Project Configuration for Intelligent Agent Selection

CRITICAL: Agent Selection Policy

This project uses an automated AI assistant configuration with the `portaal-fe-specialist` agent as the primary reference for all Portaal.be development tasks.

AI Assistant Configuration

The project includes a `.claude/` directory with specialized agent configurations:

```
.claude/
├── agents/
│   ├── portaal-fe-specialist.md    # Primary agent for Portaal.be
│   └── selector-agent-updated.md   # Modified selector with Portaal
priority
├── project-config.md               # Project-specific rules
├── init-prompt.md                  # Initialization context
└── README.md                       # Configuration documentation
```

Workflow Requirements

1. **For ANY Portaal.be task**:
    - Claude automatically uses `portaal-fe-specialist` as the primary agent
    - The specialist agent guides to other agents when needed
    - All solutions follow established Portaal.be patterns
2. **Specialized Tasks**:
    - Frontend components → `portaal-fe-specialist` → `frontend-developer`
    - API design → `portaal-fe-specialist` → `backend-architect`
    - Database/SQL → `portaal-fe-specialist` → `portaal-dashboard-pro`
    - Testing → `portaal-fe-specialist` → `test-automator`
    - Deployment → `portaal-fe-specialist` → `deployment-engineer`
3. **Priority Overrides**:
    - Production issues → `incident-responder` (direct routing)
    - Security vulnerabilities → `security-auditor`
    - Active bugs/errors → `debugger`

Available Specialized Agents

The `portaal-fe-specialist` coordinates with 45+ specialized agents including:

- Language experts (Python, JavaScript, Go, Rust, C/C++)

PROF

- Domain experts (Frontend, Backend, ML, DevOps, Cloud)
- Task specialists (Testing, Security, Performance, Database)
- Business tools (Analytics, Sales, Marketing, Support)

Implementation Pattern

```
User: "Create a new component for Sales microservice"
Claude:
1. Automatically uses portaal-fe-specialist (no manual selection needed)
2. Identifies Sales microservice context (port 3008)
3. Follows Portaal.be patterns (TypeScript, Common components,
permissions)
4. May engage frontend-developer for specific React patterns
```

Remember

- The `portaal-fe-specialist` ensures consistency with Module Federation architecture
- Using the configured agent system improves quality and maintains patterns
- Manual agent selection is rarely needed - the system auto-routes
- When in doubt, mention "Portaal.be" or "Module Federation" to ensure correct routing

Project Rules

1. **Agent selection is automatic** based on context detection
2. **All code follows Module Federation patterns** from portaal-fe-specialist
3. **Multi-agent collaboration** maintains Portaal.be context
4. **Documentation is centralized** in the .claude/ directory

This configuration ensures that every request gets routed through the Portaal.be specialist first, maintaining architectural consistency.

PROF

Architettura della Soluzione Portaal Frontend

AI Assistant Integration

This project is configured with an intelligent AI assistant that understands the complete Module Federation architecture. The assistant:

- **Automatically activates** for any Portaal.be related task
- **Maintains context** across all 15+ microservices
- **Guides development** according to established patterns
- **Routes to specialists** for specific technical tasks

To use: Simply ask questions or request changes. The AI will automatically use the appropriate knowledge base.

Struttura Generale

La soluzione è basata su **Module Federation** di Webpack e utilizza una architettura a microservizi frontend (MFE - Micro Frontend):

Microservizi Principali

1. **Core** (porta 3000) - Container principale che orchestra tutti gli altri microservizi
2. **Common** (porta 3003) - Componenti e servizi condivisi
3. **Auth** (porta 3006) - Autenticazione e gestione utenti
4. **Dashboard** (porta 3020) - Dashboard principale
5. **Dashboard Editor** (porta 3022) - Editor per creare e modificare dashboard
6. **Report Editor** (porta 3021) - Editor per report
7. **Lookups** (porta 3005) - Gestione dati di lookup
8. **Sales** (porta 3008) - Modulo vendite
9. **HR** (porta 3009) - Risorse umane
10. **Recruiting** (porta 3011) - Reclutamento
11. **Stock** (porta 3012) - Magazzino/inventario
12. **Notifications** (porta 3013) - Sistema notifiche
13. **Reports** (porta 3015) - Generazione report
14. **Chatbot** (porta 3018) - Assistente chat
15. **Personal Area** - Area personale utente

Microservizi Indispensabili

- **Core**: orchestratore principale
- **Common**: componenti condivisi
- **Auth**: autenticazione necessaria per accedere all'applicazione

Come Si Avvia la Soluzione

1. Gestione con PM2

PROF

La soluzione utilizza **PM2** come process manager per gestire tutti i microservizi:

Per avviare seguire la procedura##

1. Stop di tutti i servizi

pm2 stop portaal-fe

2. Cancellare tutti i servizi

pm2 delete portaal-fe

3 Start di tutti i servizi

pm2 start ecosystem.config.js

2. File di Configurazione Principale

`ecosystem.config.js`: Definisce tutti i microservizi e le loro configurazioni PM2:

- Ogni microservizio ha una configurazione separata
- Supporta modalità `development` e `live` (production)
- Utilizza webpack-dev-server per ogni microservizio
- Namespace comune: `portaal-fe`

Configurazione Core per Abilitare/Disabilitare Microservizi

1. Variabile d'Ambiente ENABLED_MFES

Nel file `portaal-fe-core/.env.development` si può configurare quali microservizi abilitare:

```
# Per abilitare TUTTI i microservizi (lasciare vuoto o commentato)
ENABLED_MFES=

# Per abilitare solo specifici microservizi (separati da virgola)
ENABLED_MFES=dashboard,lookups,sales,hr,recruiting,stock,notifications,r
eports,chatbot,personalarea

# Esempio: solo sales e dashboard
ENABLED_MFES=sales,dashboard
```

2. Come Funziona il Sistema

1. **webpack.config.js del Core**:
    - Legge la variabile `ENABLED_MFES`
    - Se vuota: abilita tutti i microservizi
    - Se valorizzata: abilita solo quelli specificati + common e auth (sempre inclusi)
    - Filtra i remotes di Module Federation in base alla configurazione
2. **mfeConfig.ts**:
    - Definisce gli import dei microservizi
    - Gestisce il caricamento dinamico delle route
    - Contiene la mappatura dei nomi e delle porte

PROF

3. Comportamento del Sistema

- **PM2 avvia sempre tutti i servizi** definiti in `ecosystem.config.js`
- **Solo i microservizi abilitati** vengono effettivamente caricati e utilizzati dal Core
- I servizi non abilitati rimangono in esecuzione ma non sono accessibili dall'applicazione
- Se un microservizio abilitato non è in esecuzione:

- Appare un messaggio di errore nell'interfaccia
- Il menu del microservizio non viene visualizzato
- Le route mostrano un componente di errore

4. Vantaggi della Configurazione Selettiva

- **Sviluppo più veloce**: caricamento solo dei moduli necessari
- **Testing isolato**: possibilità di testare singoli microservizi
- **Risparmio risorse**: meno moduli caricati nel browser
- **Debug semplificato**: meno complessità e log da analizzare

Note Importanti

1. **Riavvio necessario**: Dopo aver modificato `ENABLED_MFES`, è necessario riavviare il Core
2. **Gestione singoli servizi**: È possibile gestire singoli servizi con PM2:

```
pm2 stop dashboard
pm2 restart core
pm2 logs auth
```

3. **Build di produzione**: In produzione i remotes utilizzano path relativi invece di URL localhost

Debug dell'Applicazione

AI-Assisted Debugging

Il sistema di AI assistant include capacità di debug avanzate:

1. **Analisi automatica**: L'assistant può analizzare errori e suggerire soluzioni
2. **Pattern recognition**: Identifica problemi comuni in Module Federation
3. **Guided troubleshooting**: Fornisce step-by-step per risolvere issues

PROF

Strumenti MCP Playwright per Debug

Per il debug dell'applicazione, l'AI assistant può utilizzare gli strumenti **MCP Playwright** che permettono di:

Navigazione e Interazione

- `mcp__playwright__browser_navigate` - Navigare verso URL specifici
- `mcp__playwright__browser_click` - Cliccare su elementi
- `mcp__playwright__browser_type` - Digitare testo nei campi
- `mcp__playwright__browser_snapshot` - Catturare snapshot accessibili della pagina

Analisi e Debug

- `mcp__playwright__browser_console_messages` - Leggere i messaggi della console
- `mcp__playwright__browser_network_requests` - Analizzare le richieste di rete

- `mcp__playwright__browser_evaluate` - Eseguire JavaScript nel contesto della pagina
- `mcp__playwright__browser_take_screenshot` - Catturare screenshot per analisi visiva

Browser Tools MCP

- `mcp__browser-tools__getConsoleErrors` - Ottenere errori della console
- `mcp__browser-tools__getNetworkErrors` - Analizzare errori di rete
- `mcp__browser-tools__runDebuggerMode` - Avviare modalità debug completa
- `mcp__browser-tools__runAuditMode` - Eseguire audit di performance, SEO e accessibilità

Workflow di Debug Consigliato

1. **Avviare l'applicazione** con `yarn start:dev`
2. **Descrivere il problema** all'AI assistant
3. L'assistant userà automaticamente gli strumenti appropriati
4. **Seguire le raccomandazioni** fornite dall'assistant
5. **Verificare le correzioni** con l'aiuto dell'assistant

Development con AI Assistant

Quick Start

1. **Nuovo componente**: "Crea un componente ClienteCard per Sales"
    - L'AI userà automaticamente portaal-fe-specialist
    - Seguirà i pattern esistenti
    - Fornirà codice TypeScript completo
2. **Debug**: "Il menu HR non si vede"
    - Analisi automatica del problema
    - Controllo configurazioni e permessi
    - Soluzione step-by-step
3. **Nuovo microservizio**: "Aggiungi un microservizio Documents"
    - Guida completa alla creazione
    - Tutti i file necessari
    - Integrazione con l'esistente

Best Practices con AI

1. **Sii specifico**: Menziona il microservizio target
2. **Usa il contesto**: Riferisci porte o nomi servizi
3. **Chiedi verifiche**: L'AI può controllare pattern e standard
4. **Iterazione veloce**: Chiedi modifiche incrementali

Analisi Completa dell'Architettura

Funzioni dei Microservizi

[Il resto del contenuto rimane identico all'originale, includendo tutte le sezioni su:]

- Funzioni dettagliate di ogni microservizio
- Dipendenze tra Microservizi
- Sistema di Stili e Temi
- Creazione del Menu Principale
- Routing e Navigazione
- Autenticazione e Autorizzazione
- Servizi Condivisi e API
- Best Practices e Pattern Utilizzati
- Suggerimenti per lo Sviluppo

AI Configuration Files

`.claude/agents/portaal-fe-specialist.md`

Contiene la knowledge base completa del progetto, inclusi:

- Architettura Module Federation
- Pattern di sviluppo
- Configurazioni di ogni microservizio
- Best practices e convenzioni

`.claude/project-config.md`

Definisce:

- Agent predefinito per il progetto
- Regole di auto-routing
- Struttura del progetto
- Standard di codice

`.claude/init-prompt.md`

PROF

Fornisce il contesto iniziale per ogni nuova conversazione con l'AI assistant.

Manutenzione della Configurazione AI

1. **Aggiornamenti**: Modifica i file in `.claude/` quando l'architettura cambia
2. **Nuovi pattern**: Aggiungi best practices in `portaal-fe-specialist.md`
3. **Feedback**: L'AI migliora con feedback su soluzioni non ottimali
4. **Versionamento**: I file di configurazione sono parte del repository

---

Questo documento è mantenuto in sync con la configurazione AI del progetto. Per modifiche architetturali, aggiorna sia questo file che `.claude/agents/portaal-fe-specialist.md`.