Documentación Técnica y Funcional

Proyecto Arenas Website Frontend

Equipo de Desarrollo Arenas

12 de junio de 2025

Índice

1. Introducción

Este documento describe la arquitectura, tecnologías, estructura y funcionamiento del proyecto **Arenas Website Frontend**, una plataforma inmobiliaria moderna desarrollada con Next.js y React, que integra servicios de la API de Domus para la gestión de propiedades.

2. Tecnologías Utilizadas

- **Next.js**: Framework de React
- **React**: Librería para interfaces de usuario
- **Tailwind CSS**: Framework de utilidades CSS
- **Pannellum**: Visualización de imágenes 360°
- **Swiper**: Carruseles y sliders
- **React Icons**: Iconografía
- **Axios**: Cliente HTTP para consumo de APIs

3. Estructura del Proyecto

3.1. Organización de Directorios

- `/components/`
  - `/home/`: Componentes de la página principal
  - `/properties/`: Componentes de propiedades
  - `/filters/`: Componentes de filtrado
  - `/layout/`: Componentes estructurales
  - `/search/`: Componentes de búsqueda
- `/pages/`: Rutas y API endpoints
- `/styles/`: Configuración de estilos
- `/public/`: Recursos estáticos
- `/lib/`: Utilidades y helpers

4. Estructura de Páginas

La carpeta `pages` en Next.js define la estructura de rutas de la aplicación siguiendo el patrón de enrutamiento basado en archivos:

4.1. Archivos Principales

- `_app.js`: Componente raíz de la aplicación
  - Gestiona el layout global
  - Integra HubSpot Chat
  - Implementa recarga dinámica de estilos CSS
  - Maneja el estado global de la aplicación
- `index.js`: Página principal
  - Implementa el layout de la landing page
  - Integra componentes principales:
    - HeroSection
    - TrustedCompanies
    - PopularListings
    - PopularCities
    - FeaturedListing
    - Testimonials
    - OurAgents
    - Newsletter

4.2. API Routes (/api)

La carpeta `/api` contiene los endpoints del backend:

4.3. Rutas de Propiedades (/properties)

La carpeta `/properties` maneja las vistas de propiedades:

- `index.js`: Lista principal de propiedades
- `[id].js`: Vista detallada de una propiedad (ruta dinámica)

4.4. Ejemplo de _app.js

Listing 1: pages/_app.js

```
function MyApp({ Component, pageProps }) {
  useEffect(() => {
    const reloadCSS = () => {
      const links = document.querySelectorAll('link[rel="stylesheet"]');
      links.forEach(link => {
        const href = link.getAttribute('href');
        if (href && href.includes('colors.css')) {
          const newHref = `${href.split('?')[0]}?reload=${Date.now()}`;
          link.setAttribute('href', newHref);
        }
      });
    };
```

```
13
14      reloadCSS();
15      const interval = setInterval(reloadCSS, 2000);
16      return () => clearInterval(interval);
17    }, []);
18
19    return (
20      <>
21        <Head>
22          <script type="text/javascript" id="hs-script-loader"
23                  async defer src="//js.hs-scripts.com/8765689.js">
24          </script>
25        </Head>
26        <div className="relative">
27          <Layout>
28            <Component {...pageProps} />
29          </Layout>
30          <div id="hs-chat-widget" className="fixed right-0 bottom-0 z-50"
               >
31          </div>
32        </div>
33      </>
34    );
35  }
```

4.5. Ejemplo de Página Principal

Listing 2: pages/index.js

```
1   export default function Home() {
2     return (
3       <>
4         <Head>
5           <title>Arenas Real Estate - Find Your Dream Property</title>
6           <meta name="description"
7                 content="Find and explore the best properties" />
8         </Head>
9         <main className="bg-[#f8f9fa]">
10          <HeroSection />
11          <TrustedCompanies />
12          <PopularListings />
13          <PopularCities />
14          <FeaturedListing />
15          <Testimonials />
16          <OurAgents />
17          <Newsletter />
18        </main>
19        <Footer />
20      </>
21    );
22  }
```

5. Arquitectura de Componentes

La carpeta `components` está organizada en módulos funcionales que facilitan la mantenibilidad y reutilización del código:

5.1. Componentes de Página Principal (/home)

Componentes específicos para la landing page:

- **Sección Hero**:
  - `HeroSection.js`: Banner principal
  - `HeroSearch.js`: Buscador integrado
- **Secciones Destacadas**:
  - `PopularCities.js`: Ciudades populares
  - `FeaturedProjects.js`: Proyectos destacados
  - `TrustedCompanies.js`: Empresas asociadas
- **Servicios y Accesos**:
  - `QuickServices.js`: Servicios rápidos
  - `QuickAccessCards.js`: Tarjetas de acceso rápido
  - `InvestmentSection.js`: Sección de inversiones
- **Contenido Social**:
  - `Testimonials.js`: Testimonios de clientes
  - `OurAgents.js`: Equipo de agentes
  - `FeaturedAdvisors.js`: Asesores destacados

5.2. Componentes de Propiedades (/properties)

Sistema completo de visualización y gestión de propiedades:

5.3. Componentes de Layout (/layout)

Estructura base de la aplicación:

- `Layout.js`: Contenedor principal
- `Header.js`: Navegación principal
- `Footer.js`: Pie de página
- `MobileMenu.js`: Menú móvil responsive

5.4. Sistema de Filtros (/filters)

Listing 3: FilterSidebar.js

```
export default function FilterSidebar() {
  const [filters, setFilters] = useState({
    propertyType: [],
    bizType: '',
    bedrooms: [],
    bathrooms: [],
    minPrice: '',
    maxPrice: '',
    stratum: ''
  });

}
```

5.5. Sistema de Búsqueda (/search)

Listing 4: SearchBar.js

```
export default function SearchBar() {
  const [query, setQuery] = useState('');
  const [results, setResults] = useState([]);

}
```

5.6. Patrones de Diseño

Los componentes siguen estos patrones principales:

- Componentes funcionales con hooks
- Prop drilling minimizado
- Composición sobre herencia
- Lazy loading para optimización
- Estados locales para UI
- Manejo de efectos secundarios con useEffect

6. Componentes Principales

6.1. Header Component

Listing 5: components/layout/Header.js

```
export default function Header() {
  const [mobileMenuOpen, setMobileMenuOpen] = useState(false);
  const [searchQuery, setSearchQuery] = useState('');

```

```
  useEffect(() => {
    if (!searchQuery.trim()) return;
    const timer = setTimeout(() => {
      setDebouncedSearchQuery(searchQuery);
    }, 500);
    return () => clearTimeout(timer);
  }, [searchQuery]);

  return (
    <header className="bg-white border-b border-gray-100">
      {/* Contenido del header */}
    </header>
  );
}
```

6.2. FilterSidebar Component

Listing 6: components/filters/FilterSidebar.js

```
export default function FilterSidebar() {
  const [filters, setFilters] = useState({
    propertyType: [],
    bizType: '',
    bedrooms: [],
    bathrooms: [],
    minPrice: '',
    maxPrice: '',
    stratum: ''
  });

  useEffect(() => {
    const { type, biz, minPrice, maxPrice } = router.query;
  }, [router.query]);

  return (
    <aside className="bg-white p-4">
      {/* Contenido de filtros */}
    </aside>
  );
}
```

6.3. Property360Viewer Component

Listing 7: components/properties/Property360Viewer.js

```
const Property360Viewer = () => {
  const [isPannellumLoaded, setIsPannellumLoaded] = useState(false);

  useEffect(() => {
    if (window.pannellum) {
      setIsPannellumLoaded(true);
    }
  }, []);

  return (
```

```
    <div className="relative w-full">
    </div>
  );
};
```

7. Estilos y Temas

7.1. Configuración de Tailwind

Listing 8: tailwind.config.js

```
module.exports = {
  content: [
    "./pages/**/*.{js,ts,jsx,tsx}",
    "./components/**/*.{js,ts,jsx,tsx}",
  ],
  theme: {
    extend: {
      colors: {
        primary: "#702571",
        secondary: "#c245c5",
        accent: "#f59e0b",
        dark: "#000000",
        light: "#f3f4f6"
      }
    }
  }
}
```

7.2. Variables CSS Personalizadas

Listing 9: styles/colors.css

```
:root {
  --color-primary: #702571;
  --color-secondary: #c245c5;
  --color-accent: #f59e0b;
  --color-dark: #000000;
  --color-light: #f3f4f6;
}
```

8. Integración con API

8.1. Configuración de Next.js

Listing 10: next.config.js

```
module.exports = {
  async rewrites() {
    return [
      {
        source: '/api/:path*',
```

```
        destination: 'https://api.domus.la/3.0/:path*'
      }
    ];
  }
};
```

8.2. Ejemplo de Consumo de API

Listing 11: components/home/PopularListings.js

```
const fetchPropertiesByCity = async () => {
  try {
    const response = await fetch('/api/properties?city=8001&perpage=6');
    const data = await response.json();
    return data.data.map(property => ({
      id: property.codpro,
      title: property.address_alt || property.address,
      price: property.price_format,
    }));
  } catch (error) {
    console.error('Error:', error);
    return [];
  }
};
```

9. Despliegue y Ejecución

9.1. Requisitos Previos

- Node.js 22.x o superior
- NPM 10.x o superior
- Variables de entorno configuradas

9.2. Pasos de Instalación

1. Clonar el repositorio
2. Ejecutar `npm install`
3. Configurar `.env.local`
4. Ejecutar `npm run dev`

9.3. Variables de Entorno Requeridas

- `NEXT_PUBLIC_API_URL`
- `NEXT_PUBLIC_DOMUS_API_TOKEN`