

# Documentație Proiect: Pet Sitter Platform

## 1. Introducere și Viziune

**Pet Sitter App** este o platformă web modernă concepută pentru a facilita interacțiunea dintre posesorii de animale de companie și îngrijitorii. Într-o lume în care timpul proprietarilor este limitat, aplicația oferă o soluție digitală centralizată pentru servicii precum dog walking, pet sitting (îngrijire la domiciliu) și daycare.

Proiectul pune accent pe o experiență de utilizare (UX) fluidă și un design modern.

## 2. Arhitectura Tehnică (Angular 19)

Aplicația respectă standardele de coding prevăzute în metodologia Angular, utilizând cele mai noi funcționalități ale framework-ului pentru performanță maximă:

- Standalone Components: Eliminarea modulelor clasice (**NgModule**) pentru o structură mai clară și un proces de build mai rapid.
- Reactive State Management (Signals): Utilizarea Angular Signals pentru gestionarea stării aplicației, asigurând o actualizare eficientă a interfeței fără verificări inutile de schimbare (Change Detection).
- Lazy Loading: Implementarea încărcării întârziate pentru rute, ceea ce reduce dimensiunea bundle-ului inițial și îmbunătățește scorul Core Web Vitals.
- Arhitectură bazată pe Servicii: Logica de business (cum ar fi schimbarea temei sau gestionarea datelor) este extrasă în servicii injectabile pentru o mai bună testabilitate.

### 3. Funcționalități Implementate (Features)

#### Interfață și Design (UI/UX)

- Responsive Design: Interfață adaptivă pentru mobil, tabletă și desktop, utilizând un grid flexibil și un meniu tip "hamburger" pentru ecrane mici.
- Dark/Light Mode Switch: Sistem de tematică dinamic gestionat prin `DarkModeService`. Preferința utilizatorului persistă între sesiuni prin utilizarea `localStorage`.
- Internaționalizare (i18n): Suport complet pentru limbile Română și Engleză prin `@ngx-translate`. Utilizatorul poate schimba limba în timp real fără a reîncărca pagina.

#### Navigare și Routing

- Sistem de Rute Dinamice: Permite accesarea detaliilor specifice fiecărui anunț prin parametri de tip ID (ex: `/job-details/:id`).
- Rute de Fallback: Gestionarea erorilor 404 prin redirecționare automată către pagina principală pentru orice rută inexistentă.

#### Sistem de Comunicare și Contact

Pentru a facilita interacțiunea directă, am implementat două metode de comunicare:

- **Integrare EmailJS:** Formularul de contact utilizează serviciul `EmailJS`, permitând trimiterea e-mailurilor direct din interfața de Front-End fără a necesita un backend complex. Aceasta asigură primirea rapidă a mesajelor de la potențiali clienți.
- **Trimitere Directă prin Mailto:** Am integrat funcționalitatea `mailto`, care deschide automat clientul de mail implicit al utilizatorului pentru o trimitere rapidă și directă către adresa specificată, eliminând pașii intermediari.

#### Managementul Serviciilor

- Explorare și Filtrare: Pagina dedicată vizualizării tuturor sitter-ilor disponibili, cu posibilitatea de filtrare pe categorii (Walking, Training, Daycare).
- Sistem de Rezervare (Flow): Proces ghidat de rezervare, începând de la selecția serviciului până la pagina de confirmare a programării.
- Creare Anunț: Formular dedicat pentru utilizatorii care doresc să își ofere serviciile, cu validări specifice Angular.

## 4. Detalii de Implementare (Code Snippets)

Conform cerințelor de a utiliza logica reactivă, am implementat gestionarea temei folosind [Signals](#):

### TypeScript

```
export class DarkModeService {
    darkModeSignal = signal<string>(
        localStorage.getItem('theme') || 'light'
    );

    toggleDarkMode() {
        this.darkModeSignal.update(current => {
            const newTheme = current === 'light' ? 'dark' : 'light';
            localStorage.setItem('theme', newTheme);
            return newTheme;
        });
    }
}
```

Pentru a asigura o experiență rapidă, rutele sunt configurate cu Lazy Loading:

### TypeScript

```
export const ROUTES: Route[] = [
    {
        path: 'explore',
        loadComponent: () =>
            import('./features/explore/explore.component').then(m =>
                m.ExploreComponent)
    },
    {
        path: 'booking',
        loadComponent: () =>
            import('./features/booking/booking.component').then(m =>
                m.BookingComponent)
    }
];
```

**Link repository:**

<https://github.com/Alexfl14/AAFEAProject>

**Link videoclip Youtube:**

<https://youtu.be/xtejJoYOKD0>