas pracy [h]	Status realizowanego zadania
Przygotowanie struktury backendu (Express + routery, konfiguracja błędów) (T3)	2	Ukończone
Implementacja middleware autoryzacji w rolach OWNER/WORKER i kontekstu companyId (T19)	3	W realizacji
Middleware Express Rate-Limit (autoryzacja/rezerwacje) (T17)	2	W realizacji

Kamil:

Realizowane zadanie	Tygodniowy czas pracy [h]	Status realizowanego zadania
Utworzenie struktury frontendu (Vite + React + Tailwind + routing) (T2)	5	Ukończone
Formularz rezerwacji i logika widoku klienta (T14)	4	W realizacji
Widok „Moje rezerwacje” / Dashboard użytkownika (T18)	3	W realizacji
Panel ustawień firmy – interfejs właściciela (T20)	4	W realizacji

Jakub:

Realizowane zadanie	Tygodniowy czas pracy [h]	Status realizowanego zadania
Opracowanie przepływu logowania i rejestracji – formularze i logika sesji (T6)	2	Ukończone
Widok profilu użytkownika – dane + wylogowanie (T7)	2	W realizacji
Przygotowanie panelu administratora – układ i komponenty (T32, backlog Sprintu 2)	4	Zaplanowane

Daniel:

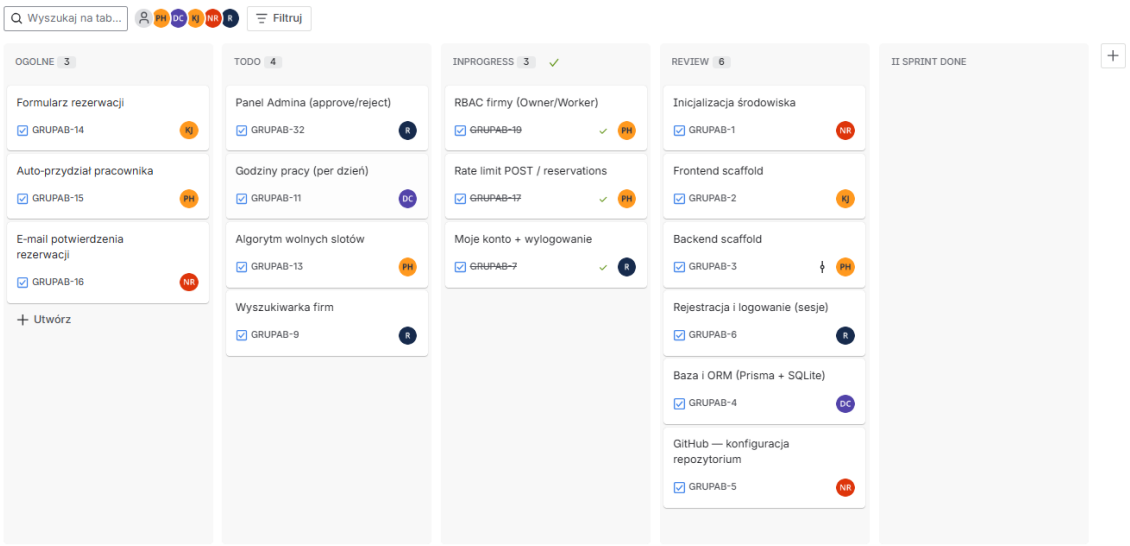
Realizowane zadanie	Tygodniowy czas pracy [h]	Status realizowanego zadania
Implementacja bazy danych SQLite– konfiguracja typów i schematów (T4)	4	W realizacji
Migracje i pliki inicjalizujące strukturę tabel oraz dane przykładowe (T4)	2	Ukończone

Projekt zespołowy systemu informatycznego

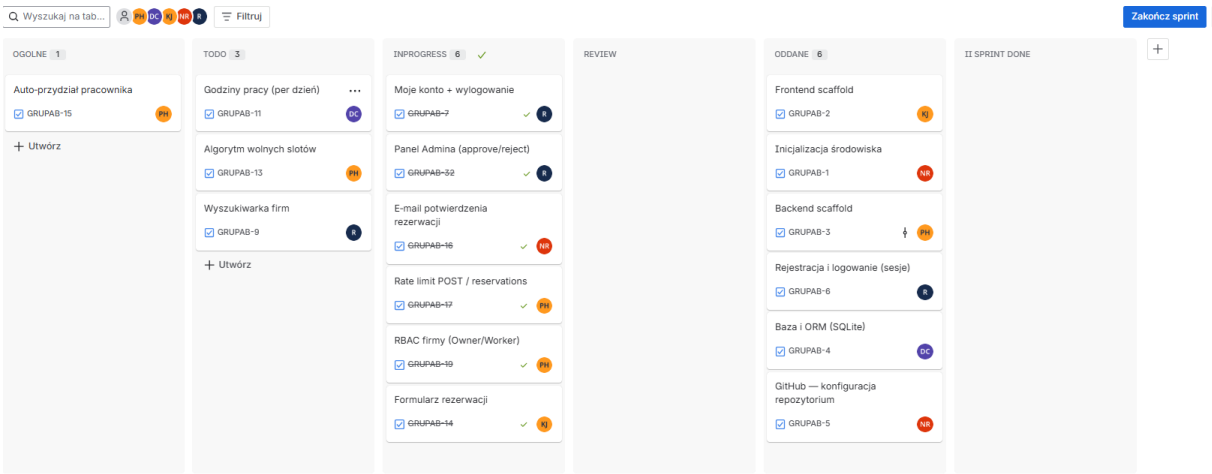
Cele osiągnięte w bieżącym tygodniu

- Zainicjalizowaliśmy projekt w architekturze monorepo (frontend + backend).
- Skonfigurowaliśmy środowisko programistyczne oraz repozytorium GitHub z działającym CI.
- Utworzyliśmy i połączyliśmy bazę danych SQLite.
- Zaimplementowaliśmy system logowania i rejestracji użytkowników (formularze + sesje).
- Przygotowaliśmy wspólne pliki konfiguracyjne TypeScript, ESLint, Prettier.
- Ustaliliśmy zasady pracy zespołowej i strukturę commitów w repozytorium.

Stan tablicy Kanban na dzień 10.11.2025



Stan tablicy Kanban na dzień 14.11.2025



Commity z Githuba – stan na dzień 14.11.2025

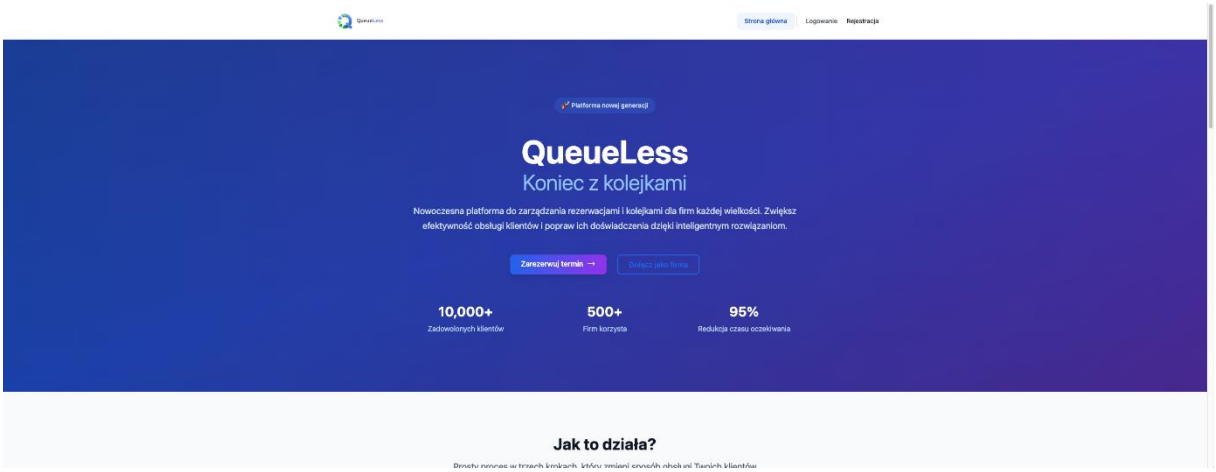

Fix missing newline at end of UnauthorizedPage.tsx
Kubaaa04 pushed 1 commit to main • d69093c...5e82d70 • yesterday
Delete packages/frontend/src/pages/UnauthorizedPage
Kubaaa04 pushed 1 commit to main • 6d47b56...d69093c • yesterday
Create UnauthorizedPage component for access control
Kubaaa04 pushed 1 commit to main • a902c4e...6d47b56 • yesterday
Update RegisterPage.tsx
Kubaaa04 pushed 1 commit to main • f35ece0...a902c4e • yesterday
Update demo account credentials on LoginPage
Kubaaa04 pushed 1 commit to main • 88f106b...f35ece0 • yesterday
Refactor mailer to use Ethereal test account
Rafaue pushed 1 commit to main • ff04ce9...88f106b • yesterday
Update TypeScript config to include db directory
Rafaue pushed 1 commit to main • f47a695...ff04ce9 • yesterday
Update package.json for backend configuration
Rafaue pushed 1 commit to main • 9abffd1...f47a695 • yesterday
Add imports for migration and seeding scripts
Rafaue pushed 1 commit to main • 4e60522...9abffd1 • yesterday


Rename path
Dan69Dan pushed 1 commit to main • 28f0105...dc76cde • 5 hours ago
Create seed.ts
Dan69Dan pushed 1 commit to main • be13edc...28f0105 • 5 hours ago
Create 001_init.sql
Dan69Dan pushed 1 commit to main • 5ca5fd0...be13edc • 5 hours ago
Update db.ts
Dan69Dan pushed 1 commit to main • 75125bd...5ca5fd0 • 5 hours ago
Create migrate.ts
Dan69Dan pushed 1 commit to main • b2c3532...75125bd • 5 hours ago
Create .gitkeep
kkkejkus pushed 1 commit to main • 88f333a...b2c3532 • 6 hours ago
Implement Worker Dashboard component
kkkejkus pushed 1 commit to main • 4e78625...88f333a • 6 hours ago
Add OwnerDashboard component for managing companies
kkkejkus pushed 1 commit to main • 860f82d...4e78625 • 6 hours ago
Add UserDashboard component for managing reservations
kkkejkus pushed 1 commit to main • f649639...860f82d • 6 hours ago
Add QueuePage component to display queue status
kkkejkus pushed 1 commit to main • 437a408...f649639 • 6 hours ago
Create CompanySettingsPage.tsx
kkkejkus pushed 1 commit to main • df9f27a...437a408 • 6 hours ago
Add ReservationPage component for booking process
kkkejkus pushed 1 commit to main • f9e4858...df9f27a • 6 hours ago
Ensure root elements have full height
kkkejkus pushed 1 commit to main • 76f9305...f9e4858 • 6 hours ago
Fix CSS syntax for body-responsive class
kkkejkus pushed 1 commit to main • c480537...76f9305 • 6 hours ago
Revise homepage text and links for clarity
kkkejkus pushed 1 commit to main • f81f5b1...c480537 • 6 hours ago
Fix formatting in ProtectedRoute component
kkkejkus pushed 1 commit to main • fc19fc3...f81f5b1 • 6 hours ago
Fix formatting in Logo.tsx component
kkkejkus pushed 1 commit to main • cb73695...fc19fc3 • 6 hours ago
Add protected routes for reservation and dashboards
kkkejkus pushed 1 commit to main • 671d571...cb73695 • 6 hours ago

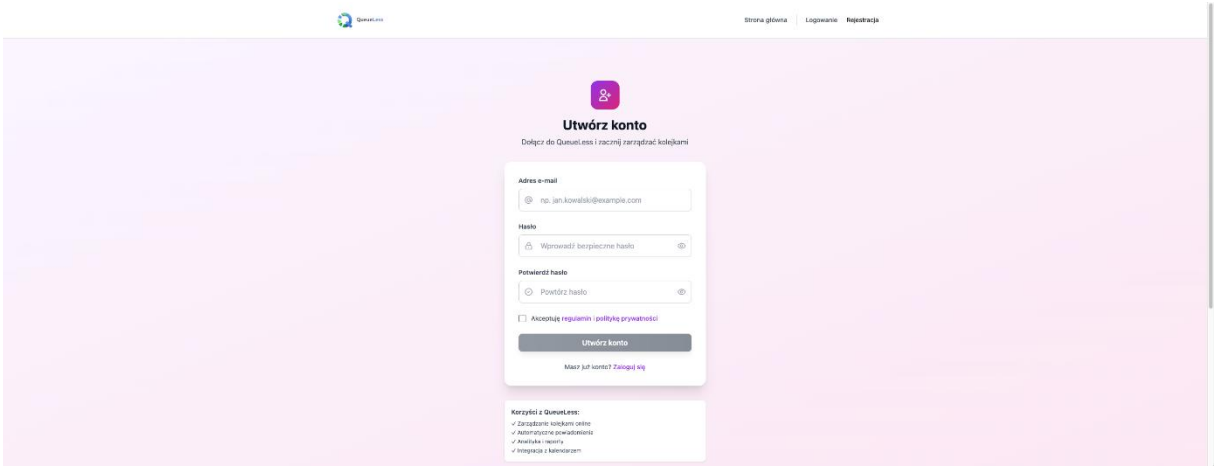

Ensure root elements have full height
kkkejkus pushed 1 commit to main • 76f9305...f9e4858 • 6 hours ago
Fix CSS syntax for body-responsive class
kkkejkus pushed 1 commit to main • c480537...76f9305 • 6 hours ago
Revise homepage text and links for clarity
kkkejkus pushed 1 commit to main • f81f5b1...c480537 • 6 hours ago
Fix formatting in ProtectedRoute component
kkkejkus pushed 1 commit to main • fc19fc3...f81f5b1 • 6 hours ago
Fix formatting in Logo.tsx component
kkkejkus pushed 1 commit to main • cb73695...fc19fc3 • 6 hours ago
Add protected routes for reservation and dashboards
kkkejkus pushed 1 commit to main • 671d571...cb73695 • 6 hours ago
Update user role handling in dashboard path
kkkejkus pushed 1 commit to main • afe7a85...671d571 • 6 hours ago
Fix formatting in main.tsx
kkkejkus pushed 1 commit to main • 7c25ab2...afe7a85 • 6 hours ago
Fix missing newline at end of index.html
kkkejkus pushed 1 commit to main • 44ad662...7c25ab2 • 6 hours ago
Refactor reservations route with Zod and direct DB access
x-PriMo pushed 1 commit to main • 9b34ee8...44ad662 • 7 hours ago
Implement rate limiting middleware
x-PriMo pushed 1 commit to main • 6cca462...9b34ee8 • 7 hours ago
Refactor authentication routes with database queries
x-PriMo pushed 1 commit to main • 31ab363...6cca462 • 7 hours ago
Enhance company routes with timezone and logo upload
x-PriMo pushed 1 commit to main • da86224...31ab363 • 7 hours ago
Add reservations routes to the backend
x-PriMo pushed 1 commit to main • a9b37a7...da86224 • 7 hours ago
Add user management and loading functionality
Kubaaa04 pushed 1 commit to main • a61ec49...a9b37a7 • yesterday
Handle error response in API calls
Kubaaa04 pushed 1 commit to main • c93d49b...a61ec49 • yesterday
Fix missing newline at end of AuthContext.tsx
Kubaaa04 pushed 1 commit to main • 22c2e1d...c93d49b • yesterday
Fix missing newline at end of AuthContext.tsx
Kubaaa04 pushed 1 commit to main • 5e82d70...22c2e1d • yesterday

Implementowane widoki aplikacji

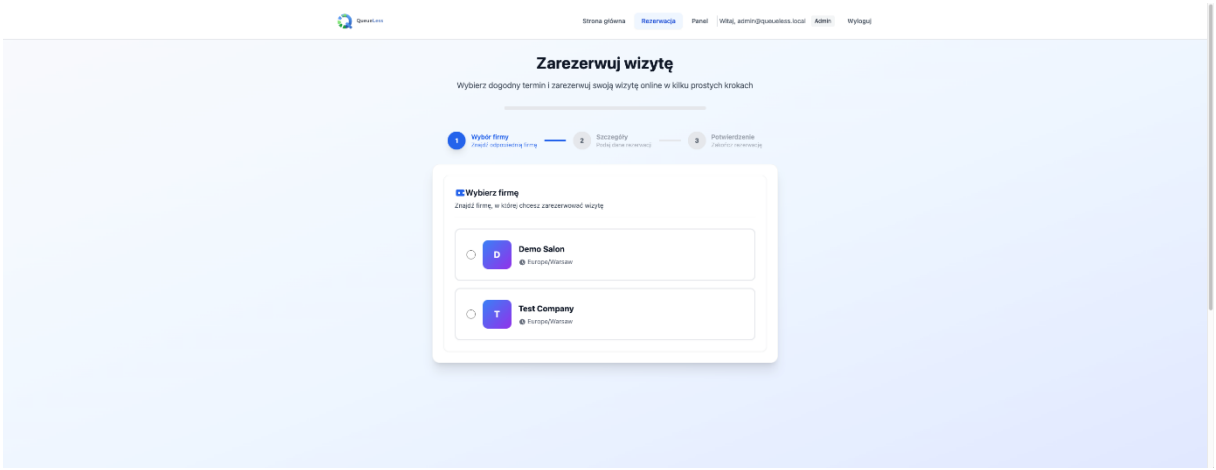
Strona głowna:



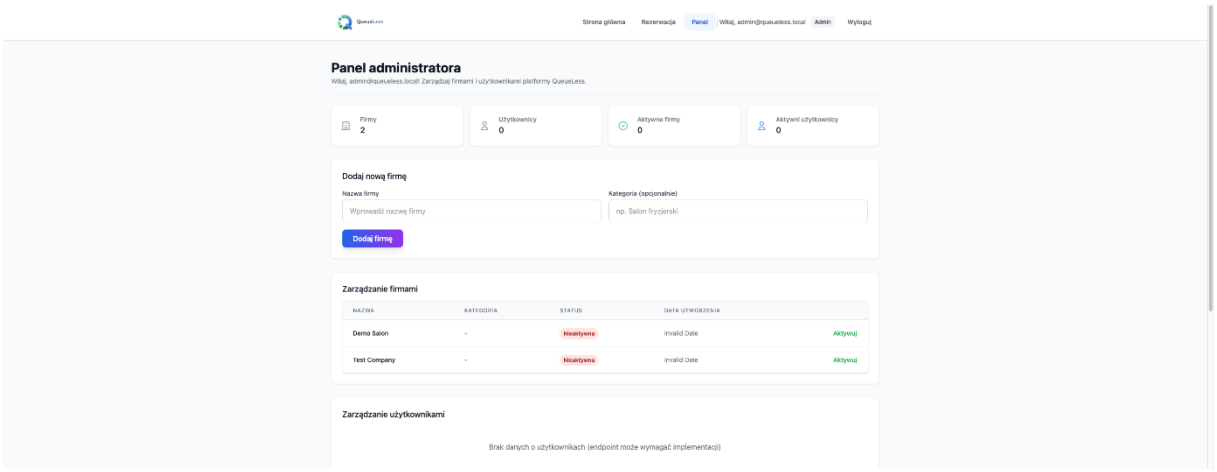
Zakładanie konta użytkownika:



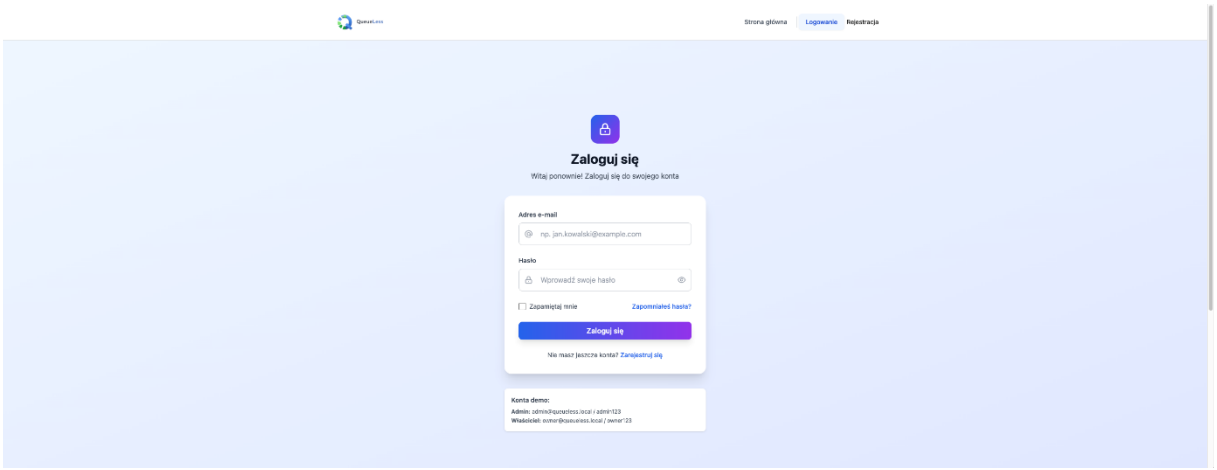
Strona rezerwacji wizyty:



Panel Administratora (Dla klienta prowadzącego firme):



Panel logowania:



Kod źródłowy – istotne fragmenty opracowane w bieżącym tygodniu

packages > backend > src > middleware > TS rateLimit.ts > ...

```ts
import { Request, Response, NextFunction } from 'express';

type Keyer = (req: Request) => string;

interface Rule {
  windowMs: number;
  max: number;
  keyer: Keyer;
}

const store = new Map<string, { count: number; resetAt: number }>();

export function rateLimit(rule: Rule) {
  return (req: Request, res: Response, next: NextFunction) => {
    const key = rule.keyer(req);
    const now = Date.now();
    const entry = store.get(key);
    if (!entry || now > entry.resetAt) {
      store.set(key, { count: 1, resetAt: now + rule.windowMs });
      return next();
    }
    if (entry.count >= rule.max) {
      return res.status(429).json({ error: 'Too many requests' });
    }
    entry.count += 1;
    next();
  };
```

Kod dostarcza funkcję getTransport() tworzącą transporter SMTP przez nodemailer .
W trybie dev automatycznie zakłada konto Ethereal i zwraca gotowy transporter (host smtp.ethereal.email , port 587 , secure: false).
W bloku catch zwracany jest „no-op” transport z metodą sendMail , który nic nie robi, dzięki czemu wywołania mailera nie wywalają aplikacji bez sieci/SMTP.
Taki fallback ułatwia lokalny rozwój i testy/CI — kod wysyłki można wołać bez warunków na środowisko.

packages > backend > src > lib > TS mailer.ts > getTransport

```ts
import nodemailer from 'nodemailer';

export async function getTransport() {
  try {
    // W dev korzystamy z konta testowego Ethereal
    const testAccount = await nodemailer.createTestAccount();
    const transporter = nodemailer.createTransport({
      host: 'smtp.ethereal.email',
      port: 587,
      secure: false,
      auth: {
        user: testAccount.user,
        pass: testAccount.pass,
      },
    });
    return transporter;
  } catch (e) {
    // Brak sieci → zwróć obiekt transportu, który nic nie robi
    return {
      sendMail: async () => ({})
    } as any;
  }
```

Kod implementuje prosty, generyczny rate-limiter: funkcja rateLimit(rule) przyjmuje okno czasowe (windowMs), limit (max) i funkcję klucza (keyer). Stan trzymany jest w Map , resetowany po wygaśnięciu okna; przy przekroczeniu zwracany jest HTTP 429.
Gotowe reguły:
loginRateLimit : 5 prób logowania na minutę per IP.
reservationRateLimit : 1 rezerwacja na 30 s per IP+email (lowercase).