KodeMed

Architettura del sistema

Per CTO, amministrazione IT & DevOps

Version 2026.5.15.59502 | KodeMed AG

Panoramica del sistema

Componenti della piattaforma KodeMed

KodeMed.Server

Backend Java (REST + WebSocket)

KodeMed.DataServer

API dati di classificazione

KodeMed.GrouperServer

API di raggruppamento DRG (SwissDRG, TARPSY, ST Reha)

KodeMed.CodingUI

Frontend web React/Vite

KodeMed.Interface

DLL COM (.NET 9.0)

KodeMed.CodingClient

App system tray – EXE portatile
WebSocket, webhook, selettore lingua

```
Flusso: SIO/Terzi → Server (REST/WS) → CodingClient (WS) → DLL → CodingUI (WebView2)

Flusso: SIO (diretto) → DLL (COM) → Server → CodingUI (WebView2)
```

Integrazione SIO

Integrazione del sistema informativo ospedaliero via DLL COM

API DLL (IKodeMed)

Method	Description
SendConfig(xml)	Load configuration
DoCodingWithFormat(data, format)	Full UI flow (auth + browser)
DoCoding(data, format)	Headless processing
GetResults()	Retrieve coding results
GetResultData()	Get clean SpiGes XML
DisposeClient()	Release resources

Esempio di integrazione (C#)

```
// 1. Create instance
var coding = new Coding();

// 2. Configure
coding.SendConfig(configXml);

// 3. Process with UI
coding.DoCodingWithFormat(
  spigesData, FormatType.SpiGes);

// 4. Check action
if (coding.LastCodingAction
    == CodingAction.Applied)
{
  var results = coding.GetResults();
  var xml = coding.GetResultData();
}

// 5. Cleanup
coding.DisposeClient();
```

Installazione del client

Tre metodi di distribuzione

Installatore MSI (consigliato)

- Pacchetto KodeMed.msi
- Per utente (senza admin) o per macchina (Citrix)
- Installazione silenziosa via GPO/SCCM/Intune
- Registrazione COM + configurazione inclusa
- Runtime WebView2 richiesto

Manuale / Xcopy

- Copiare DLL + dipendenze
- Registrazione COM manuale
- Variabili d’ambiente manuali
- Per utenti esperti

CodingClient

- Installato via MSI
- System tray + WebSocket
- Auto-riconnessione + webhook
- Compatibile Citrix/VDI
- Interfaccia basata su WebView2

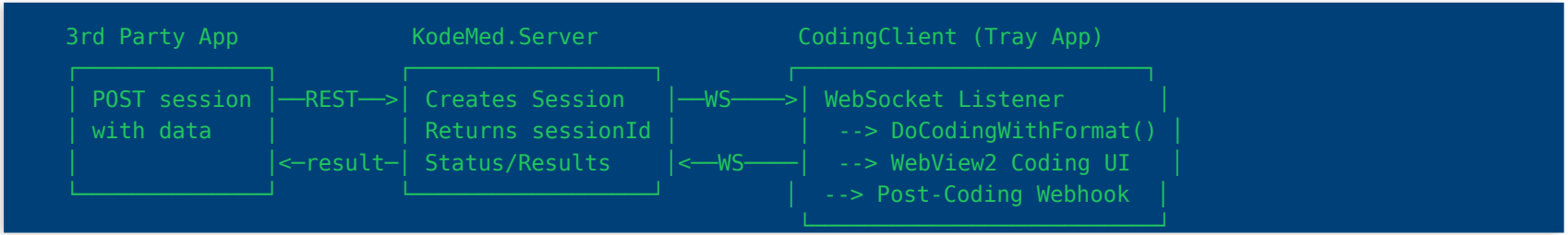
Configurazione (impostata automaticamente dall’installatore MSI o al primo avvio)

Variable	Description	Example
KODEMED_HOME	Installation directory	C:\Users\joe\KodeMed\
KODEMED_DLL_PATH	Full path to KodeMed.dll	C:\Users\joe\KodeMed\KodeMed.dll
KODEMED_EXE_PATH	Path to CodingClient.exe	C:\Users\joe\KodeMed\KodeMed.CodingClient.exe

Architettura CodingClient

Applicazione system tray per l’integrazione di terze parti

Flusso di integrazione



Funzionalità

- Istanza singola (mutex globale)
- Listener WebSocket per CODING_SESSION_LAUNCH
- Webhook post-codifica (fire-and-forget, retry)
- Selettore lingua (DE/FR/IT/EN, menu tray)
- Accesso/disconnessione OAuth2 via menu tray
- Avvio automatico all’accesso Windows (HKCU\Run)
- Apertura sessioni da file locali

Punti di forza Citrix / VDI

- Autonomo: nessuna dipendenza .NET Runtime
- Tutte le scritture a livello utente (HKCU, variabili d’amb.)
- Portatile: copiare la cartella e avviare
- COM via HKCU\Software\Classes (nessun admin)
- Re-setup ad ogni accesso VDI non persistente

Architettura server

Backend Java con API REST e WebSocket

KodeMed.Server (Java)

- Applicazione Spring Boot
- API REST: config, sessioni, audit, istanze, layout, health
- Endpoint WebSocket (/ws/dll, /ws/app)
- Persistenza JPA/Hibernate (H2/PostgreSQL)
- Audit trail (eventi sessione, azioni utente)
- Gestione layout (salvare/caricare/condividere)
- Validazione token OAuth2
- Integrazione grouper (SwissDRG)
- Distribuzione Docker

KodeMed.DataServer

- API dati di classificazione
- Codici ICD-10-GM + descrizioni
- Codici procedure CHOP
- Codici farmaci ATC
- Catalogo grouper SwissDRG
- Ricerca full-text con autocompletamento
- Cataloghi versionati (aggiornamenti annuali)
- API REST con cache

Interfaccia di codifica

Frontend web React/Vite

KodeMed.CodingUI (React + Vite)

- React 18 con TypeScript + Vite
- Gestione stato Zustand
- 14+ blocchi dati (trascinabili, react-grid-layout)
- Tabelle codici modificabili (ICD, CHOP, ATC)
- Ricerca codici con autocompletamento
- Modifica attributi SpiGes per riga
- Formulari dati amministrativi
- Risultati grouper in tempo reale (SwissDRG)
- Interfaccia multilingue (DE/FR/IT/EN)
- Barra cronologia annullamento (navigazione visuale)
- Integrato nella DLL via WebView2
- Modalità browser autonoma (OAuth2/OIDC)
- Aggiornamenti in tempo reale via WebSocket
- Salvare/caricare/condividere layout personalizzati
- Design responsive + tema scuro/chiaro
- Navigazione da tastiera
- Funzionalità export/import

Autenticazione e sicurezza

OAuth2 / OpenID Connect

OAuth2 / OIDC

- Flusso Authorization Code + PKCE
- Compatibile Keycloak, Auth0, Azure AD
- DLL: login tramite browser integrato
- UI: redirect OIDC standard
- Aggiornamento automatico dei token
- Multi-tenant tramite realm
- Controllo accessi basato su gruppi
- Claims utente OpenID Connect

Misure di sicurezza

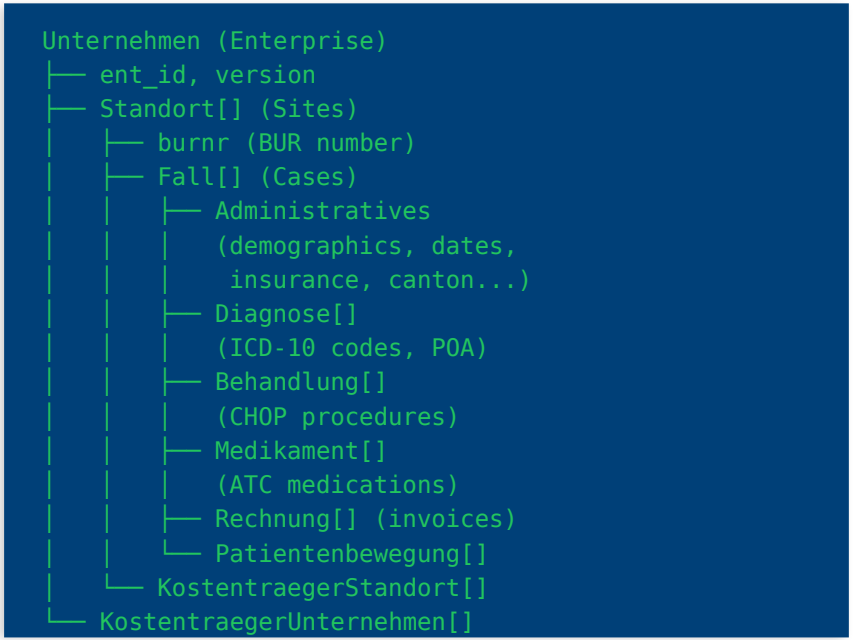
- HTTPS ovunque (TLS 1.2+)
- WSS per le connessioni WebSocket
- Validazione JWT sul server
- Applicazione politiche CORS
- Scadenza sessione (configurabile)
- Logging di audit (tutti gli eventi)

Conformità GDPR / nLPD

- Minimizzazione dati per default (webhook opt-in)
- includeResultData/includeGrouperResults: disattivato
- Protezione SSRF sugli URL webhook
- Nessun segreto nella configurazione client

Modello dati

Struttura XML SpiGes (UST / Ufficio federale di statistica)



Entità principali

Entity	Key Fields
Administratives	abc_fall, alter, geschlecht, ein/austritt
Diagnose	diagnose_kode (ICD-10), diagnose_poa
Behandlung	behandlung_chop (CHOP), behandlung_beginn
Medikament	medi_atc, medi_dosis, medi_einheit
GrouperResult	DRG, MDC, PCCL, cost weight

Webhook post-codifica

Callback HTTP fire-and-forget dopo le sessioni di codifica

Come funziona

- Il codificatore completa la sessione (Applica/Annulla/...)
- CodingClient invia JSON in POST all'URL configurato
- Asincrono: non blocca il codificatore
- Retry con backoff esponenziale (1s→2s→4s...60s)
- Nessun retry su 4xx (eccetto 429)

Configurazione (variabili d'ambiente)

Variable	Default
KODEMED_HOOK_ENABLED	false
KODEMED_HOOK_URL	(empty)
KODEMED_HOOK_INCLUDE_RESULT_DATA	false
KODEMED_HOOK_INCLUDE_GROUPER_RESULTS	false
KODEMED_HOOK_EVENTS	applied

Payload del webhook

```
POST https://his.example.com/api/results
Content-Type: application/json

{
  "eventType": "coding_session_completed",
  "sessionId": "sess_abc123",
  "codingAction": "Applied",
  "applied": true,
  "username": "dr.muller",
  "sourceFormat": "SpiGes",
  "casesProcessed": 3,
  "diagnosesCount": 12,
  "treatmentsCount": 7,
  "resultData": "...(opt-in)...",
  "grouperResults": [...(opt-in)...]
}
```

Auth: token bearer o header personalizzato (solo config locale)

Distribuzione

Docker, on-premise, cloud

Componenti server

```
# Docker Compose
services:
  kodemed-server:
    image: kodemed/server:latest
    ports: ["8080:8080"]

  kodemed-dataserver:
    image: kodemed/dataserver:latest
    ports: ["8081:8081"]

  kodemed-grouper-server:
    image: kodemed/grouper-server:latest
    ports: ["8082:8082"]

  kodemed-ui:
    image: kodemed/codingui:latest
    ports: ["3000:3000"]

# OIDC Provider (external IdP, not bundled)
# https://customer-sso.example.com/auth
```

Distribuzione client

- CodingClient: EXE portatile autonomo
- Copiare in qualsiasi cartella e avviare
- Registrazione automatica DLL + variabili d’ambiente
- Citrix/VDI: pubblicare come app o nell’immagine
- Group Policy: impostare KODEMED_HOME
- Setup silenzioso al primo avvio
- Override configurazione: kodemed-client-config.json
- Aggiornamento automatico: previsto per versioni future

Topologia di distribuzione

Kubernetes / OpenShift con Harbor Registry

Harbor Registry

- kodemed/server:latest
- kodemed/dataserver:latest
- kodemed/grouper:latest
- kodemed/codingui:latest

Trivy scan + Cosign/Notary

pull >>>

Kubernetes / OpenShift

Ingress / Route

Service	HPA	Port
kodemed-server	Yes	:8080
kodemed-dataserver	Yes	:8081
kodemed-grouper	-	:8082
kodemed-ui	-	:3000

PostgreSQL (StatefulSet / external)

- Helm Charts o manifest K8s semplici
- Autoscaling orizzontale dei pod (HPA) per Server & DataServer

- Registro privato per archiviazione e distribuzione immagini
- Scansione vulnerabilità (Trivy) ad ogni push

Configurazione

Config XML/JSON, variabili d’ambiente, webhook

Config DLL / Client (XML o JSON)

```
<KodeMedConfig>
  <ServerUrl>https://api.example.com
  </ServerUrl>
  <CodingUIUrl>https://ui.example.com
  </CodingUIUrl>

  <!-- OAuth2 -->
  <OAuth2Url>https://sso/auth</OAuth2Url>
  <OAuth2Realm>kodemed</OAuth2Realm>
  <OAuth2ClientId>kodemed-dll</OAuth2ClientId>

  <!-- Language (de/fr/it/en, auto-detect) -->
  <Language>de</Language>

  <!-- Processing -->
  <OutputFormat>XML</OutputFormat>
  <ValidateSchema>true</ValidateSchema>
</KodeMedConfig>
```

Impostazioni principali

Setting	Default	Description
ServerUrl	-	Main server URL
OAuth2Url	-	Identity provider
Language	auto	de, fr, it, en
OutputFormat	XML	XML or JSON
SessionExpiry	60	Minutes
OfflineTimeout	5	Minutes

Variabili d’ambiente webhook (server)

Variable	Default
KODEMED_HOOK_ENABLED	false
KODEMED_HOOK_URL	(empty)
KODEMED_HOOK_EVENTS	applied
KODEMED_HOOK_RETRY_COUNT	3

Avvio rapido

Operativo in 3 passaggi

Passaggio 1: Distribuire il server

```
docker compose up -d    # Starts Server, DataServer, GrouperServer, CodingUI
```

Passaggio 2: Installare il client (scegliere un’opzione)

```
# Option A: CodingClient (recommended)
# Just run KodeMed.CodingClient.exe
# Auto-setup handles everything
```

```
# Option B: MSI silent install (GPO/SCCM)
msiexec /i KodeMed.msi /quiet SERVERURL="https://..."
```

Passaggio 3: Iniziare la codifica

```
var coding = new Coding();
coding.SendConfig(File.ReadAllText("kodemed-config.xml"));
coding.DoCodingWithFormat(spigesXml, FormatType.SpiGes);     // Opens auth + coding UI
string results = coding.GetResults();                         // XML or JSON results
```