JGA.1 - Panoramica

GPEnext — Documentazione Completa di Prodotto

> **Versione:** v1-RC1 | **Data:** giugno 2026 | **Stato:** pronto per primo test cliente

---

Indice rapido

1.1 - Panoramica di Prodotto

Cos'è GPEnext

GPEnext è un editor web centralizzato di modelli di stampa per ERP. Consente a operatori tecnici e clienti finali di creare, personalizzare e gestire i template per tutti i documenti generati dall'ERP Genesys (ordini, fatture, DDT, estratti conto, buste paga, etichette, report, ecc.).

È il successore moderno del **GPE** (**G**enesys **P**rint **E**ditor), applicazione desktop proprietaria sviluppata in Java Swing che ha accompagnato l'ERP COBOL legacy per anni.

1.2 - Perché è stato realizzato

I problemi del GPE legacy

Il GPE desktop presentava **19 bug documentati** che ne rendevano l'uso quotidiano difficile per gli operatori:

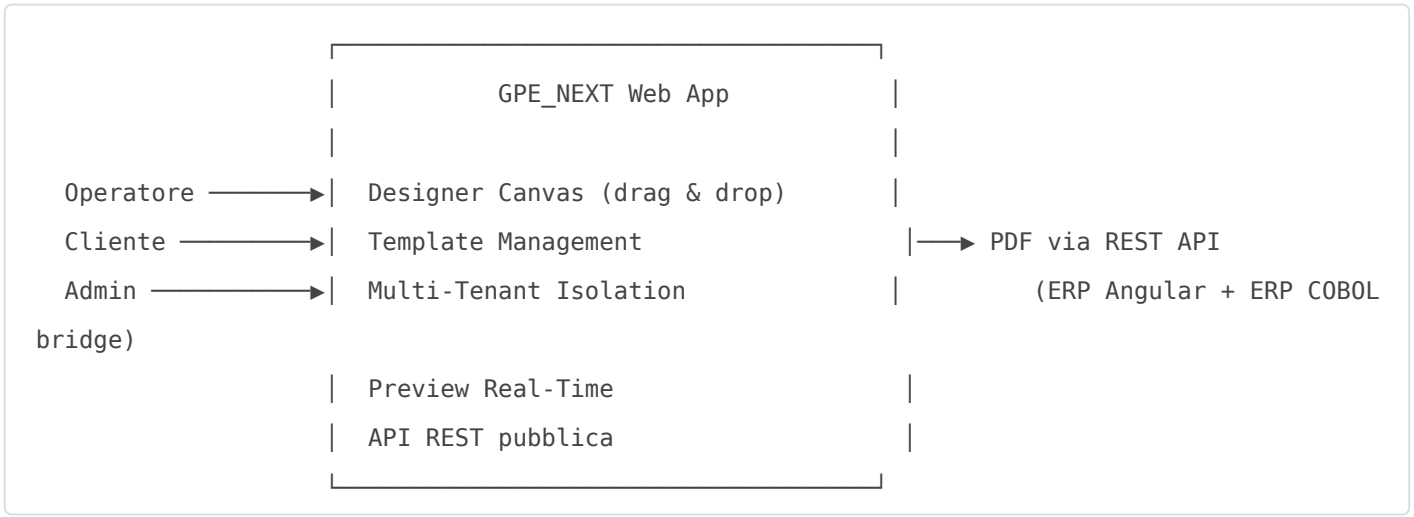
Categoria	Bug	Impatto
Compatibilità	Adobe Acrobat 64bit blocca l'avvio	Blocca l'uso ogni volta
Layout	Sezione testa si azzera dalla pagina 2	Stampe illeggibili
Tabelle	Impossibile inserire colonne se spazio ridotto	Workaround manuali obbligatori
Tabelle	Max 2-3 pagine con conteggio pagina Break corretto	Bug sistematico
Tabelle	Impossibile tabelle affiancate	Limita fortemente il layout
Tabelle	Non gestisce alto numero di colonne XML	Crash o workaround
Immagini	Loghi sgranati in stampa	Qualità professionale impossibile
Multi-cliente	Un modello per tutti: logo ridimensiona la testa	Impossibile per-tenant
Immagini	Resize server-side distrugge la risoluzione	
Numerazione	Numerazione pagine non funzionante o sporca	

Problemi strutturali:

- Applicazione desktop → non distribuibile via web, ogni operatore deve installarla localmente
- Nessuna gestione multi-tenant → impossibile personalizzazioni per cliente
- Formato output `.xslt-xfo` non documentato → dipendenza totale dal vendor
- Nessuna API → integrazione col nuovo ERP Angular+Spring Boot impossibile

---

1.3 - Cosa fa GPE_NEXT (in sintesi)



Flusso operativo (semplificato)

1. L'ERP genera un flusso **XML dati** (ordine, fattura, DDT…)
2. L'operatore carica l'XML in GPE_NEXT e crea/modifica il **modello di stampa** con il designer visuale
3. Il modello viene **pubblicato** in GPE_NEXT e diventa immediatamente disponibile via API REST
4. Quando l'ERP deve stampare:
   - **ERP COBOL eSIGEA**: il bridge integrato nell'ERP chiama `POST /api/render` con l'XML dati → riceve il **PDF finale**
   - **Nuovo ERP Angular+Spring Boot**: chiama `POST /api/render` direttamente → riceve il **PDF finale**

1.4 - Utenti del sistema

OPERATOR (Operatore Genesys)

Staff tecnico Genesys che configura i modelli per i clienti.

- Accesso completo al designer canvas
- Può creare/modificare/pubblicare tutti i modelli
- Gestisce schemi XSD e file XML di test
- Può importare modelli legacy (`.mdl`)
- Vede il codice XSLT/XSL-FO generato

CLIENT (Cliente finale)

Il cliente dell'ERP (es. azienda che usa SIGEA).

- Ha la sua **dashboard** con i documenti disponibili
- Può selezionare il **profilo di stampa** da usare (tra quelli configurati dall'Operatore)
- Può visualizzare l'anteprima dei propri documenti
- **Non** può modificare struttura o template base

SUPER_ADMIN (Administrator Genesys)

Gestione cross-tenant.

- Vede e gestisce tutti i tenant
- Imposta la versione ERP per ogni tenant
- Gestisce le migrazioni forzate di template

1.5 - Valore per il cliente

Prima (GPE Desktop)	Ora (GPE_NEXT)
Desktop Java, installa locale	Web app, qualsiasi browser
Un modello per tutti i clienti	Profili personalizzati per tenant
19 bug documentati che bloccano	Zero bug noti sulle funzionalità core

Prima (GPE Desktop)	Ora (GPE_NEXT)
Nessuna API	REST API documentata (Swagger)
Immagini sgranate	Risoluzione originale preservata
Impossibile multi-tenant	Isolamento totale per-tenant
Nessuna anteprima affidabile	Preview real-time WYSIWYG
Dipendenza da Adobe Acrobat	Nessuna dipendenza
Solo COBOL legacy, output fisso su filesystem	REST API unificata: ERP COBOL bridge + nuovo ERP Angular

---

1.6 - Status attuale (giugno 2026)

GPEnext è pronto per il primo test con cliente reale:

- ✅ **Tutti i 45 task P0** completati (100%)
- ✅ **Feature parity** completa con GPE legacy (v1)
- ✅ **Canvas UX** professionale (snap, guide, zoom, multi-select, undo/redo)
- ✅ **Integrazione Keycloak OIDC** funzionante
- ✅ **900+ test** automatici, 0 failure
- ✅ **Import legacy MDL** funzionante su tutti i modelli testati
- ✅ **Output ERP COBOL** validato su file reali (230+ XML)
- ✅ **Output PDF** via Gotenberg (Chromium) con layout WYSIWYG
- ✅ **Bridge ERP COBOL** operativo (eSIGEA chiama `POST /api/render`)
- ? **Deploy OCI** in preparazione (EPIC A in corso)
- ? **Admin Panel** in pianificazione (TASK-048)

1.7 - Stack tecnologico (sintesi)

Layer	Tecnologia
Frontend	Angular 18+ (standalone), PrimeNG Aura, Angular CDK
Backend	Spring Boot 3.3, Java 21
PDF engine	Gotenberg 8+ (Chromium/Blink via Docker)
XSLT-FO	Apache FOP 2.10 (diagnostica/download)

Layer	Tecnologia
Database	PostgreSQL 16, Flyway, JSONB
Auth	Keycloak OIDC (realm `gsauth`)
Infrastruttura	Docker, Nginx, OCI Linux
Preview browser	Paged.js (preview browser-side)

1.8 - Posizionamento in SIGEAnext

GPEnext è il **Modulo 1** del progetto di evoluzione aziendale **SIGEAnext**:

```
a-SIGEA (futuro)
├── [M1] GPE_NEXT □ ← QUI SIAMO
├── [M2] Nuovo ERP Angular+Spring (in sviluppo)
├── [M3] Strato agentico AI
├── [M4] BI Generativa
├── [M5] Gestione Documentale e Knowledge Management
└── [M6] HAI (Humanized Augmented Intelligence) per Logistica
```

---

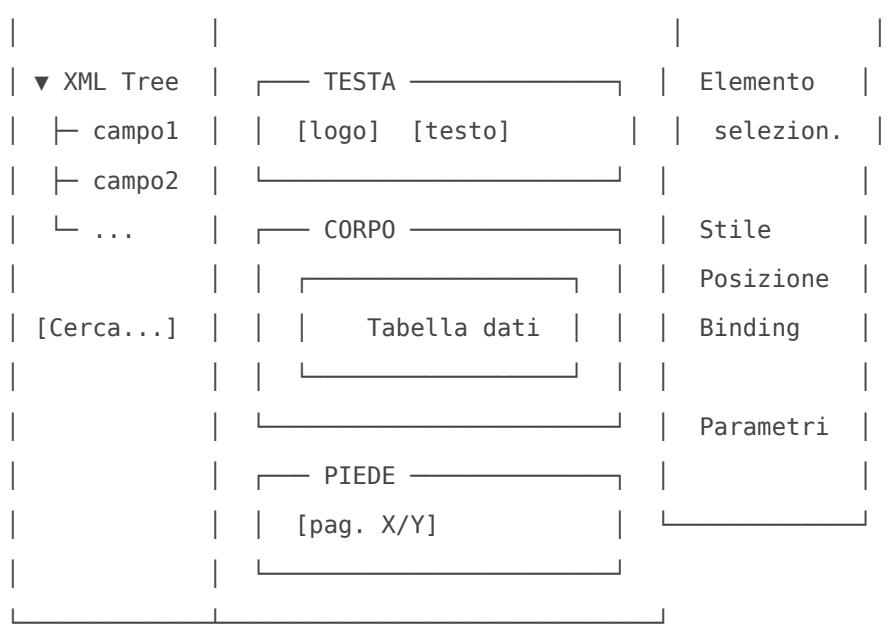
2. Funzionalità Attuali (v1-RC1)

> Tutte le funzionalità elencate sono **implementate e funzionanti** — non sono roadmap.

2.1 Designer Canvas

Il cuore di GPE_NEXT è il **canvas di disegno visuale** a 3 pannelli:

```
┌──────────────┬──────────────────────────────┬──────────────┐
│  Data Tree   │        Canvas Pagina         │  Proprietà   │
```



Sezioni della Pagina

Sezione	Descrizione	ERP equivalente
Testa	Intestazione ripetuta su ogni pagina	`region-before` XSL-FO
Corpo	Contenuto principale (tabelle dati)	`region-body`
Piede	Piè di pagina (numerazione, firme)	`region-after`
Margine Sinistro	Colonna laterale sinistra	`region-start`
Margine Destro	Colonna laterale destra	`region-end`

Elementi del Canvas

Elemento	Descrizione	Binding
Static Field	Testo statico o con segmenti misti	—
Data Field	Campo XML dell'ERP	XPath dal flusso dati
Image/Logo	Immagine o logo aziendale	Path filesystem + variabili `$IMGDIR` / `$PROCLIB`
Barcode	Codice a barre CODE_128	Campo XML
code-xslt-field	Blocco XSLT raw (solo OPERATOR)	XSLT arbitrario

Canvas UX — Strumenti di Disegno

Funzionalità	Shortcut	Note

Snap to Grid	Toggle toolbar	Passo configurabile in mm
Smart Guides	Sempre attivo	Snap a bordi e centri elementi
Righelli mm	`R`	HiDPI, indicatore mouse in tempo reale
Multi-selezione	`Ctrl+Click`, lasso	Bounding box overlay
Allineamento	`Alt+L/C/R/T/M/B`	Align & distribute
Sposta con tasto	`←↑→↓`	Nudge 1mm, `Shift`=10mm
Zoom	`Ctrl+scroll`, `Ctrl+0/1/2`	25–400%
Pan	`Spazio+drag`	
Draw mode	Toggle toolbar	Disegna nuovo elemento
Guide draggabili	Dal righello	Snap per posizionamento
Copy/Paste	`Ctrl+C/V`	Tra sezioni e pagine
Duplicate	`Ctrl+D`	
Z-order	Toolbar	Porta in primo piano / manda in fondo
Undo/Redo	`Ctrl+Z/Y`	Stack completo
Coordinate rapide	`Enter` su elemento	Input numerico mm
Shortcut help	`?` / `F1`	Overlay 5 categorie
Modalità PDF Fidelity	Auto	Chrome nascosto a riposo

Modalità Vista Canvas

- **Struttura** — vede tutti i chrome (bordi, guide, badge)
- **Dati** — mostra i valori XML reali caricati
- **PDF Fidelity** — chrome nascosto, vista "come sarà il PDF"

2.2 Tabelle Dati

Le tabelle sono l'elemento più potente del sistema — coprono il 90% dei documenti ERP.

Tipi di Riga

Tipo	Descrizione	XSLT generato
Header	Intestazione colonna	Statica

Tipo	Descrizione	XSLT generato
Data	Riga dati dal flusso XML	`xsl:for-each`
Total	Riga aggregazione	`xsl:call-template`
Group Break	Salto gruppo / rottura	`xsl:for-each-group`

Funzionalità Tabella

Feature	Dettaglio
Tabelle affiancate	Più tabelle nella stessa sezione (risolve bug GPE legacy)
Nested tables	Fino a 3 livelli di annidamento
Multi-row-type	Più tipi di riga nella stessa tabella
Merge celle	Colspan UI con merge/split visuale
Bordi per cella	Controllo lato per lato (top/right/bottom/left)
Zebra striping	Colorazione alternata righe
Header colorato	Background colore testa
Righe opzionali	`xsl:if` su riga intera
Page break	Controllo salto pagina (keep-together, minRowBefore)
Anchor Row	Gruppi logici che non si spezzano tra pagine
Sort Keys	Ordinamento dati (crescente/decrescente)
Group Break	Intestazione subtotale per ogni gruppo
Aggregate	sum/avg/count/min/max per colonna
RecOutMode	REC_TO_CELL (record → colonne)
Tabelle condizionali	`xsl:if` wrapper sulla tabella intera
Col-span dinamico	merge fino a N colonne

Contenuto Misto nelle Celle

Ogni cella può contenere una sequenza di **segmenti misti**:

```
Cella: [ "Articolo: " | {CODART} | " - " | {DESART} ]
           testo       campo XML   testo   campo XML
```

Funzioni e Macro sui Campi

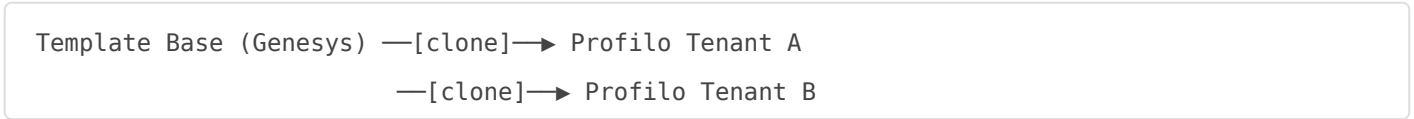
Funzione/Macro	Descrizione
`LOWER()`	Converti in minuscolo
`RPAD()`	Padding destro
`TRIM()`	Rimuovi spazi
`SubstitutionList`	Mappa valori (es. `01→Aperto`, `02→Chiuso`)
`CURR_DATE`	Data corrente
`CURR_TIME`	Ora corrente
`PAGE_NUM`	Numero pagina corrente
`PAGE_TOT`	Totale pagine
Formati COBOL numerici	`N9_999v99s`, `sN9999p99`
Formati COBOL date	`DDpMMpYYYY`, `DDtMMtYYYY`, `YYYYtMMtDD`
XPath parametrici	`xsl:param` per espressioni dinamiche

2.3 Gestione Modelli (Template)

Ciclo di vita del Modello

- **Bozza (Draft):** solo OPERATOR la vede, modificabile
- **Pubblicato:** accessibile ai client, immutabile (solo snapshot)
- **Versioning ERP:** ogni versione ha un `erpMinVersion` — il sistema seleziona automaticamente la versione compatibile con la versione ERP del tenant

Profili Client



Funzioni CRUD

- Crea nuovo modello (wizard 2 step)
- Clona modello esistente
- Importa da file `.mdl` legacy
- Import massivo (bulk) di file `.mdl`
- Modifica in designer, salva bozza, pubblica versione
- Attiva profilo per tenant
- Storico versioni, anteprima versione storica, ripristino versione
- Elimina (con conferma)
- Lista con filtri e ricerca

Multi-Pagina (Multi-PageModel)

Un documento può avere **più modelli pagina** (pagina 1 diversa dalle successive, intestazioni diverse per cliente):

```
Documento: ┌─ PageModel 1 (estr_cont) — ogni cliente
           └─ PageModel 2 (estr_cont_este) — ogni cliente esteso
```

Navigazione tra pagine con tab-bar nel designer. Ogni pagina ha i propri margini, sezioni e contenuti.

2.4 Gestione Schema / Tipo Documento

Flusso XML ? Schema ? Modello

```
ERP genera XML ──► Upload XML in GPE_NEXT
                          │
                          ▼
                Conversione automatica XML→XSD
                          │
                          ▼
                Schema (Tipo Documento) creato
                          │
                          ▼
                Modello di stampa associato allo Schema
```

Funzionalità Schema

- Upload XML → conversione automatica XSD (inferenza tipi COBOL)
- Import XSD manuale
- Carica file XML multipli per lo stesso schema
- Parser XSD → albero visuale dei campi nel designer
- Associazione file XML al modello (per anteprima con dati reali)
- Validazione: impossibile caricare XML senza XSD
- Validazione: impossibile creare modello senza schema associato
- Supporto XPath relativi e assoluti
- Gestione Schema Ditta: XPath per `CODE_DITT` (logo per-azienda)
- Auto-detect XPath ditta dal tree, con auto-detect su import MDL

Import Legacy MDL

GPE_NEXT importa i file `.mdl` del vecchio GPE con **fedeltà completa**:

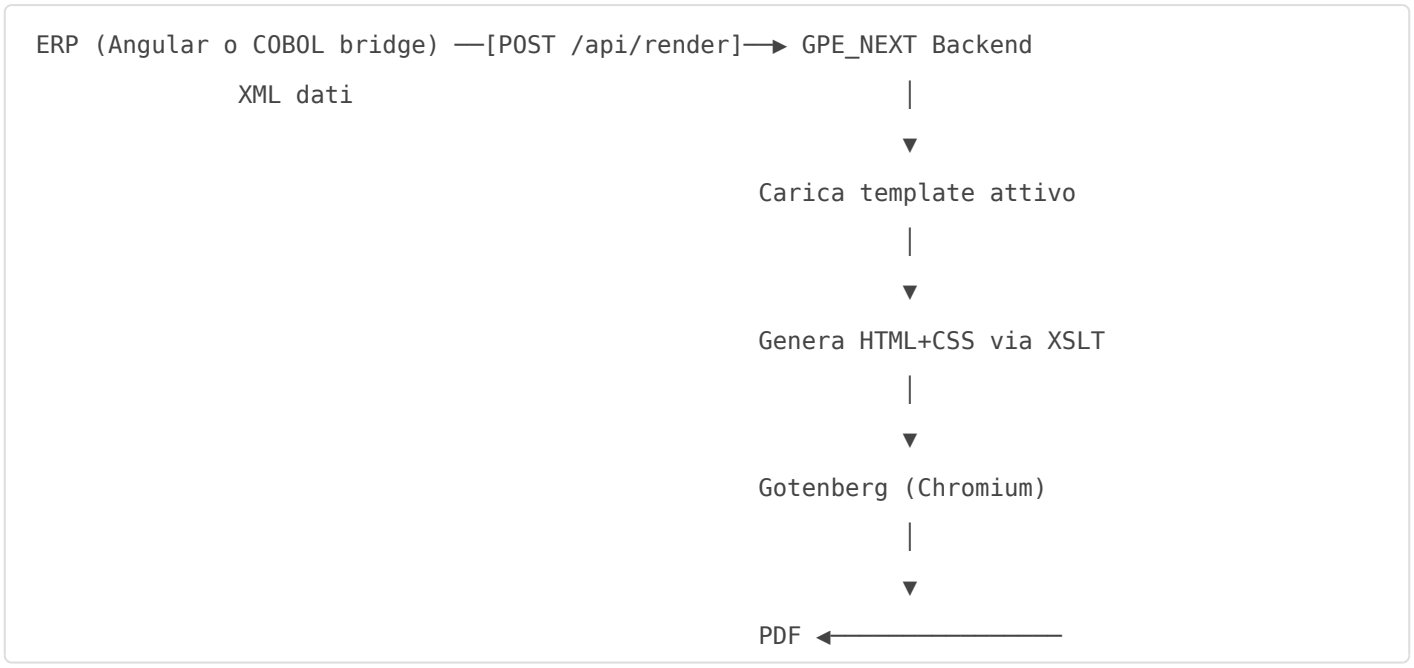
Elemento MDL	Supporto	Note
`page-model`	✅	Mapping completo
`table`	✅	Incluse condizioni
`table-row` (H/D/T/GB)	✅	Tutti i tipi
`table-column`	✅	Larghezze mm
`text-block`	✅	Testo statico e misto
`data-field`	✅	XPath binding
`image`	✅	Path con variabili
`code-xslt-field`	✅	Preservato byte-per-byte
`barcode`	✅	CODE_128
Multi-page	✅	
Condizioni `xsl:if`	✅	Su elementi, righe, tabelle
SubstitutionList	✅	
FuncCall	✅	LOWER, RPAD, TRIM
Macro date/ora	✅	
XPath parametrici	✅	
Regioni LEFT/RIGHT	✅	
Tabelle affiancate	✅	
RecOutMode	✅	REC_TO_CELL

2.5 Preview e Anteprima

- Preview real-time con dati XML reali nel pannello destro
- Rendering via **Paged.js** (browser-side, instant) e **Gotenberg** (PDF reale)
- Auto-generate all'apertura del pannello
- Mostra TUTTE le pagine del documento, numerazione `X/Y` corretta
- **Stampa al Volo (EPO):** override temporaneo di valori XML direttamente dall'anteprima (`contenteditable` su iframe, non persistente)

2.6 Output e Integrazione ERP

PDF via REST API (entrambi gli ERP)



API Render:

- `POST /api/render` — body: XML ERP → response: PDF binario
- Risoluzione automatica tenant dal JWT
- Risoluzione automatica template attivo per tipo documento
- Supporto versioning ERP (usa la versione template compatibile)
- Iniezione logo per-ditta automatica

Integrazione ERP COBOL eSIGEA (bridge REST API)

L'ERP COBOL integra con GPE_NEXT tramite un **bridge integrato nel motore ERP**, che chiama la stessa `POST /api/render` usata dal nuovo ERP Angular. Non è necessaria nessuna operazione manuale di deposit file.

Come funziona:

- Il bridge nell'ERP COBOL chiama `POST /api/render` con l'XML dati e le credenziali JWT
- GPE_NEXT risolve automaticamente il tenant, il template attivo e la versione ERP compatibile
- La risposta è un PDF binario pronto all'uso

Esportazione XSLT-FO (funzionalità secondaria)

GPE_NEXT conserva la capacità di generare e scaricare il file `.xslt-xfo` tramite il pulsante "Scarica XSLT-FO" — utile per ispezione, diagnostica o clienti non ancora migrati al bridge. **Non è il metodo primario di integrazione COBOL.**

Caratteristiche tecniche del file generato:

- Encoding ISO-8859-1, namespace `gs-gpe:`, output autocontenuto
- Post-processing per compatibilità `libGSSPM.dll` (7 trasformazioni)
- Path variabili `$IMGDIR`/`$PROCLIB` preservate come testo letterale

Print API Pubblica (in sviluppo)

Epic **PAPI** pianificato (PAPI-001→012):

- `POST /api/print/{documentType}` — stampa per tipo documento
- Swagger UI navigabile + documentazione OpenAPI 3.0

2.7 Multi-Tenant

- Ogni **tenant** (cliente Genesys) ha i propri dati completamente isolati
- `tenant_id` estratto dal JWT Keycloak (claim `groups`) — MAI da query param
- Provisioning automatico al primo login
- Solo `SUPER_ADMIN` ha vista cross-tenant (endpoint `/api/admin/**`)
- Wizard migrazione 3-step per upgrade forzato versione ERP
- Logica `findBestPublishedVersion` automatica

2.8 Sicurezza e Autenticazione

Feature	Dettaglio
Identity Provider	Keycloak OIDC (`gsauth.genesyssoftware.eu`)
Token	JWT access token + refresh token
Tenant	Derivato da claim `groups` Keycloak
Ruoli	`SUPER_ADMIN`, `OPERATOR`, `CLIENT`
Multi-tenant isolation	Tutti gli endpoint filtrano per `tenant_id`
Rate limiting	`/api/auth/login` protetto
XPath whitelist	Solo XPath sicuri in engine XSLT
MDC logging	Pseudonimizzato (no dati personali nei log)
Error sanitization	Nessun dettaglio interno esposto nelle risposte

2.9 Monitoraggio e Logging

- **Structured logging** JSON (profilo prod) / testo colorato (profilo dev)
- **MDC** con `tenant_id`, `request_id`, `user_id` (pseudonimizzati)
- **Performance logs** su operazioni critiche (generazione PDF, render XSLT)
- **Sentry** self-hosted (configurato, deployment in pianificazione)
- **Actuator** Spring Boot (health check, metrics)

3. Architettura Tecnica

3.1 Vista d'insieme del sistema

```
graph TB
    subgraph "Client Layer"
        B[Browser\nAngular 18+\nPaged.js]
        ERP_ANG[ERP Angular\nnuovo]
        ERP_COBOL[ERP COBOL eSIGEA\nbridge integrato]
```

```
    end

    subgraph "GPE_NEXT Platform"
        NGINX[Nginx\nReverse Proxy]
        FE[Frontend\nAngular SPA]
        BE[Backend\nSpring Boot 3.3]
        DB[(PostgreSQL 16\nFlyway migrations\nJSONB)]
        GTB[Gotenberg 8\nChromium PDF]
        FOP[Apache FOP 2.10\nXSL-FO only]
    end

    subgraph "Identity & Security"
        KC[Keycloak OIDC\ngsauth.genesyssoftware.eu]
    end

    subgraph "XSLT-FO — funzione secondaria"
        XSLOUT[Download .xslt-xfo\nper ispezione/diagnostica]
    end

    B --> NGINX
    ERP_ANG -->|POST /api/render\nXML → PDF| NGINX
    ERP_COBOL -->|POST /api/render\nXML → PDF\nbridge nell'ERP| NGINX
    NGINX --> FE
    NGINX --> BE
    BE --> DB
    BE --> GTB
    BE -->|FOP path solo| FOP
    BE -.->|publish .xslt-xfo\nfunzionalità secondaria| XSLOUT
    B --> KC
    BE --> KC
```

3.2 Stack tecnico dettagliato

Frontend

```
Angular 18+ (standalone components)
├── PrimeNG (Aura dark preset)
```

```
├── Angular CDK Drag & Drop
├── Paged.js (preview browser-side)
├── Lucide Icons (tree-shaken)
├── AutoAnimate (motion)
├── CodeMirror 6 (editor XSLT)
└── Chart.js [pianificato v2]
```

Pattern architetturali: standalone components, Signals per state locale, `inject()`, `ChangeDetectionStrategy.OnPush`, template syntax `@if` / `@for` / `@switch`, SCSS con CSS custom properties.

Backend

```
Spring Boot 3.3 (Java 21)
├── Spring Security + Keycloak JWT
├── Spring Data JPA (PostgreSQL)
├── Flyway migrations (V1→V30+)
├── RestClient (Spring 6) per Gotenberg
├── Saxon-HE (XSLT 2.0 processor)
├── Apache FOP 2.10 (XSL-FO diagnostica)
├── PDFBox (manipolazione PDF)
├── Springdoc OpenAPI [pianificato PAPI]
└── Logback JSON (structured logging)
```

Struttura package:

```
controller/ → service/ → repository/ → model/
                │
                ├── dto/        (API contracts)
                ├── html/       (HTML template engine → Gotenberg)
                ├── xslfo/      (XSL-FO engine → download/diagnostica)
                └── xml/        (XML→XSD conversion, XSD parser)
```

Database

```
-- Schema principale (PostgreSQL 16)
tenants           -- Multi-tenant root
  └─ users        -- Utenti per tenant
  └─ schemas      -- Tipi documento (XSD + XML files)
```

```
  └─ templates    -- Modelli di stampa (JSONB content)
     └─ template_base_versions  -- Versioning draft/publish
  └─ tenant_active_profiles     -- Profilo attivo per tipo doc
  └─ images        -- Loghi e immagini
  └─ refresh_tokens
```

Ogni tabella (tranne `tenants`) ha `tenant_id NOT NULL` con indice. Template content in JSONB. UUID come PK ovunque.

3.3 Flusso dati — Generazione PDF (stesso percorso per entrambi gli ERP)

```
sequenceDiagram
    participant ERP as ERP (Angular o COBOL bridge)
    participant BE as Backend Spring Boot
    participant XSLT as Saxon XSLT 2.0
    participant GTB as Gotenberg (Chromium)
    participant DB as PostgreSQL

    ERP->>BE: POST /api/render\nAuthorization: Bearer JWT\nBody: XML dati ERP

    BE->>BE: Estrai tenant_id dal JWT\n(claim groups)
    BE->>DB: Trova template attivo\nper (tenant_id, document_type, erp_version)
    DB-->>BE: Template JSON (JSONB)

    BE->>XSLT: Trasforma template JSON\nin HTML+CSS XSLT template
    XSLT-->>BE: HTML XSLT template

    BE->>XSLT: Applica XSLT template\nsull'XML dati ERP
    XSLT-->>BE: HTML paginato

    BE->>GTB: POST /forms/chromium/convert/html\nBody: HTML + CSS
    GTB-->>BE: PDF binario

    BE-->>ERP: PDF binario\nContent-Type: application/pdf
```

3.4 Flusso dati — Integrazione ERP COBOL (bridge REST API)

Il bridge integrato nell'ERP COBOL chiama GPE_NEXT esattamente come fa il nuovo ERP Angular.

```
sequenceDiagram
    participant ERP_C as ERP COBOL eSIGEA
    participant BRG as Bridge ERP COBOL
    participant BE as Backend Spring Boot
    participant XSLT as Saxon XSLT 2.0
    participant GTB as Gotenberg (Chromium)
    participant DB as PostgreSQL

    ERP_C->>BRG: Richiesta stampa documento + XML dati
    BRG->>BE: POST /api/render\nAuthorization: Bearer JWT\nBody: XML dati ERP
    BE->>BE: Estrai tenant_id dal JWT
    BE->>DB: Trova template attivo
    DB-->>BE: Template JSON (JSONB)
    BE->>XSLT: Genera HTML+CSS dal template
    XSLT-->>BE: HTML paginato
    BE->>GTB: Converti HTML → PDF
    GTB-->>BE: PDF binario
    BE-->>BRG: PDF binario
    BRG-->>ERP_C: PDF pronto
```

3.4.1 Esportazione XSLT-FO (funzionalità secondaria)

L'operatore può scaricare il file `.xslt-xfo` per ispezione, diagnostica o clienti non migrati al bridge.

```
sequenceDiagram
    participant OP as Operatore
    participant FE as Frontend Angular
    participant BE as Backend Spring Boot
    participant GEN as XslFoGeneratorService
    participant POST as XslFoErpPostprocessor
```

```
    OP->>FE: Click "Scarica XSLT-FO"
    FE->>BE: POST /api/templates/{id}/publish-xslfo
    BE->>GEN: generate(templateJson)
    GEN-->>BE: XSLT-FO (namespace gs-gpe:,\nISO-8859-1, autocontenuto)
    BE->>POST: apply ERP postprocessing\n(7 trasformazioni compatibilità)
    POST-->>BE: XSLT-FO finale
    BE->>BE: Valida well-formedness XML
    BE-->>FE: 200 OK + bytes file\n+ header X-Published-Path
    FE-->>OP: Download file .xslt-xfo
```

3.5 Architettura Multi-Tenant

```
graph LR
    JWT[JWT Token\nclaim groups:\n/client/TENANT_A]

    subgraph "Request Pipeline"
        F1[TenantContextFilter] --> F2[JwtAuthenticationFilter]
        F2 --> F3[Controller]
        F3 --> F4[Service]
        F4 --> F5[Repository]
    end

    JWT --> F1
    F1 -->|tenant_id estratto\ne messo in ThreadLocal| F4
    F5 -->|findByIdAndTenantId\nMAI findById| DB[(PostgreSQL)]
```

Regola assoluta: ogni query al DB deve includere `tenant_id`. Unica eccezione: endpoint `/api/admin/**` con `@PreAuthorize("hasRole('SUPER_ADMIN')")`.

3.6 Engine XSLT-FO — Struttura interna

```
XslFoGeneratorService.generate(GpeTemplateDto)
├── writeXslHeader()          → dichiarazione, namespace, encoding
```

```
├── writeParams()              → xsl:param per campi ERP
├── writeUtilityTemplates()    → template riutilizzabili inline
├── writeSimplePageMaster()    → layout pagina (margini, regioni)
│   └── [per ogni PageModel]
├── writePageSequenceMaster()  → alternanza pagine (first/rest/last)
└── writePageSequence()        → contenuto
    ├── writeStaticContent()   → TESTA + PIEDE (region-before/after)
    └── writeFlowContent()     → CORPO (region-body)
        └── [per ogni tabella] → writeTable()
```

Post-processor ERP (7 trasformazioni per `libGSSPM.dll`):

1. Aggiunge `xsl:template match="/"` wrapper
2. Rimuove `border-collapse` da `fo:table`
3. Fix attributi `fo:table-column`
4. Fix `fo:table-body` attributi extra
5. Rimuove wrapper `fo:block` da `fo:static-content`
6. Sostituisce variabili `$IMGDIR` / `$PROCLIB` con testo letterale
7. Fix namespace `gs-gpe:style-list`

3.7 Engine HTML (per Gotenberg) — Struttura interna

```
HtmlTemplateService
├── generateXsltTemplate(template)   → produce XSLT template (HTML)
│   ├── buildPageCss()               → CSS Paged Media (@page, @media print)
│   ├── HtmlPageLayoutRenderer
│   │   ├── renderHeader()
│   │   ├── renderBody()
│   │   └── renderFooter()
│   └── buildGlobalCss()             → .page-layout, .row-anchor-group
├── HtmlTableRenderer
│   ├── renderMergedDataRows()       → xsl:for-each + anchor groups
│   ├── renderHeaderRows()
│   └── renderTotalRows()
└── HtmlFieldRenderer

PdfGenerationService
```

```
├── Pass 1: render HTML → conta pagine (PDFBox)
├── Pass 2: inject page-total nelle <span> pre-riempite
└── Pass 3: chiamata Gotenberg → PDF finale
```

3.8 Performance Budget (Frontend)

Asset	Limite
JS primo route	< 180kb gzip
CSS totale	< 45kb gzip
Font Inter variable	< 100kb woff2
Font JetBrains Mono	< 50kb woff2
Icone Lucide	< 15kb gzip

3.9 Deployment (OCI Linux)

```
graph TB
    subgraph OCI_SERVER["OCI Linux Server"]
        subgraph DOCKER["Docker Compose"]
            NGINX2[nginx 80/443]
            FE2[frontend Angular + nginx]
            BE2[backend Spring Boot]
            DB2[(PostgreSQL 16)]
            GTB2[Gotenberg 3000]
        end
        XSLOUT[XSLT-FO download path]
    end

    INTERNET[Internet] --> NGINX2
    NGINX2 --> FE2
    NGINX2 --> BE2
    BE2 --> DB2
    BE2 --> GTB2
    BE2 -.-> XSLOUT
```

Environment variables critiche:

```
JWT_SECRET=...
DB_PASSWORD=...
GOTENBERG_URL=http://gotenberg:3000
XSLFO_OUTPUT_PATH=/gsnr/gpe-next/data/xslfo
KEYCLOAK_ISSUER_URI=https://gsauth.genesyssoftware.eu/realms/gsauth
SPRING_PROFILES_ACTIVE=production
```

3.10 Database Schema — Entità principali

```
erDiagram
    tenants ||--o{ users : "ha"
    tenants ||--o{ schemas : "possiede"
    tenants ||--o{ templates : "possiede"
    tenants ||--o{ images : "possiede"
    tenants ||--o{ tenant_active_profiles : "ha"

    schemas ||--o{ schema_xml_files : "ha"
    schemas ||--o{ templates : "riferimento"

    templates ||--o{ template_base_versions : "versioni"
    templates ||--o{ tenant_active_profiles : "profilo attivo"

    tenants {
        UUID id PK
        string slug UK
        string company_name
        UUID company_logo_image_id FK
        string keycloak_group_path UK
        string erp_version
        TIMESTAMPTZ created_at
    }

    schemas {
        UUID id PK
        UUID tenant_id FK
```

```
        string name
        string document_type
        string document_label
        string erp_min_version
        text xsd_content
        string ditta_code_xpath
        boolean force_migration
        TIMESTAMPTZ updated_at
    }

    templates {
        UUID id PK
        UUID tenant_id FK
        UUID base_schema_id FK
        string name
        string profile_name
        boolean is_base_template
        UUID xml_file_id FK
        JSONB content
        JSONB draft_content
        TIMESTAMPTZ created_at
        TIMESTAMPTZ updated_at
    }

    images {
        UUID id PK
        UUID tenant_id FK
        string filename
        string display_name
        string content_type
        bytea data
    }
```

6. Ecosistema a-SIGEA / SIGEA-Next

6.1 Il contesto: 40 anni di ERP COBOL

Genesys Software S.r.l. ha sviluppato in quattro decenni un ecosistema ERP proprietario completo — **eSIGEA** / **SIGEAdb** — che gestisce l'intero ciclo aziendale:

```
eSIGEA / SIGEAdb (COBOL legacy)
├── Magazzino · Logistica (picking, packing, spedizioni)
├── Acquisti · Ciclo Passivo (fatture fornitori, pagamenti)
├── Amministrazione · Contabilità · Finanza
├── Produzione · Commerciale
└── Ciclo Attivo (offerte, ordini, fatture clienti, DDT)
```

Caratteristiche AS-IS:

- Thin client (interfaccia testuale o UI leggera)
- Database proprietario (rimane tale — **non viene migrato**)
- Comunicazione via API/TCP tra thin client e backend COBOL
- Stampe documenti via GPE (editor desktop proprietario) + XSLT-FO + libGSSPM.dll

---

6.2 Il progetto a-SIGEA

Il progetto **a-SIGEA** (agentic-SIGEA) è il piano strategico di modernizzazione dell'intero ecosistema, finanziato dal bando **STEP - Regione Puglia** e con il supporto scientifico del **Politecnico di Bari (DMMM)**.

Due direttive di ricerca:

1. Evoluzione agentica della piattaforma SIGEA — architettura microservizi + AI
2. HAI per la Logistica — Humanized Augmented Intelligence, AR + AI per operatori logistici

Transition AS-IS ? TO-BE

```
graph LR
    subgraph "AS-IS (oggi)"
        TC1[Thin Client legacy]
        ERP1[eSIGEA COBOL monolite]
        DB1[(Database proprietario eSIGEA\n– invariato)]
        GPE1[GPE Desktop stampe]
    end
```

```
    subgraph "TO-BE (a-SIGEA target)"
        TC2[Angular SPA nuovo ERP]
        MS[Microservizi Spring Boot]
        DB_NEW[(PostgreSQL\nnuovi servizi\nGPE_NEXT + nuovo ERP)]
        AI_LAYER[Strato Agentico AI]
        DOCS[Gestione Documentale\n+ Knowledge Management]
        BI[BI Generativa]
        HAI2[HAI Interface AR + smart glasses]
        GPE2[GPE_NEXT  stampe web]
    end

    TC1 -.->|evolve| TC2
    ERP1 -.->|coesiste e usa bridge API| GPE2
    GPE1 -.->|sostituito da| GPE2
```

> **Nota:** il database proprietario di eSIGEA **non migra** a PostgreSQL. PostgreSQL è il database di GPE_NEXT e del nuovo ERP Angular — sistemi nuovi che si affiancano all'ERP COBOL esistente, non lo sostituiscono in blocco.

6.3 Posizionamento di GPE_NEXT

```
graph TB
    subgraph "a-SIGEA — Mappa Moduli"
        M1[GPE_NEXT\nEditor Stampe Web\n PRONTO]
        M2[Nuovo ERP\nAngular + Spring Boot\n? In sviluppo]
        M3[Strato Agentico\nMicroservizi AI\n? Pianificato]
        M4[BI Generativa\nReportistica NL\n? Pianificato]
        M5[Gestione Documentale\nKnowledge Management\n? Pianificato]
        M6[HAI Logistica\nAR + Smart Glasses\n? Ricerca]
    end

    M1 ---|integra con| M2
    M2 ---|usa| M3
    M3 ---|alimenta| M4
    M3 ---|gestisce| M5
```

```
    M3 ---|potenzia| M6
    M1 ---|output documenti per| M4
```

GPE_NEXT oggi:

- Modulo 1 completato di a-SIGEA
- Risolve un pain point immediato e misurabile (stampe)
- Genera valore per i clienti correnti senza aspettare altri moduli
- Pone le basi architetturali (multi-tenant, REST API, JWT Keycloak) riutilizzabili dai moduli successivi

---

6.4 Come GPE_NEXT connette i due mondi ERP

Entrambi gli ERP ottengono i PDF attraverso la **stessa API REST** di GPE_NEXT. La differenza è solo nel chiamante.

```
graph TB
    subgraph "ERP COBOL eSIGEA (legacy)"
        ERP_C[eSIGEA COBOL\ngenera XML dati]
        BRIDGE[Bridge integrato\nnell'ERP COBOL]
        PDF_C[PDF output per utente]
    end

    subgraph "GPE_NEXT"
        GPN_API[POST /api/render\nREST API]
        HTML_GEN[HtmlTemplateService\n+ Saxon XSLT 2.0]
        GOTENBERG[Gotenberg\nChromium PDF]
        DB[(PostgreSQL\ntenant + template)]
    end

    subgraph "Nuovo ERP Angular"
        ERP_A[Nuovo ERP Angular\ngenera XML dati]
        PDF_A[PDF output WYSIWYG]
    end

    ERP_C --> BRIDGE
    BRIDGE -->|POST /api/render\nBearer JWT + XML| GPN_API
```

```
ERP_A -->|POST /api/render\nBearer JWT + XML| GPN_API
GPN_API --> DB
GPN_API --> HTML_GEN --> GOTENBERG
GOTENBERG -->|PDF| BRIDGE --> PDF_C
GOTENBERG -->|PDF| PDF_A
```

> **XSLT-FO:** GPE_NEXT conserva la funzionalità di generare file `.xslt-xfo` scaricabili — strumento di ispezione/diagnostica, non il flusso stampa attivo.

6.5 GPE_NEXT come foundation per i moduli futuri

Integrazione con Modulo 2 (Nuovo ERP) — operativa

GPE_NEXT è **già integrato** con il nuovo ERP Angular:

- `POST /api/render` — lo stesso endpoint usato dal bridge COBOL
- JWT multi-tenant compatibile con Keycloak (stesso realm `gsauth`)
- Template versioning con selezione automatica per versione ERP
- Wizard migrazione template per upgrade versione ERP

Questa è l'unica integrazione diretta e confermata tra GPE_NEXT e gli altri moduli a-SIGEA.

Relazione con M3, M4, M5, M6 — nessuna integrazione pianificata

GPE_NEXT è un **modulo verticale specifico** (stampe): risolve un problema preciso e lo fa bene. Non è una piattaforma trasversale.

- **M3 Strato Agentico**: si collegherà direttamente al DB dell'ERP — non passerà per GPE_NEXT
- **M4 BI Generativa**: lavora con dati strutturati dal DB — i PDF non sono input utili alla BI
- **M5 Gestione Documentale**: se e quando esisterà, potrà archiviare i PDF generati da GPE_NEXT, ma è una relazione indiretta e non pianificata

- **M6 HAI Logistica**: GPE_NEXT genera i documenti logistici (DDT, etichette) che gli operatori usano — ma l'interfaccia AR/smart glasses è un progetto separato senza integrazione diretta con GPE_NEXT

GPE_NEXT porta valore **autonomamente**, indipendentemente dall'evoluzione degli altri moduli.

6.6 Integrazioni esterne identificate

Sistema	Tipo integrazione	Status
ERP COBOL (eSIGEA)	Bridge nell'ERP chiama `POST /api/render`	✅ Operativo
Nuovo ERP Angular (M2)	REST API `POST /api/render` — stessa API del COBOL	✅ Operativo
Keycloak OIDC	Auth SSO (realm `gsauth`)	✅ Operativo
Gotenberg (Chromium)	PDF generation interna	✅ Operativo
Viewer XML in-browser (BRG)	XSL bridge per aprire XML ERP nel browser	? Pianificato
Altri moduli a-SIGEA (M3/M4/M5/M6)	Nessuna integrazione pianificata	—

6.7 Integrazioni ERP — dettaglio

Bridge eSIGEA ? GPE_NEXT (operativo)

Il bridge integrato nell'ERP COBOL consente a eSIGEA di chiamare GPE_NEXT tramite la sua REST API. Quando eSIGEA deve stampare:

1. eSIGEA genera l'XML dati del documento
2. Il bridge chiama `POST /api/render` con il JWT di autenticazione e l'XML come body
3. GPE_NEXT risolve il tenant, il template attivo e la versione ERP compatibile
4. GPE_NEXT genera HTML → Gotenberg → restituisce il PDF al bridge
5. eSIGEA consegna il PDF all'utente

Viewer XML in-browser (BRG — pianificato)

Un XSL "ponte" separato (`gpenext-bridge.xsl`, epic BRG-001→005) è pianificato per permettere di aprire qualsiasi XML ERP direttamente nel browser come visualizzazione PDF, senza passare dall'applicazione desktop. Questo epic è **distinto e separato** dal bridge operativo — è uno

strumento UI per operatori, non un'integrazione di backend.

6.8 Timeline evolutiva

```
gantt
    title a-SIGEA — Timeline evolutiva (stima)
    dateFormat  YYYY-MM
    axisFormat  %m/%Y

    section GPE_NEXT (M1)
    Sviluppo core              :done, 2026-03, 2026-05
    Feature parity + UX        :done, 2026-04, 2026-06
    Primo test cliente          :active, 2026-06, 2026-07
    Deploy produzione           :2026-07, 2026-08
    v2 features (grafici, etc.) :2026-08, 2026-12

    section Nuovo ERP (M2)
    Sviluppo in corso           :active, 2025-01, 2027-06
    Integrazione GPE_NEXT       :2026-06, 2026-10
    Beta clienti                :2027-01, 2027-06

    section Strato AI (M3)
    Ricerca industriale (RI)    :2026-07, 2027-01
    Sviluppo microservizi       :2027-01, 2028-06

    section BI Generativa (M4)
    Sviluppo                    :2027-06, 2028-12

    section HAI Logistica (M6)
    Ricerca (TRL5→TRL8)         :2026-07, 2028-12
```

6.9 Valore strategico del progetto

Per Genesys Software

- **Differenziazione competitiva:** unico ERP COBOL italiano con modulo web di design stampe
- **Fidelizzazione clienti:** migliore UX → maggiore soddisfazione → minor churn
- **Scalabilità:** architettura web multi-tenant → nuovi clienti senza installazioni locali
- **Foundation tecnologica:** pattern architetturali riusabili per tutti i moduli a-SIGEA

Per i clienti dell'ERP

- **Autonomia:** possono personalizzare i propri modelli senza dipendere da Genesys
- **Velocità:** nuovo modello in ore invece che giorni
- **Qualità:** output PDF professionale con preview WYSIWYG
- **Continuità:** compatibilità totale con il vecchio COBOL durante la transizione

Nel contesto di a-SIGEA

GPE_NEXT è il **Modulo 1 completato** di a-SIGEA: risolve un problema reale e immediato (le stampe ERP) e consegna valore ai clienti oggi, prima ancora che gli altri moduli esistano. Il suo ruolo nel piano a-SIGEA è questo — non è un layer trasversale, è un pezzo autonomo che funziona da solo e si integra concretamente solo con il nuovo ERP (M2).

---

7. Roadmap

Legenda

Simbolo	Significato
□	Completato
?	In corso
?	Pianificato (backlog)
?	Futuro (v2+)
?	Bloccante / Alta priorità

7.1 v1 — Release corrente (giugno 2026)

Funzionalità completate ?

Foundation:

- Design system SCSS + PrimeNG Aura dark
- Backend Spring Boot + PostgreSQL + Flyway
- JWT Auth + Keycloak OIDC multi-tenant
- Multi-tenant isolation (tenant da JWT groups)

Core Engine:

- XML → XSD conversion engine (230+ XML ERP testati)
- XSD Parser + Data Field Tree API
- Template JSON model (dual-side TS+Java)
- Template CRUD REST API
- XSLT-FO Generator Engine (piena compatibilità COBOL, per download/diagnostica)
- PDF Engine via Gotenberg (Chromium, WYSIWYG)
- 7 postprocessing fixes per `libGSSPM.dll`

Designer Canvas:

- Canvas 3-pannelli (data tree, canvas, property panel)
- Multi-sezione (Testa, Corpo, Piede, Left, Right)
- Elementi: Static, Data, Image, Barcode, code-xslt-field
- Snap to grid, Smart guides, Righelli mm
- Multi-select, Align & Distribute
- Arrow key nudge, Draw mode, Guide draggabili
- Zoom/Pan (25–400%), Position overlay
- Copy/Paste/Duplicate, Z-order
- Undo/Redo stack completo
- Shortcut help overlay (F1)
- Keyboard accessibility WCAG 2.2 AA
- Canvas PDF-Fidelity mode
- Onboarding tour guidato (7 step)

Tabelle:

- 4 tipi riga (Header/Data/Total/Group Break)
- Tabelle affiancate, nested (3 livelli)
- Merge celle (colspan), bordi per cella

- Zebra striping, zebra per modulo
- Page break, anchor row, keep-together
- Sort keys, aggregate, group break
- SubstitutionList, FuncCall, Macro
- RecOutMode REC_TO_CELL
- Righe opzionali, condizioni tabella
- Cell content editor (segmenti misti)

Multi-pagina:

- Multi-PageModel (layout diversi per pagina)
- Navigazione tab-bar tra pagine
- XSL-FO multi-page-sequence
- Preview multi-pagina

Output e Integrazione:

- Live preview (Paged.js + Gotenberg)
- Viewer codice XSLT/XSL-FO (CodeMirror)
- REST API render `POST /api/render` per ERP (usata da nuovo ERP Angular + bridge ERP COBOL)
- Integrazione ERP COBOL eSIGEA via bridge REST API (bridge operativo in eSIGEA)
- Esportazione XSLT-FO (download file `.xslt-xfo` — funzionalità secondaria/diagnostica)
- Template versioning (draft/publish, storico versioni, ripristino)
- ERP version management + wizard migrazione
- Client dashboard + profili
- MDL legacy import (singolo + bulk)
- Stampa al volo (Ephemeral Print Override)

Retrocompatibilità XSL-FO ERP:

- Fix parametri XPath (`rawSelect` as-is) — TASK-149 □
- Fix `$IMGDIR`/`$PROCLIB` come testo letterale — TASK-156 □
- Fallback path immagini `src-alt-N` — TASK-150 □
- Fix `align` dual-output (ERP `fo:external-graphic` vs FOP `fo:block`) — TASK-157 □

Sicurezza:

- Rate limiting login, XPath whitelist
- MDC pseudonimizzato, error sanitization
- Structured logging JSON

7.2 v1.1 — Q3 2026 (in pianificazione)

Priorità alta ?

Feature	Epic ID	Note
Admin Panel (Utenti/Tenant)	TASK-048	Gestione tenant da UI
Client Personalization avanzata	TASK-032	Personalizzazioni per-tenant
Print API pubblica + Swagger	PAPI-001→012	API documentata per nuovo ERP
Viewer XML in-browser (BRG)	BRG-001→005	Apri XML ERP direttamente nel browser
Elementi liberi nel Body	TASK-124	Free-position nel body

Priorità media

Feature	Epic ID	Note
Sentry self-hosted OCI	MANUAL-005	Error tracking
Bulk Format Tabella	TASK-089	Formattazione massiva
Stile Condizionato per Cella	TASK-093	CSS condizionale celle
Gestione Logo Per-Tenant	TASK-151	Gallery logo per ditta
Custom Fields Ditta	DCF-001→003	Equivalente custom_tag.xml
Import MDL: Warning elementi	BULK-006/007	Visibilità parsing warning
Virtual Scrolling Data Tree	TASK-039	Performance tree grandi

7.3 v2 — Q4 2026 / Q1 2027

Wizard Grafico (Charts)

```
graph LR
    XML[Dati XML ERP] --> AGG[ChartDataAggregatorService\nsum/avg/count per XPath]
    AGG --> CHART_WIZ[Wizard 4 step\ntipo · dati · stile · preview]
    CHART_WIZ --> CANVAS_EL[ChartCanvasElement\nChart.js preview]
    CHART_WIZ --> SVG_BE[SVG renderer Backend\nApache Batik]
    SVG_BE --> PDF_CHART[Grafico nel PDF]
```

Bar chart, Line chart, Pie chart da dati XML aggregati via XPath — output SVG nel PDF.

Funzionalità aggiuntive v2

Feature	ID	Note
Anteprima massiva PDF (ZIP)	PDF-001→005	Batch preview tutti i modelli
Macro PAGE_NUM_INC / PAGE_TOT_INC	TASK-080	Numerazione avanzata
Mini-Outline Template Navigator	TASK-082	Navigazione struttura
Dark/Light Theme Toggle	TASK-046	Per designer
Advanced Template Search & Filter	TASK-045	Ricerca avanzata
CI/CD GitHub Actions	DEPLOY-019	Pipeline automatica

7.4 v3 — 2027 (visione)

Integrazione Strato Agentico

```
graph TB
    USER([Operatore]) --> AI[AI Agent\nstrato agentico a-SIGEA]
    AI -->|Crea modello fattura| GPN_API[GPE_NEXT API]
    GPN_API -->|Analizza XSD| SCHEMA_ANALYSIS[Schema Analysis]
    SCHEMA_ANALYSIS -->|Proposta layout| AI
    AI -->|Crea template| GPN_API
    GPN_API -->|Template generato| PREVIEW[Preview automatica]
    PREVIEW --> USER
    USER -->|Approva| PUBLISH[Pubblica]
```

- Template generation via AI: AI analizza XSD e propone layout automatico
- Natural language configuration: "sposta il logo in alto a destra"
- Smart field mapping: AI suggerisce binding XPath contestuale

Integrazione BI Generativa e HAI Logistica

- GPE_NEXT espone metadati semantici dei documenti → BI Generativa aggrega e analizza
- Template stampa documenti logistici (DDT, etichette) → stampa da smart glasses AR

7.5 KPI

KPI	Target v1	Target v2
Template importati da legacy MDL	100% fedeltà	—
Tempo creazione modello vs GPE	-60%	-80%
Bug segnalati in produzione	< 5 critici	0
Tempo risposta API render	< 2s	< 1s
Clienti su piattaforma	3-5 pilota	20+
Tipi documento supportati	230+	300+
Test automatici passing	900+	1200+

Documento generato: giugno 2026 — versione consolidata da 8 file di documentazione

---

Revisione #12
Creato 22 giugno 2026 09:41:45 da Giuseppe Anelli
Aggiornato 6 luglio 2026 19:15:50 da Giuseppe Marangi