KodeMed

Architecture du système

Pour les CTOs, l’administration IT & DevOps

Version 2026.5.15.59502 | KodeMed AG

Vue d’ensemble

Composants de la plateforme KodeMed

KodeMed.Server

Backend Java (REST + WebSocket)

KodeMed.DataServer

API de données de classification

KodeMed.GrouperServer

API de groupage DRG (SwissDRG, TARPSY, ST Reha)

KodeMed.CodingUI

Frontend web React/Vite

KodeMed.Interface

DLL COM (.NET 9.0)

KodeMed.CodingClient

Application system tray – EXE portable
WebSocket, webhook, sélecteur de langue

```
Flux : SIH/Tiers → Serveur (REST/WS) → CodingClient (WS) → DLL → CodingUI (WebView2)

Flux : SIH (direct) → DLL (COM) → Serveur → CodingUI (WebView2)
```

Intégration SIH

Intégration du système d’information hospitalier 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

Exemple d’intégration (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();
```

Installation du client

Trois méthodes de déploiement

Installateur MSI (recommandé)

- Package KodeMed.msi
- Par utilisateur (sans admin) ou par machine (Citrix)
- Installation silencieuse via GPO/SCCM/Intune
- Enregistrement COM + configuration inclus
- Runtime WebView2 requis

Manuel / Xcopy

- Copier DLL + dépendances
- Enregistrement COM manuel
- Variables d’environnement manuelles
- Pour utilisateurs avancés

CodingClient

- Installé via MSI
- System tray + WebSocket
- Auto-reconnexion + webhook
- Compatible Citrix/VDI
- Interface basée sur WebView2

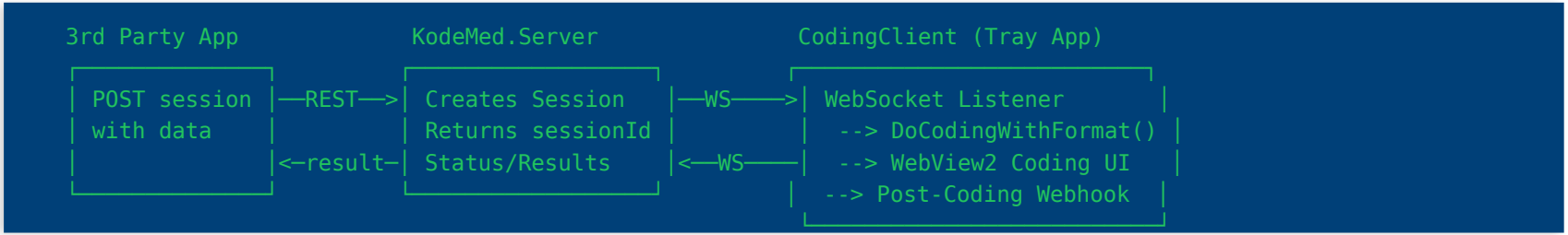
Configuration (définie automatiquement par l’installateur MSI ou au premier lancement)

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

Architecture CodingClient

Application system tray pour l’intégration tierce

Flux d’intégration



Fonctionnalités

- Instance unique (mutex global)
- Écoute WebSocket pour CODING_SESSION_LAUNCH
- Webhook post-codage (fire-and-forget, retry)
- Sélecteur de langue (DE/FR/IT/EN, menu tray)
- Connexion/déconnexion OAuth2 via menu tray
- Démarrage automatique à l’ouverture de session Windows (HKCU\Run)
- Ouvrir des sessions depuis des fichiers locaux

Points forts Citrix / VDI

- Autonome : aucune dépendance .NET Runtime
- Toutes les écritures au niveau utilisateur (HKCU, variables d’env)
- Portable : copier le dossier et exécuter
- COM via HKCU\Software\Classes (pas d’admin)
- Re-setup à chaque connexion VDI non persistante

Architecture serveur

Backend Java avec API REST et WebSocket

KodeMed.Server (Java)

- Application Spring Boot
- API REST : config, sessions, audit, instances, layouts, health
- Endpoint WebSocket (/ws/dll, /ws/app)
- Persistance JPA/Hibernate (H2/PostgreSQL)
- Piste d’audit (événements de session, actions utilisateur)
- Gestion des layouts (sauvegarder/charger/partager)
- Validation des tokens OAuth2
- Intégration groupeur (SwissDRG)
- Déploiement Docker

KodeMed.DataServer

- API de données de classification
- Codes ICD-10-GM + descriptions
- Codes de procédures CHOP
- Codes médicaments ATC
- Catalogue groupeur SwissDRG
- Recherche plein texte avec auto-complétion
- Catalogues versionnés (mises à jour annuelles)
- API REST avec cache

Interface de codage

Frontend web React/Vite

KodeMed.CodingUI (React + Vite)

- React 18 avec TypeScript + Vite
- Gestion d’état Zustand
- 14+ blocs de données (déplaçables, react-grid-layout)
- Tables de codes éditables (ICD, CHOP, ATC)
- Recherche de codes avec auto-complétion
- Édition des attributs SpiGes par ligne
- Formulaires de données administratives
- Résultats groupeur en temps réel (SwissDRG)
- Interface multilingue (DE/FR/IT/EN)
- Barre d’historique d’annulation (parcours visuel)
- Intégré dans la DLL via WebView2
- Mode navigateur autonome (OAuth2/OIDC)
- Mises à jour en direct via WebSocket
- Sauvegarder/charger/partager des layouts personnalisés
- Design responsive + thème sombre/clair
- Navigation au clavier
- Fonctionnalité d’export/import

Authentification & sécurité

OAuth2 / OpenID Connect

OAuth2 / OIDC

- Flux Authorization Code + PKCE
- Compatible Keycloak, Auth0, Azure AD
- DLL : connexion via navigateur intégré
- UI : redirection OIDC standard
- Rafraîchissement automatique des tokens
- Multi-tenant via realms
- Contrôle d’accès par groupes
- Claims utilisateur OpenID Connect

Mesures de sécurité

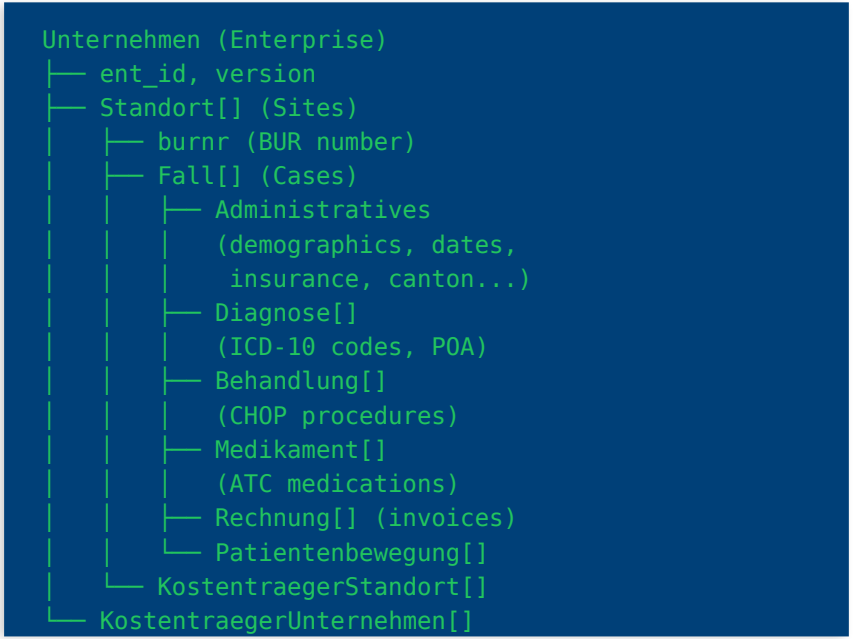
- HTTPS partout (TLS 1.2+)
- WSS pour les connexions WebSocket
- Validation JWT sur le serveur
- Application des politiques CORS
- Expiration de session (configurable)
- Journalisation d’audit (tous les événements)

Conformité RGPD / nLPD

- Minimisation des données par défaut (webhook opt-in)
- includeResultData/includeGrouperResults : désactivé
- Protection SSRF sur les URL de webhook
- Aucun secret dans la configuration client

Modèle de données

Structure XML SpiGes (OFS / Office fédéral de la statistique)



Entités clés

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-codage

Callback HTTP fire-and-forget après les sessions de codage

Fonctionnement

- Le codeur termine la session (Appliquer/Annuler/...)
- CodingClient envoie du JSON en POST à l'URL configurée
- Asynchrone : ne bloque pas le codeur
- Retry avec backoff exponentiel (1s→2s→4s...60s)
- Pas de retry sur 4xx (sauf 429)

Configuration (variables d'env)

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 du 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 ou en-tête personnalisé (config locale uniquement)

Déploiement

Docker, on-premise, cloud

Composants serveur

```
# 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
```

Déploiement client

- CodingClient : EXE portable autonome
- Copier dans n'importe quel dossier et exécuter
- Enregistrement automatique DLL + variables d'env
- Citrix/VDI : publier comme app ou dans l'image
- Stratégie de groupe : définir KODEMED_HOME
- Setup silencieux au premier lancement
- Remplacement de config : kodemed-client-config.json
- Mise à jour automatique : prévue pour les futures versions

Topologie de déploiement

Kubernetes / OpenShift avec 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 ou manifests K8s simples
- Autoscaling horizontal des pods (HPA) pour Server & DataServer

- Registre privé pour le stockage et la distribution d'images
- Analyse de vulnérabilités (Trivy) à chaque push

Configuration

Config XML/JSON, variables d’environnement, webhook

Config DLL / Client (XML ou 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>
```

Paramètres clés

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

Variables d’env webhook (serveur)

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

Démarrage rapide

Opérationnel en 3 étapes

Étape 1 : Déployer le serveur

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

Étape 2 : Installer le client (choisir une option)

```
# 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://..."
```

Étape 3 : Commencer le codage

```
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
```