Phoenix

Specifiche Gestionale RehabSphere e Integrazione con Agente AI

[Progetto "*RehabSphere - Cutting-Edge Virtual Reality Platform for Comprehensive Patient-Centric Rehabilitation*"; Codici di progetto CUP E79J24004360004 COR 23005670 e COVAR 1499428. Progetto finanziato dall'Unione europea – Next Generation EU - PNRR – Missione 4, Componente 2, Investimento 1.5 "Ecosistemi dell'Innovazione", Programma di Ricerca dell'Ecosistema dellInnovazione Interconnected Nord-Est Innovation Ecosistem I-NEST, codice identificativo ECS00000043 e CUP E63C22001030007Spoke 2 (Health, Food & Lifestyles) – Università di Trento.]

Phoenix

Indice

Phoenix

Proprietà del documento

Registro delle modifiche

Edizione	Revisione	Data	Modifiche
01	00	23.04.2025	

Scopo

Questo documento definisce le specifiche del sistema gestionale di di RehabSphere.

Destinatari

Il seguente documento è destinato ai fornitori per l'elaborazione di una proposta tecnico economica per la realizzazione del sistema gestionale di RehabSphere (capitali da 2 a 8) e integrazione con un Agente AI esterno (capitolo 9).

Acronimi e definizioni

VR: Virtual Reality – Realtà Virtuale

AR: Augmented Reality – Realtà Aumentata

API: Application Programming Interface

MFA: Multi-Factor Authentication

GDPR: General Data Protection Regulation – Regolamento Europeo per la protezione dei dati

RBAC: Role-Based Access Control – Controllo di accesso basato sui ruoli

RTO/RPO: Recovery Time Objective / Recovery Point Objective – Obiettivi di ripristino

RACI: Responsible, Accountable, Consulted, Informed – Modello di responsabilità

SLA: Service Level Agreement – Accordo sui livelli di servizio

KPI: Key Performance Indicator – Indicatore chiave di prestazione

Phoenix

1 Descrizione di RehabSphere

Il progetto RehabSphere si colloca nel settore della riabilitazione neuromotoria e post-operatoria, sfruttando tecnologie di realtà virtuale (VR), intelligenza artificiale (IA) e machine learning (ML) per migliorare l'efficacia dei trattamenti fisioterapici. L'approccio tradizionale alla riabilitazione prevede sessioni supervisionate in presenza, con limitazioni legate alla disponibilità di fisioterapisti, ai costi e agli spostamenti dei pazienti. RehabSphere propone una soluzione innovativa che combina:

- Ambienti VR immersivi e interattivi per migliorare l'engagement dei pazienti durante gli esercizi.
- Interfaccia utente (UI) accessibile e intuitiva, adatta a diverse tipologie di utenti.
- Architettura software scalabile, con backend a microservizi e API RESTful per la gestione dei dati e delle sessioni.
- Agente intelligente basato su NLP, in grado di fornire supporto informativo e assistenza in linguaggio naturale.
- Database progettato per archiviare dati strutturati e non strutturati (video, immagini, parametri biometrici).
- Algoritmi di machine learning per l'analisi di grandi volumi di dati clinici al fine di identificare pattern e predire i risultati dei trattamenti, migliorando così l'efficacia delle terapie.

Grazie a RehabSphere, i pazienti potranno svolgere la riabilitazione in modo più comodo, personalizzato ed efficace, riducendo i tempi di recupero, aumentando la motivazione e migliorando l'aderenza al percorso terapeutico, anche da casa.

2 Casi d'uso e attori del sistema

I casi d'uso del sistema da implementare sono qui elencati:

- **UC-01 Login & MFA** (Actor: Tutti)
- **UC-02 Gestione PIN** (Actor: Admin, API VR/AR)
- **UC-03 Creazione Piano Terapeutico** (Actor: Doctor)
- **UC-04 Gestione Esercizi** (Actor: Therapist)
- **UC-05 Associazione Esercizi ↔ Piani** (Actor: Doctor/Therapist)
- **UC-06 Upload/Trim Video** (Actor: Therapist)
- **UC-07 Esecuzione & Registrazione Sessione** (Actor: Patient, App VR)
- **UC-08 Dashboard & Report** (Actor: Doctor/Therapist/Admin)
- **UC-09 Integrazione VR: Verify / With-Code / Completed** (Actor: API Client)

Phoenix

2.1 Attori del sistema

- Admin: gestisce il sistema, configura le policy e ha accesso completo.
- Doctor: prescrive piani terapeutici e monitora i pazienti.
- Therapist: definisce e gestisce gli esercizi, collabora con i medici.
- Patient: esegue gli esercizi e fornisce feedback.
- API Client VR/AR: applicazioni esterne che interagiscono tramite API

3 Requisiti Funzionali

Convenzioni utilizzate nei requisiti funzionali (RF):

- MUST: requisito obbligatorio e prioritario.
- SHOULD: requisito raccomandato, ma non strettamente necessario.
- COULD: requisito opzionale, implementabile se possibile.
- CA (Criteri di Accettazione): condizioni che permettono di verificare l'avvenuto soddisfacimento del requisito.
- E (Errori): condizioni di errore e relative risposte del sistema.
- SEC (Sicurezza): regole di sicurezza associate al requisito.
- AUD (Audit): tracciamenti e controlli previsti.

3.1 RF-001 – Gestione Utenti e Autenticazione

Descrizione	Sistema di autenticazione e gestione profili con RBAC e MFA opzionale.
Attori	Admin, Doctor, Therapist, Patient.
Precondizioni	Utente registrato; email verificata.
Flusso principale	1. Login con email/password. 2. MFA opzionale via OTP/email. 3. Emissione sessione o token.
Estensioni	Account bloccato dopo tentativi falliti; reset password via email; logout all devices.
Regole di business	Password policy; session timeout; idle timeout; remember-me 14 giorni.
Dati e validazioni	Nome, cognome, email (RFC 5322), password hash, ruolo; profili estesi per attori.
Sicurezza	HTTPS, HSTS, prevenzione brute-force, rate limit.
Audit	Login/logout, reset password, cambio ruolo.

Criteri di accettazione	Accesso entro 2s con credenziali valide; blocco account dopo 5 tentativi falliti.
Priorità	MUST
Dipendenze	RNF-Sicurezza, RNF-Performance.

3.2 RF-002 – Gestione Piani Terapeutici

Descrizione	Creazione/modifica piani, assegnazioni e versioning.
Attori	Doctor (owner), Therapist (collaboratore lettura/suggerimenti), Patient (lettura).
Precondizioni	Paziente e medico esistenti.
Flusso principale	1. Doctor crea piano (titolo, descrizione, obiettivi, date). 2. Assegna paziente e (opzionale) fisioterapista. 3. Imposta stato (active/completed/suspended). 4. Versioning: ogni modifica crea revision (autore, timestamp, diff
Estensioni	sospensione temporanea; riattivazione; archiviazione a fine percorso; clonazione da template
Regole di business	date coerenti (start ≤ end); un paziente può avere più piani, max N attivi (default 2) configurabile per policy.
Dati e validazioni	title[3..120], description[0..2000], goals[0..1000], status∈{active,completed,suspended}, start_date, end_date, doctor_id, patient_id
Sicurezza	solo Doctor owner e Admin possono modificare; Therapist vede se assegnato; Patient sola lettura.
Audit	creazione, cambio stato, versioni, assegnazioni.
Criteri di accettazione	- Creando un piano con campi minimi validi, esso appare nella dashboard del medico e del paziente entro 1 s dalla persistenza. - Il versioning consente di visualizzare almeno le ultime 20 revisioni con diff.
Priorità	MUST
Dipendenze	RF-001, RF-003.

3.3 RF-003 – Gestione Esercizi

Descrizione	Libreria esercizi definita dai Therapist, riusabile nei piani.

Attori	Therapist (CRUD), Doctor (associa/suggerisce), Patient (legge)
Precondizioni	Therapist autenticato.
Flusso principale	1. Creazione esercizio (nome, descrizione, istruzioni, parametri). 2. Categorizzazione (difficoltà, parte del corpo). 3. Generazione codice univoco a 5 cifre (non sequenziale prevedibile). 4. Attivazione/disattivazione.
Estensioni	versione esercizio (draft→published); duplicazione; localizzazione contenuti IT/EN.
Regole di business	il codice è un identificatore pubblico per VR; collisioni evitate con indice unico + retry random; durata/ripetizioni coerenti (es. durata 1..180 min, reps 0..1000).
Dati e validazioni	name[3..80], description[0..1000], instructions[0..3000], difficulty∈{easy,medium,hard}, body_part enum, duration[min], repetitions, code[\d{5}], is_active, demo JSON.
Sicurezza	solo Therapist crea/modifica i propri; Doctor può proporre parametri nel piano.
Audit	CRUD esercizi, cambi stato
Criteri di accettazione	Alla creazione, il sistema assegna un code a 5 cifre univoco e lo rende ricercabile via API. - Disattivando un esercizio, esso non è più selezionabile per nuovi piani ma resta visibile nei piani storici
Priorità	MUST
Dipendenze	RF-004, RF-007

3.4 RF-004 – Gestione Contenuti Multimediali (Video)

Descrizione	Upload, registrazione da webcam, trimming e associazione ai singoli esercizi.
Attori	Therapist (owner), Doctor (lettura), Patient (lettura).
Precondizioni	esercizio esistente
Flusso principale	upload video (mp4, webm), estrazione metadati (durata, dimensione), creazione thumbnail, trimming lato server
Estensioni	versioni multiple di video per stesso esercizio (lingua/angolazione), sostituzione conservando riferimenti storici.
Regole di business	dimensione max configurabile (default 500MB), durata max 20', bitrate adattivo HLS/DASH; transcodifica asincrona con stato (queued, processing, ready, failed).

Dati e validazioni	title[3..120], description[0..500], path, duration[sec], file_size[bytes], language, captions(vtt).
Sicurezza	accesso controllato via URL firmate (scadenza 15')
Audit	upload, delete, trim, accessi.
Criteri di accettazione	Dopo upload valido, entro T (configurabile) il video diventa streaming-ready e visibile nell'esercizio. - Il trimming produce un nuovo asset senza perdere l'originale.
Priorità	SHOULD
Dipendenze	RNF-Performance/Storage

3.5 RF-005 – Monitoraggio e Tracciamento

Descrizione	Registrazione esercizi completati, parametri di esecuzione, feedback paziente.
Attori	Patient (input), Doctor/Therapist (lettura/analisi).
Precondizioni	piano attivo, esercizi assegnati
Flusso principale	1. Il paziente registra il completamento (manuale o via VR callback). 2. Inserisce (o riceve) durata, ripetizioni, difficoltà percepita (1–10), dolore (0–10), note. 3. Salvataggio di eventuale execution JSON (tracce, performance, biometrici).
Estensioni	validazione automatica di coerenza (durata compatibile con durata esercizio ± tolleranza); allegati (immagini referti futuri); misure derivate (adherence score).
Regole di business	almeno uno tra durata o ripetizioni; campi numerici con range; timestamp in TZ utente.
Dati e validazioni	patient_id, exercise_id, completed_date, duration, repetitions, difficulty_rating(1..10), pain_level(0..10), notes[0..1000], execution JSON.
Sicurezza	solo patient può creare/aggiornare i propri record; medici/fisioterapisti in lettura; Data Minimization per JSON.
Audit	creazione/modifica/eliminazione, origine (web/VR/API), hash anti-manomissione opzionale.
Criteri di accettazione	Un record valido appare in dashboard entro 1 s; - I range invalidi vengono rifiutati con messaggio specifico; - I dati VR vengono salvati come JSON validato contro schema versione vr_exec_v1.
Priorità	MUST

Dipendenze	RF-007

3.6 RF-006 – Sistema PIN Code

Descrizione	PIN a 4 cifre per accesso rapido/app VR, con gestione ciclo di vita.
Attori	Admin (policy), API Client (uso), Patient (uso), Therapist/Doctor (generazione per scopi specifici).
Precondizioni	policy PIN configurata (TTL, single-use, scope).
Flusso principale	generazione PIN (random 0000–9999, esclusi black-list es. 0000, 1234), hashing e salvataggio, stato active; verifica e consumo (used) o scadenza (expired).
Estensioni	PIN scoped allo scopo (es. VR_LOGIN, VR_EXEC), e opzionale binding a user_id o a exercise_code.
Regole di business	rate limit creazione PIN per utente (max 5/h), clean-up automatico scaduti; riutilizzo codice numerico permesso solo se hash differente e stato precedente expired/used da > N giorni (default 7).
Dati e validazioni	code[\d{4}], hash, status∈{active,expired,used}, purpose, generated_at, expires_at, used_at, user_id?, metadata JSON.
Sicurezza	codice mai salvato in chiaro oltre TTL breve (es. 5 min); comparazione tramite hash costante; canali HTTPS; header X-Patient-Code validato.
Audit	generazione, verifiche, consumi, errori. - Errori: 400 codice formato errato; 401 non valido/expired; 409 già usato; 429 rate limit.
Criteri di accettazione	Dato un PIN valido e attivo, la verifica risponde 200 con scope e binding; - Un PIN usato diventa immediatamente used e non più verificabile.
Priorità	MUST
Dipendenze	RF-001, RNF-Sicurezza

3.7 RF-007 – Integrazione VR/AR

Descrizione	API per app VR/AR: verifica codice esercizio, caricamento demo, registrazione completamento
Attori	API Client VR/AR, Patient
Precondizioni	token API o PIN valido; esercizio/piano esistente. - Endpoint (contratti principali): - POST /api/exercises/verify-code → body {

	code: "12345" } → 200 { exercise_id, name, is_active, parameters, version } | 404 | 410 (inactive). - POST /api/exercises/with-code → body { code, exercise: { name, environment, difficulty, tools[], parameters{} } } → 201 con merge/override demo. - POST /api/exercises/completed con header X-Patient-Code: 1234 → body { exercise_id, duration, repetitions, difficulty_rating, pain_level, execution{...} } → 201 | 400 | 401
Flusso principale	
Estensioni	
Regole di business	versionamento schema vr_demo_v1, vr_exec_v1; idempotency via header Idempotency-Key opzionale; clock skew tolleranza ±60s.
Dati e validazioni	
Sicurezza	autenticazione Bearer Token con scopes (vr:read, vr:write), oppure PIN per low-friction; CORS limitato a origin note.
Audit	ogni chiamata con request-id, timestamp, actor; mascheramento PII nei log
Criteri di accettazione	- verify-code risponde < 500 ms per codice valido. - completed valida schema e persiste un CompletedExercise associato al paziente (via PIN) in < 1 s.
Priorità	MUST
Dipendenze	RF-003, RF-006, RF-005

3.8 RF-008 – Dashboard e Reporting

Descrizione	Dashboard per ruolo e reportistica
Attori	Doctor, Therapist, Admin, Patient (limitata).
Precondizionii	
Contenuti minimi per ruolo	- Doctor: pazienti attivi, aderenza (%) per piano, trend dolore/difficoltà, alert regressi. - Therapist: esercizi più/meno completati, durata media, performance; code coverage libreria esercizi. - Admin: utenti attivi, errori, tempi risposta, storage, job multimediali. - Patient: progressi personali, badge motivazionali opzionali. - Report: esportazione CSV/JSON (UTF-8), filtri per date/ruolo/paziente.

Estensioni	
Regole di business	
Dati e validazioni	
Sicurezza	visibilità dati perminata (no cross-patient)
Audit	export tracciato con motivo e scopo
Criteri di accettazione	Ogni dashboard carica i widget principali < 2 s con dataset ≤ 10k record aggregati. - Export genera file entro 30 s per dataset ≤ 1M righe (job asincrono con notifica).
Priorità	SHOULD
Dipendenze	RF-005, RNF-Performance.

3.9 RF-009 – API e Integrazioni

Descrizione	API RESTful complete con token e permessi granulari, doc interattiva, webhook.
Attori	
Precondizioni	
Flusso	Token: Bearer con scopes (api:read, api:write, vr:*), TTL configurabile, revoca immediata. - Doc: Swagger/OpenAPI 3.0 servita con UI interattiva; esempi cURL/JS/PHP/Python. - Webhook: eventi exercise.completed, video.ready, pin.expired; firma HMAC-SHA256, retry con backoff. - Rate limiting: default 60 req/min (generiche), 30 req/min (critiche), 10 req/min (bulk); header X-RateLimit-*. - Errori: standard RFC 7807 (problema JSON) con type, title, detail, status, instance.
Sicurezza	visibilità dati perminata (no cross-patient)
Audit	export tracciato con motivo e scopo
Criteri di accettazione	Doc disponibile e coerente con implementazione (lint + test contrattuali). - Webhook inviano richiesta firmata entro 5 s dall'evento, con 3 retry
Priorità	MUST
Dipendenze	RF-001, RNF-Sicurezza

4 Requisiti Non Funzionali

Phoenix

4.1 RNF-001 – Performance & Scalabilità

- Tempi risposta:
  - 95° percentile < 2 s per operazioni standard;
  - 99° percentile < 5 s.
- Throughput: ≥ 100 utenti concorrenti; testati con scenari: login, dashboard, API VR completed burst 50 rps per 60 s.
- Scalabilità orizzontale: stateless API + cache Redis; job queue separata per video.
- Caching: HTTP cache per GET, Redis per liste, ETag/If-None-Match.
- Verifica: test Gatling/K6 ambienti staging, SLO monitorati.

4.2 RNF-002 – Sicurezza

- Transport: TLS 1.2+; HSTS; TLS modern cipher; CSP default-src 'self' (whitelist media cdn).
- Storage: AES-256-at-rest per dati sensibili; segreti in vault.
- MFA: OTP email/app (TOTP).
- RBAC: enforcement a livello middleware e query.
- Hardening: prevenzione XSS, CSRF, SQLi, SSRF; limit upload MIME; antivirus opzionale su media.
- Pentest: almeno 1/anno; SAST/DAST in CI.
- Verifica: check OWASP ASVS L2.

4.3 RNF-003 – Usabilità & Accessibilità

- UI: responsive (Bootstrap 5), testo ≥ 16px, contrasto WCAG, navigazione tastiera.
- Localizzazione: IT pronto, EN opzionale (i18n).
- Mobile: layout ottimizzato, touch target ≥ 44px.
- Verifica: audit Lighthouse accessibilità ≥ 90.

4.4 RNF-004 – Affidabilità & Continuità Operativa

- Disponibilità: >= 99.5% / mese (esclusa manutenzione programmata).
- Backup: DB full giornaliero + PITR; retention 30 giorni; restore test trimestrale.
- DR: RTO < 4h, RPO < 1h; runbook documentato.
- Monitoring: metriche app (APM), log centralizzati, alert SLO breach.

4.5 RNF-005 – Manutenibilità & Qualità

- Architettura: MVC, SOLID, servizi modulari.
- Test: coverage ≥ 80% (unit, integration, e2e API); contract test; smoke test post-deploy.
- CI/CD: lint, SCA, migrazioni DB safe, feature flags, canary opzionale.
- Documentazione: docstring, ADR per decisioni architetturali.

Phoenix

4.6 RNF-006 – Compatibilità & Portabilità

- Browser: ultime 2 versioni di Chrome, Firefox, Safari, Edge.
- API: REST con JSON UTF-8; tolleranza a campi extra; versionamento /v1.

4.7 RNF-007 – Privacy & Compliance (GDPR)

- Base giuridica & consenso: tracciato per-paziente; banner/consenso per dati sensibili.
- Diritti interessato: export, rettifica, cancellazione con SLA 30gg; soft-delete + purge.
- Data minimization: raccolta solo dati necessari; pseudonimizzazione dove possibile.
- Registro trattamenti & DPIA light: aggiornati.

5 Modello Dati

Per ogni entità sono indicati: attributi, tipi, constraint, indici.

5.1 User

- **Attributi:** `id PK`, `name`, `surname`, `email UNIQUE`, `password_hash`, `role_id FK`, `mfa_enabled bool`, `locale`, `created_at`, `updated_at`.
- **Constraint:** email valida e unica; password non reversibile.
- **Indici:** `email`, `role_id`.

5.2 Role & Permission

- **Role:** `id`, `name ∈ {admin, doctor, therapist, patient}`, `description`.
- **Permission:** tabella `role_permission` (molti-a-molti), es. `api:create`, `patient:read:own`.

5.3 Patient / Doctor / Therapist (profili)

- **Patient:** `user_id FK`, `dob`, `gender enum`, `phone`, `address`, `medical_notes encrypted`.
- **Doctor:** `user_id FK`, `specialization`, `license_number`, `department`.
- **Therapist:** `user_id FK`, `specialization`, `license_number`, `experience_years`.

5.4 7.4 TreatmentPlan

- `id PK`, `patient_id FK`, `doctor_id FK`, `title`, `description`, `goals`, `start_date`, `end_date`, `status enum`, timestamps.

- **Indices composti:** `(patient_id, status)`, `(doctor_id, status)`.
- **Revisioning:** tabella `treatment_plan_revisions` con `diff JSON`.

5.5 Exercise

- `id` PK, `therapist_id` FK, `name`, `description`, `instructions`, `difficulty` enum, `body_part` enum, `duration`, `repetitions`, `code` UNIQUE, `is_active`, `demo` JSON.

5.6 Video

- `id` PK, `exercise_id` FK, `title`, `description`, `path`, `duration`, `file_size`, `language`, `captions_path`, `status` enum.

5.7 CompletedExercise

- `id` PK, `patient_id` FK, `exercise_id` FK, `completed_date`, `duration`, `repetitions`, `difficulty_rating`, `pain_level`, `notes`, `execution` JSON (schemato e versionato), `source` enum{web,vr,api}.

5.8 PinCode

- `id` PK, `user_id` FK?, `code` transient, `hash`, `status`, `purpose`, `generated_at`, `expires_at`, `used_at`, `metadata` JSON.

6 Contratti API

Formato errori: RFC7807 Problem+JSON; **Auth:** Bearer o PIN.

6.1 POST /api/exercises/verify-code

Request { code: string(5) }
Response 200 { exercise_id:int, name:string, is_active:boolean, parameters:{...}, version:"v1" }
Errori 404 Not Found, 410 Gone se disattivo, 422 validazione.

6.2 POST /api/exercises/with-code

Request { code, exercise:{ name, environment, difficulty, tools[], parameters{} } }
Response 201 { id, code, demo:{...}, status:"merged" }
Errori 401/403, 422.

6.3 POST /api/exercises/completed

Header X-Patient-Code: ####
Request { exercise_id, duration?, repetitions?, difficulty_rating 1..10, pain_level 0..10, execution{...} }
Response 201 { completed_exercise_id, at }
Errori 400, 401, 409, 422.

6.4 PIN

- `POST /api/pin-codes/verify` → `{ code }` → 200 `{ purpose, user_id?, scope, expires_at }`
- `POST /api/pin-codes/info` → `{ code }` → 200 `{ status, purpose }`
- `POST /api/pin-codes/purpose` → `{ code, purpose }` → 200/422

7 Sicurezza & Privacy

7.1 Threat Model (STRIDE – sintesi)

- **Spoofing:** MFA, pin hashing, token scopi limitati.
- **Tampering:** firma webhook, checksum asset, audit immutabile.
- **Repudiation:** log con request-id e orario sincronizzato (NTP).
- **Information Disclosure:** CIP data minimization, cifratura, masking log.
- **DoS:** rate limiting, circuit breaker, WAF.
- **Elevation of Privilege:** RBAC + least privilege, review permessi trimestrale.

7.2 GDPR – processi

- **Consenso:** modale dedicata, registrazione evento (`consent_log`).
- **Access/Export:** JSON/CSV per singolo paziente entro 30 gg.
- **Rettifica/Cancellazione:** workflow con soft-delete + purge schedulato.
- **Data Retention:** policy per entità (es. `CompletedExercise` 10 anni salvo diverse disposizioni; configurabile per centro).

8 Requisiti tecnologici

Il sistema in specifica dovrà essere realizzato con riferimento al seguente stack tecnologico:

- Backend: Laravel 11 (PHP ≥8.2)

- Database: MySQL 8 / MariaDB

- Cache: Redis / Memcached

- Containerizzazione: Docker

- Storage: S3 compatibile per video

- Sicurezza: HTTPS, storage cifrato, MFA opzionale

9 Specifiche integrazione Agente AI e Sistema Gestionale RehabSphere

Phoenix

Lo scopo di questa sezione è definire le specifiche di integrazione tra Agente AI (AI Gateway) e il Gestionale RehabSphere.

L'integrazione tra AI Gateway (che orchestra ASR/NLP/RAG/LLM) e Sistema Gestionale RehabSphere adotta un'architettura API-first basata su canale REST JSON su HTTPS per tutte le operazioni sincrone e deterministiche (lettura dati, validazioni, registrazioni), con un canale WebSocket opzionale per il dialogo in tempo reale con l'app VR. Per gli eventi asincroni che richiedono propagazione tra domini applicativi (ad es. aggiornamenti piani, completamento esercizi), sono previsti webhook firmati. L'obiettivo è garantire bassa latenza in sessione, robustezza agli errori e un modello di sicurezza coerente con GDPR e principi di minimizzazione.

Il canale di comunicazione è REST JSON/HTTPS per:

- recuperare il contesto minimo del paziente e dell'esercizio (read-only, idempotente);
- validare codici/pin di sessione ed esercizi;
- registrare eventi transazionali (es. exercise.completed).

Le richieste sono stateless e versionate (es. /api/v1/...), con risposta in formato Problem+JSON in caso di errore e header di correlazione (X-Request-Id) per l'audit.

WebSocket è un canale opzionale e si attiva per i soli casi d'uso in cui la conversazione AI deve scambiare messaggi a bassa latenza con l'app VR (testo parziale, segnali per lip-sync, eventi "typing/streaming"). Il canale è autenticato all'instaurazione della sessione e rimane effimero (chiusura automatica al termine della sessione o per inattività). Il contenuto veicolato è non persistente nel gestionale: la persistenza degli esiti clinici rimane sul canale REST.

Per l'autenticazione e l'autorizzazione si distinguono i due domini:

- Lato Gestionale: le API sono protette con OAuth2 (client-credentials o authorization-code secondo caso d'uso) e JWT come bearer token. Gli scope (es. patients.read, plans.read, exercises.write) applicano il least-privilege. I JWT includono claim minimi (issuer, audience, scadenza) e sono soggetti a rotazione e revoca. Le chiamate provenienti dall'AI Gateway verso il gestionale usano mTLS opzionale o IP allow-listing quando richiesto dal contesto di deploy.
- Lato AI Gateway: l'accesso da parte dell'app VR e dei servizi interni avviene tramite token applicativo (ad es. API key o JWT emesso dal gateway) con TTL breve; il token è richiesto sia per il REST sia per l'upgrade WebSocket. Il gateway applica rate-limit, WAF e quote per mitigare abusi e denial-of-service.

Alcuni cambi di stato devono essere propagati asincronamente tra i domini per mantenere l'allineamento del contesto:

- exercise.completed: il gestionale, a fronte di una registrazione REST andata a buon fine, emette un webhook (opzionale) verso l'AI Gateway per aggiornare cache/embedding o attivare post-processing.

- plan.updated / assignment.changed: alla modifica di piano o assegnazioni, il gestionale notifica l'AI Gateway così che le risposte in sessione riflettano il nuovo perimetro.

I webhook sono firmati (es. HMAC su payload + timestamp) e includono event_id e attempt per consentire idempotenza e retry esponenziale lato mittente (fino a N tentativi). Il ricevente deve rispondere 2xx con tempi brevi; in caso di fallimento persistente si applica un dead-letter o un meccanismo di parking per l'analisi. Gli endpoint di webhook sono isolati, con verifica della firma, validazione schema del payload e back-pressure.

9.1 Contratti API del Gestionale

Le API esposte dal Gestionale saranno consumate dall'AI Gateway per arricchire il contesto del paziente e dell'esercizio.

Endpoint	Metodo	Uso AI Gateway
/api/pazienti/{id}	GET	Recupero anagrafica e profilo clinico minimo
/api/pazienti/{id}/piani	GET	Recupero piani/playlist esercizi
/api/exercises/verify-code	POST	Validazione codice esercizio in sessione VR
/api/exercises/completed	POST	Registrazione completamento esercizio (header X-Patient-Code)

9.2 Mapping Dati

Concetto	Origine / Note
Paziente (id, ruolo, limiti)	GET /api/pazienti/{id}
Piano/Esercizi assegnati	GET /api/pazienti/{id}/piani
Esercizio corrente	POST /api/exercises/verify-code
Completamento esercizio	POST /api/exercises/completed
PIN/Scope sessione	Endpoint PIN (verify/info/purpose)

9.3 Sequence Principali

Q&A in sessione

1. App VR invia domanda + contesto (codice esercizio / id paziente) → AI Gateway.
2. AI Gateway valida PIN/codice ed estrae dati minimali dal Gestionale.
3. AI Gateway interroga Vector DB e costruisce prompt RAG → LLM.

4. Risposta restituita all'App VR (testo + segnali lip-sync); logging/audit.

Registrazione completamento esercizio

5. App VR invia POST /api/exercises/completed con X-Patient-Code.
6. Gestionale valida codice/utente e persiste CompletedExercise.
7. Webhook (opzionale) verso AI Gateway per aggiornare il contesto/embedding.

9.4 Errori & Retry

- Formato errori: RFC7807 Problem+JSON; codici 4xx/5xx standard.
- Rate-limit e circuit breaker su AI Gateway; retry esponenziali su webhook (3 tentativi).
- Fallback risposta sicura dell'avatar quando mancano dati o il consenso è assente.

9.5 Sicurezza, Privacy e Compliance

- TLS 1.2+; HSTS; OAuth2/JWT; RBAC; principle of least privilege.
- Cifratura at-rest (AES-256); segreti in vault; audit immutabile con request-id.