Documentació de l'estructura del projecte APP d'Abasta

APP d'Abasta – Frontend

Aquest document descriu l'estructura interna del projecte Abasta (App Web), desenvolupat amb React, TypeScript i Bootstrap. Està pensat per a facilitar la navegació, la comprensió i el manteniment per part dels desenvolupadors i col·laboradors.

Arquitectura general

El projecte segueix una arquitectura modular i escalable, dividida en carpetes per responsabilitat i àmbit funcional. Es manté una separació clara entre la lògica, la presentació i la comunicació amb serveis externs.

Estructura de Directoris

A continuació, es detalla l'estructura del projecte amb una descripció del propòsit de cada carpeta.

Data de documentació: Octubre, 2025

Abasta

```
/src
│
├── /assets        → Recursos estàtics (imatges, icones, estils)
│   ├── /styles    → Fitxers CSS/SCSS globals o temes personalitzats
│   ├── /icons     → Fitxers SVG, PNG o altres icones utilitzades
│   └── /images    → Imatges per banners, placeholders, logos...
│
├── /components    → Components React reutilitzables
│   ├── /common    → Botons, inputs, modals... sense dependència de context
│   ├── /auth      → Formularis de registre, login, recuperació...
│   └── /layout    → Navbar, Sidebar, Footer i altres estructures globals
│
├── /config        → Configuracions globals (API, rutes, constants)
│
├── /docs
│   └── /frontend  → Documentació funcional i tècnica del frontend
│
├── /pages         → Components de pàgines (Home, Perfil, Registre...)
│
├── /services      → Interfícies per a comunicació amb APIs
│                  → Ex: authService, userService...
│
├── /hooks         → React hooks personalitzats reutilitzables
│                  → Ex: useAuth, useFetch, useForm...
│
├── /context       → React Contexts per a estats globals compartits
│                  → Ex: AuthContext, ThemeContext...
│
├── /utils         → Funcions utilitàries (helpers) reutilitzables
│                  → Ex: formatDate, validateEmail...
│
├── /types         → Tipus i interfícies TypeScript globals
│                  → Ex: User, Company, ApiResponse...
│
└── /routes        → Definició de rutes i protecció de navegació
                   → Ex: AppRoutes, PrivateRoute, RouteConfig...
```

Data de documentació: Octubre, 2025

Abasta

Descripció de les carpetes

/assets : Recursos estàtics de l'aplicació.

- /styles: Fitxers CSS/SCSS globals, variables de tema, estils personalitzats.
- /icons: Fitxers SVG o PNG d'icones.
- /images: Imatges generals (logos, fons, avatars, etc.).

/components : Components React reutilitzables, organitzats per àmbits d'ús.

- /common: Components genèrics reutilitzables a tot el projecte.Sense dependència de context o estat global. Exemples: Button.tsx, Input.tsx, Modal.tsx.
- /auth: Components específics per a l'autenticació d'usuaris. Inclou validacions i interacció amb l'API d'autenticació.
  Exemples: Login.tsx, Register.tsx, ForgotPassword.tsx.
- /layout: Components d'estructura visual global. S'utilitzen a múltiples pàgines com a base del layout. Exemples: Navbar.tsx, Sidebar.tsx, Footer.tsx.

/config : Configuracions globals.

- Constants de l'aplicació (URLs d'API, rutes permeses, temes...).
- Configuració centralitzada de les funcions fetch, localStorage, etc.

/context : React Contexts per gestionar estats globals compartits.

- AuthContext.tsx: estat d'autenticació.
- Altres contexts segons creixi el projecte (idioma, preferències...).

/docs/frontend : Documentació funcional i tècnica de l'aplicació frontend.
Aquesta carpeta serveix de referència interna per a nous desenvolupadors.

- **api.md** → Descripció de l'estructura i ús dels serveis d'API (endpoints, mètodes HTTP, autenticació i gestió d'errors).
- **structure.md** → Explicació detallada de l'estructura del projecte, organització de carpetes i bones pràctiques de desenvolupament.
- **style-guide.md** → Guia d'estil per a mantenir coherència en el codi (nomenclatura, convencions TypeScript, estil de components i formatat general).

Data de documentació: Octubre, 2025

Abasta

/hooks : React Hooks personalitzats. Encapsulen lògica reutilitzable fora dels components de presentació.

- useAuth.ts: estat i accions relacionades amb l'usuari autenticat.
- useForm.ts: gestió de formularis.
- useFetch.ts: peticions HTTP amb gestió d'estat (loading, error, data).

/pages : Components que representen pàgines completes (vistes).Composen components, connecten contextos i serveis, i controlen la navegació.

- Home.tsx: Pàgina principal.
- Login.tsx, Register.tsx: Vistes públiques.
- Dashboard.tsx, Profile.tsx: Vistes privades per a usuaris autenticats.

/routes : Definició de rutes amb React Router.

- AppRoutes.tsx: Arbre de rutes principal.
- PrivateRoute.tsx: Ruta protegida (redirigeix si no hi ha autenticació).
- Pot incloure Lazy Loading i agrupació per rols si s'escala el projecte.

/services : Capes d'accés a dades i API. Centralitzen la lògica de comunicació amb el backend.

- api.ts: Funcions auxiliars per a fer peticions HTTP amb fetch (GET, POST, PUT, DELETE). Gestiona headers, tokens i errors globals.
- authService.ts: Registre, login, logout, validació de token.
- userService.ts: CRUD de dades de l'usuari.

/types : Tipus i interfícies globals per a TypeScript. Milloren la seguretat del tipus i ajuden amb l'autocompletat.

- User.ts, Company.ts, ApiResponse.ts...
- Enllaçats amb les respostes de l'API i estructures de dades internes.

/utils : Funcions utilitàries reutilitzables. Sempre han de ser funcions pures i independents de la UI.

- formatDate.ts: Conversió i format de dates.
- validateEmail.ts: Validació de correus electrònics.

Data de documentació: Octubre, 2025

Funcionament General

Flux de Navegació de l'Usuari

1. **Inici:** L'usuari accedeix a la pàgina principal (/) amb informació general.
2. **Registre:** Des del login podrà seleccionar si es vol registrar, o en la pàgina principal hi haurà un enllaç.
3. **Login:** Des de la barra de navegació, pot accedir a la pàgina de login (/login) per iniciar sessió o en cas necessari registrar-se.
4. **Autenticació:** Les crides a l'API d'autenticació es fan a través de authService.ts. Si són exitoses, es desa un token (normalment al localStorage) i es redirigeix a una pàgina protegida.
5. **Navegació protegida:** Algunes rutes estan protegides amb un PrivateRoute que comprova si hi ha sessió iniciada via AuthContext.

Comunicació amb l'API

- Totes les peticions es gestionen des de la carpeta /services.
- Es fa servir la funció nativa fetch com a client HTTP, amb una utilitat centralitzada (api.ts) per gestionar headers, tokens i errors globals.
- Exemple de serveis:
    - authService.ts: login, registre, logout
    - userService.ts: informació d'usuari, actualització de perfil

Components i Layout

- Els components **/common** són genèrics i reutilitzables: Button.tsx, Input.tsx, Modal.tsx, etc.
- Els components de **layout** com Navbar o Footer es troben a /components/layout.

Data de documentació: Octubre, 2025

Autenticació i Rutes Privades

- S'utilitza un context (AuthContext) per gestionar l'estat d'autenticació.
- Les rutes privades es protegeixen mitjançant un component PrivateRoute, que redirigeix si no hi ha sessió iniciada.
- Els tokens s'emmagatzemen al localStorage i s'afegeixen automàticament als headers de les peticions fetch a través de la capa centralitzada api.ts.

Configuració

- La carpeta /config centralitza les configuracions de:
    - API (URLs, timeouts)
    - Rutes predefinides (routesConfig.ts)
    - Idioma, tema o variables d'entorn

Responsive Design

- S'utilitza Bootstrap Grid i utilitats per garantir una interfície adaptable a dispositius mòbils, tablets i escriptoris.
- Els components s'han dissenyat tenint en compte les bones pràctiques de disseny responsive.

Bones Pràctiques Aplicades

- Components petits i amb responsabilitats clares
- Ús estricte de TypeScript amb interfícies i tipus definits
- Separació clara entre dades, UI i lògica de negoci
- Reutilització màxima de components i hooks
- Estructures previsibles i mantenibles

Data de documentació: Octubre, 2025