Erweiterte Entwicklungsrichtlinien für AGENT_LAND_SAARLAND

Diese Richtlinien erweitern die bestehenden .clauderules für das AGENT_LAND_SAARLAND-Projekt und integrieren die Anforderungen aus der technischen Roadmap.

1. Technologische Grundsätze

```toml
[project.technologies]
ai_frameworks = ["langchain", "llamaindex", "langGraph"]
backend_frameworks = ["nextjs", "fastapi"]
database = "postgresql"
vector_extension = "pgvector"
containerization = ["docker", "kubernetes"]
```

2. Architekturprinzipien

```toml
[architecture]
pattern = "multiagent"
style = "microservices"
approach = "api_first"
communication = "event_driven"
principles = ["privacy_by_design", "ethics_by_design"]

[architecture.layers]
orchestration = "NavigatorAgent"
specialized_agents = true
tool_integration = true
knowledge_base = "VectorIndex"
```

3. Agentenstruktur

toml

```toml
[agents.structure]
central_agents = [
  "NavigatorAgent",
  "ResearchAgent",
  "PlannerAgent",
  "AdminAgent",
  "MonitoringAgent"
]

specialized_agents = [
  "TourismAgent",
  "BusinessAgent",
  "EducationAgent",
  "CultureAgent"
]

[agents.coordination]
pattern = "CrewAI"
transparency = "chain_of_thought"
feedback_loop = true
```

4. Regionale Integration

toml

```toml
[regional_integration]
data_integration = [
  "administration_connectors",
  "tourism_connectors",
  "culture_connectors"
]

language_support = ["de", "fr", "ltz"]
dialect_support = "saarländisch"

technical_integration = [
  "egovernment_services",
  "transport_services",
  "business_services"
]
```

5. Coderichtlinien

toml

```toml
[code.typescript]
type_safety = "strict"
naming_convention = "camelCase"
component_naming = "PascalCase"
file_naming = "kebab-case"
interface_prefix = "I"
type_prefix = "T"
enum_suffix = "Enum"

[code.react]
components_path = "components"
hooks_path = "hooks"
context_path = "context"
utilities_path = "utils"
hooks_prefix = "use"
component_extension = ".tsx"
```

6. Entwicklungsrichtlinien in Textform

6.1 Spezifische TypeScript-Richtlinien

- Verwende immer explizite Typen und vermeide `any` oder implizite Typisierung
- Nutze Interfaces für Datenstrukturen und Komponenten-Props
- Implementiere Enums für fest definierte Wertebereiche
- Verwende TypeGuards für komplexe Typprüfungen
- Nutze die neuesten TypeScript-Features (optional chaining, nullish coalescing)
- Exportiere alle Typen aus einer zentralen `types.ts`-Datei pro Modul

6.2 React und Next.js Richtlinien

- Verwende funktionale Komponenten mit Hooks statt Klassenkomponenten
- Implementiere Custom Hooks für wiederverwendbare Logik
- Nutze Context API für globalen Zustand
- Verwende React.memo() für Performance-Optimierung
- Implementiere Lazy Loading für große Komponenten
- Nutze SWR oder React Query für API-Calls

6.3 Agentenentwicklung

- Jeder Agent muss eine klar definierte Rolle haben
- Implementiere einheitliche Schnittstellen für Agentenkommunikation

- Dokumentiere Agentenverhalten mit Beispielen
- Implementiere Logging für Agentenaktivitäten
- Stelle sicher, dass jeder Agent einen Fallback-Mechanismus hat
- Entwickle Testszenarien für Agenten-Interaktionen

6.4 Regionale Integration

- Stelle sicher, dass alle benutzerorientierten Texte auf Deutsch verfügbar sind
- Implementiere einen Dialekt-Modus mit saarländischen Ausdrücken
- Integriere regionale Datenquellen mit entsprechenden Connectoren
- Dokumentiere regionale Besonderheiten, die in der KI berücksichtigt werden müssen
- Nutze lokale Beispiele in Dokumentation und Tests

6.5 Sicherheit und Datenschutz

- Implementiere DSGVO-konforme Datenspeicherung
- Nutze Verschlüsselung für sensible Daten
- Führe regelmäßige Sicherheitsaudits durch
- Implementiere Nutzerzustimmung für Datenverarbeitung
- Dokumentiere Datenflüsse transparent
- Stelle sicher, dass KI-Entscheidungen nachvollziehbar sind

6.6 CI/CD und Deployment

- Automatisiere Tests vor jedem Deployment
- Implementiere Staging-Umgebungen für Tests
- Nutze Infrastructure as Code für Cloud-Ressourcen
- Implementiere Blue-Green Deployment für reibungslose Updates
- Stelle Monitoring für KI-Dienste bereit
- Entwickle Rollback-Strategien für fehlgeschlagene Deployments

7. KI-Modellintegration

toml

```toml
[ai_models]
primary = "Claude"
fallback = "local_open_source"
speed_optimized = "Groq"
specialized = "domain_specific_models"

[ai_integration]
abstraction_layer = true
message_queue = true
circuit_breaker = true
rate_limiting = true
caching_strategy = true
```

8. Dokumentationsstandards

toml

```toml
[documentation]
api_standard = "OpenAPI"
code_comments = true
architecture_diagrams = true
decision_records = true
user_guides = true
developer_guides = true
regional_context = true
```

9. Qualitätsstandards

toml

```toml
[quality]
code_coverage = 80
performance_metrics = true
accessibility = "WCAG_AA"
internationalization = true
compatibility = ["desktop", "mobile", "tablet"]
testing_levels = ["unit", "integration", "e2e"]
```

10. Projektstruktururweiterung

toml

```toml
[project.structure]
specialized_folders = [
  "regional-data/",
  "agent-definitions/",
  "ai-models/",
  "integration-connectors/"
]

enforce_conventional_commits = true
pull_request_template = true
issue_templates = true
```