UNIVERSITÀ
DI PAVIA

FACOLTÀ DI INGEGNERIA

DIPARTIMENTO DI INGEGNERIA INDUSTRIALE E DELL'INFORMAZIONE

CORSO DI LAUREA MAGISTRALE IN BIOINGEGNERIA

TESI DI LAUREA

SVILUPPO DI UN SISTEMA AUTOMATIZZATO PER LA GESTIONE DEL FLUSSO DI DATI DA OMOP A REDCAP

Candidato: Eleonora Geraci

Relatore: Prof.ssa Cristiana Larizza
Correlatori: Matteo Gabetta, Giorgia Petrini

A.A. 2024/2025

Indice

Capitolo 1

Introduzione

La gestione e l'integrazione dei dati clinici costituiscono ad oggi due sfide centrali nell'ambito dei sistemi informativi sanitari, soprattutto in contesti caratterizzati da architetture distribuite e dall'adozione di standard di interoperabilità. La crescente digitalizzazione della sanità ha infatti portato alla produzione e alla disponibilità di grandi volumi di dati eterogenei, provenienti da sistemi differenti e spesso non pienamente compatibili tra loro. In questo scenario, la capacità di garantire uno scambio efficiente, coerente e semanticamente corretto delle informazioni assume un ruolo chiave sia per il supporto all'attività clinica sia per lo sviluppo della ricerca. Uno degli aspetti critici riguarda la necessità di ridurre le operazioni manuali di inserimento e trasferimento dei dati, che risultano non solo dispendiose in termini di tempo, ma anche soggette a errori e incoerenze. Diventa quindi fondamentale progettare soluzioni in grado di automatizzare i processi di integrazione, mantenendo al contempo elevati standard di qualità, tracciabilità e riusabilità del dato.

In questo contesto si inserisce il presente lavoro di tesi, che ha come obiettivo la progettazione e lo sviluppo di un framework software per il trasferimento automatico di dati clinici da un database conforme al modello OMOP Common Data Model a un progetto REDCap. Il sistema è concepito per supportare studi osservazionali e attività di ricerca clinica. Il framework implementa un processo automatizzato di estrazione, trasformazione e caricamento dei dati clinici, guidato da configurazioni flessibili che consentono di adattare il comportamento del sistema a diversi contesti applicativi. In questo modo, i dati vengono selezionati, rielaborati e resi compatibili con il modello di destinazione, garantendo coerenza tra le informazioni e facilitandone il riutilizzo. Un elemento distintivo dell'approccio adottato è la separazione delle responsabilità tra i diversi componenti del sistema, che consente di isolare la

logica di accesso ai dati, le regole di trasformazione e le modalità di interazione con i sistemi esterni. Questa scelta progettuale rende il framework modulare, estendibile e facilmente riutilizzabile in scenari differenti.

Questo lavoro di tesi evidenzia come il valore dei dati clinici non risieda esclusivamente nella loro quantità, ma soprattutto nella capacità di integrarli, interpretarli e riutilizzarli in modo consapevole. In tale prospettiva, soluzioni come quella proposta rappresentano un passo significativo verso una gestione più efficace e affidabile dell'informazione sanitaria destinata alla ricerca.

Capitolo 2

Contesto

2.1 Evoluzione digitale dell'ecosistema informativo sanitario

Negli ultimi anni, il settore sanitario ha attraversato una trasformazione strutturale dovuta alla progressiva digitalizzazione dei processi clinici, amministrativi e di ricerca. Il concetto di Digital Health non si limita all'introduzione di strumenti informatici nei contesti ospedalieri, ma descrive un cambiamento sistemico nell'organizzazione, nella gestione e nella valorizzazione del patrimonio informativo sanitario.

La sanità contemporanea si basa su un ecosistema informativo complesso, costituito da sistemi eterogenei che operano su livelli differenti e che devono cooperare tra loro per garantire la continuità assistenziale, la qualità del dato e l'interoperabilità tra le diverse piattaforme tecnologiche. All'interno di tale struttura convivono sistemi informativi ospedalieri, di laboratorio e radiologici, cartelle cliniche elettroniche, archivi documentali, data warehouse clinici e piattaforme dedicate alla ricerca.

Ogni sistema è concepito e progettato per soddisfare specifiche esigenze operative e per supportare determinati processi clinici o amministrativi. Tuttavia, tale specializzazione ha storicamente condotto a una segmentazione strutturale dei dati. Nel tempo, infatti, sono stati sviluppati modelli dati differenti, terminologie non uniformi e architetture tecnologiche disomogenee, rendendo complessa l'integrazione e la condivisione delle informazioni tra sistemi distinti. Questa frammentazione rappresenta uno dei principali ostacoli alla piena valorizzazione dei dati sanitari, in

particolare quando si intende riutilizzarli per finalità di ricerca o per analisi secondarie su larga scala. In questo contesto, l'interoperabilità non costituisce soltanto una caratteristica tecnica desiderabile, ma un requisito fondamentale per garantire l'integrità, la coerenza semantica, l'affidabilità e la riusabilità delle informazioni cliniche.

Come illustrato in Figura 2.1, l'evoluzione dei sistemi sanitari segue un percorso progressivo che porta da modelli tradizionali, caratterizzati da processi frammentati e cartacei, verso ecosistemi digitali integrati e orientati al paziente. La trasformazione coinvolge simultaneamente ambiti amministrativi, clinici, infrastrutturali e relazionali, evidenziando come la digitalizzazione non sia un cambiamento puntuale ma un processo sistemico e multidimensionale. In particolare, emerge il ruolo centrale dell'integrazione tra sistemi e della collaborazione tra attori, elementi chiave per la realizzazione di uno Smart Healthcare System.

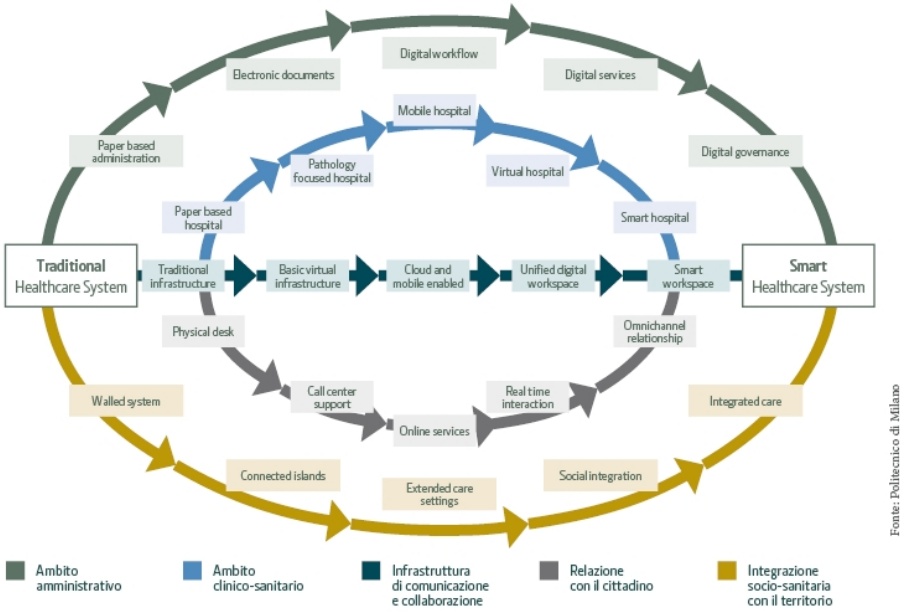


Figura 2.1: Evoluzione del sistema sanitario verso il modello digitale.
Fonte: Osservatorio ICT in Sanità, Politecnico di Milano

2.2 Dato sanitario e Big Data

Il dato sanitario rappresenta la registrazione digitale di un evento associato al percorso di cura di un paziente e comprende elementi di natura clinica, diagnostica, terapeutica e amministrativa. A questo si affiancano contenuti generati da disposi-

tivi medici, sistemi di monitoraggio continuo e piattaforme digitali di supporto alla cura.

Con la progressiva digitalizzazione dei processi assistenziali, tali informazioni vengono registrate e conservate in formato elettronico, accumulandosi nel tempo e dando origine ai cosiddetti Big Data. In ambito sanitario, i Big Data non si distinguono esclusivamente per l'elevato volume, ma anche per la pluralità e varietà delle fonti e per l'eterogeneità dei formati. Questi sono comunemente descritti attraverso le tre dimensioni fondamentali di volume, varietà e velocità, alle quali vengono affiancati ulteriori aspetti quali veridicità e valore, particolarmente rilevanti in ambito clinico.

I dati possono essere strutturati, semi-strutturati o non strutturati. Appartengono alla prima categoria dati organizzati secondo schemi relazionali e codifiche standard come i risultati di laboratorio o delle prescrizioni farmacologiche e all'ultima i referti testuali, le immagini diagnostiche e le annotazioni cliniche. Questa eterogeneità rende complessa l'integrazione e l'elaborazione automatizzata, in particolar modo quando le diverse piattaforme implementano modelli logici e strutture dati non compatibili tra loro.

Nonostante tali complessità, i Big Data sanitari rappresentano una risorsa strategica per la ricerca clinica e per lo sviluppo di modelli decisionali basati sull'analisi quantitativa. L'elaborazione su larga scala consente di valutare outcome terapeutici, individuare correlazioni tra parametri clinici e fattori di rischio, monitorare l'andamento temporale delle patologie, supportare la definizione di strategie di prevenzione e contribuire allo sviluppo di modelli predittivi a supporto delle decisioni sanitarie.

Affinché tale potenziale possa essere concretamente sfruttato è necessario che il patrimonio informativo sia organizzato secondo modelli strutturati e interoperabili, in grado di garantire coerenza semantica, uniformità di rappresentazione e integrazione tra piattaforme differenti. L'assenza di schemi condivisi limita la possibilità di aggregare e confrontare informazioni provenienti da contesti diversi, ostacolando l'automatizzazione dei processi di estrazione, trasformazione e trasferimento. La standardizzazione del modello dati diventa quindi un requisito fondamentale per costruire basi solide e affidabili a supporto della ricerca scientifica.

2.3 Uso primario e uso secondario del dato

Nel contesto sanitario è fondamentale fare una distinzione tra uso primario e uso secondario del dato. L'uso primario riguarda l'impiego diretto delle registrazioni cliniche per finalità assistenziali, quali diagnosi, prescrizione terapeutica, monitoraggio del paziente e gestione amministrativa delle procedure di cura. In questa fase, il dato è funzionale al singolo percorso clinico ed è generalmente memorizzato all'interno del sistema informativo locale.

L'uso secondario, invece, si riferisce al riutilizzo delle informazioni raccolte durante l'attività assistenziale per finalità diverse, come ricerca clinica, studi osservazionali, analisi epidemiologiche e sviluppo di modelli predittivi. Tale riutilizzo richiede requisiti aggiuntivi rispetto all'uso primario: maggiore strutturazione, coerenza semantica, tracciabilità delle trasformazioni e capacità di integrazione tra fonti differenti.

In assenza di linee guida condivise e di modelli di standardizzazione formalizzati, le registrazioni rimangono confinate all'interno dei singoli sistemi sorgente, ostacolando l'integrazione trasversale e limitando la possibilità di generare conoscenza su scala più ampia. La definizione e l'adozione di meccanismi strutturati che rendano possibile l'uso secondario dei dati rappresentano pertanto una delle sfide centrali nello sviluppo della sanità digitale contemporanea.

2.4 Criticità

L'impiego strategico dei dati sanitari è condizionato da diverse criticità che possono essere ricondotte principalmente alla qualità del dato, all'eterogeneità strutturale e alle modalità di acquisizione.

Un primo elemento critico riguarda la qualità e l'affidabilità dei dati. Errori di codifica, incompletezza delle registrazioni e differenze nelle modalità di raccolta possono compromettere il confronto tra dataset provenienti da fonti diverse. In ambito clinico e di ricerca, tali problematiche incidono direttamente sulla validità delle analisi, sulla robustezza dei risultati e sulla riproducibilità degli studi.

Un ulteriore aspetto rilevante riguarda l'eterogeneità strutturale dei dati sanitari, che deriva dalla presenza di formati, modelli e sistemi di codifica differenti tra le diverse piattaforme e organizzazioni. I dati possono essere archiviati secondo

standard non uniformi, utilizzare terminologie diverse o presentare livelli di strutturazione variabili (dati strutturati, semi-strutturati o testuali). Questa mancanza di armonizzazione, in particolare nei sistemi di codifica clinica, rende complessa l'integrazione e l'interoperabilità tra sistemi informativi sanitari, ostacolando lo scambio e l'analisi congiunta delle informazioni.

Un'altra criticità riguarda le modalità di acquisizione delle informazioni cliniche. In molti contesti operativi, una parte significativa delle registrazioni viene inserita manualmente dagli operatori sanitari. Tale modalità espone al rischio di errori di digitazione, omissioni, incoerenze sintattiche e utilizzo non uniforme delle codifiche.

Queste imprecisioni possono propagarsi nei processi di integrazione e analisi, compromettendo la qualità complessiva del dataset e riducendo l'affidabilità delle evidenze prodotte. La gestione e il controllo della qualità del dato rappresentano quindi un elemento centrale nei sistemi orientati alla valorizzazione dei Big Data sanitari.

2.5 Processi di integrazione e pipeline ETL

Per rendere possibile l'uso secondario dei dati raccolti è necessario implementare processi strutturati di integrazione. In ambito informatico tali processi sono comunemente ricondotti alle pipeline di Extract, Transform, Load (ETL), che consentono di estrarre i dati dalle sorgenti originarie, trasformarli secondo uno schema comune e caricarli all'interno di un modello dati condiviso.

La fase di estrazione prevede l'acquisizione delle registrazioni dai sistemi sorgente; la trasformazione implica la normalizzazione dei formati, l'armonizzazione semantica e la mappatura verso codifiche standard; infine, il caricamento prevede l'organizzazione dei dati raccolti all'interno di una struttura coerente e interrogabile.

L'adozione di pipeline ETL robuste è essenziale per ridurre la frammentazione, migliorare la qualità del dato e abilitare l'automazione dei flussi informativi tra piattaforme differenti. In assenza di tali meccanismi, l'integrazione tra sistemi rimane parziale e spesso affidata a procedure manuali o a esportazioni non standardizzate, con conseguente aumento del rischio di errore e perdita di consistenza.

Come illustrato in Figura 2.2, una pipeline ETL si configura come un processo sequenziale che collega diverse sorgenti eterogenee, come database, file e API, a sistemi di destinazione centralizzati, attraverso una fase intermedia di trasformazione. La presenza di una staging area e di data warehouse, locali o in cloud, evidenzia

il ruolo chiave dell'organizzazione e della standardizzazione del dato nel garantire interoperabilità e scalabilità dei sistemi informativi.

La progettazione di soluzioni tecnologiche orientate all'automazione e alla standardizzazione dei processi di trasferimento rappresenta quindi un passaggio chiave per supportare in modo efficace le attività di ricerca basate sui Big Data sanitari. In questo scenario si inserisce il presente lavoro di tesi, che affronta il problema della standardizzazione e integrazione dei dati clinici al fine di renderli interoperabili, strutturati e riutilizzabili per finalità di uso secondario, attraverso l'adozione di modelli condivisi e processi di trasformazione automatizzati.

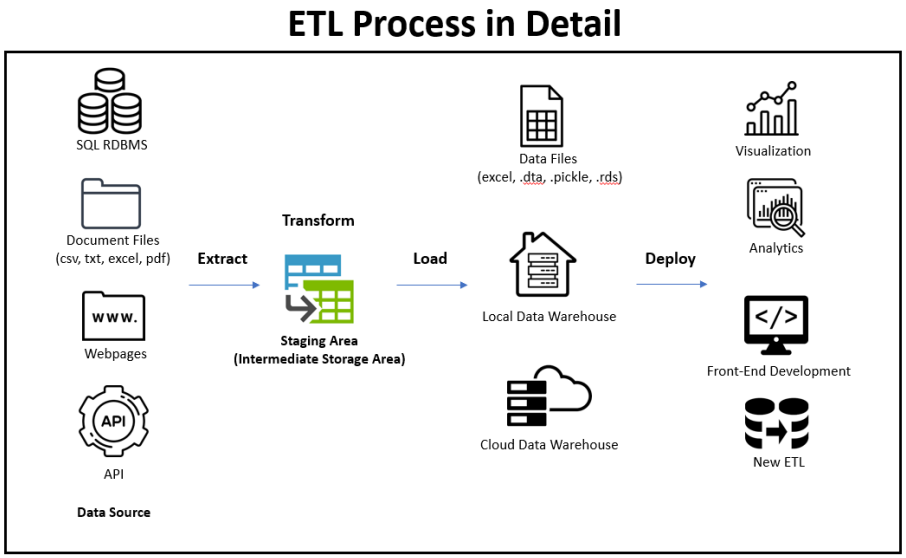


Figura 2.2: Pipeline ETL per l'integrazione e l'analisi dei dati

S

Capitolo 3

Tecnologie utilizzate

Alla luce delle criticità descritte nel capitolo precedente, emerge con chiarezza come l'uso secondario dei dati sanitari richieda l'adozione di strumenti tecnologici in grado di garantire standardizzazione, interoperabilità e integrazione strutturata delle informazioni provenienti da sistemi diversi.

L'eterogeneità dei modelli dati, la frammentazione delle fonti informative e la necessità di garantire coerenza e confrontabilità tra dataset differenti rendono indispensabile l'utilizzo di architetture e framework condivisi, capaci di normalizzare la rappresentazione del dato clinico e di supportarne l'elaborazione su larga scala.

In questo contesto si inseriscono le tecnologie adottate nel presente lavoro di tesi, che comprendono modelli dati standardizzati, strumenti per l'acquisizione e lo scambio di informazioni cliniche, framework di sviluppo software e soluzioni per la serializzazione e gestione dei dati. Il presente capitolo descrive le principali tecnologie impiegate, illustrandone le caratteristiche architetturali e il ruolo all'interno del sistema progettato.

3.1 Il modello dati OMOP

Tra i modelli dati standardizzati maggiormente diffusi per la gestione e l'analisi di dati sanitari a fini di ricerca figura l'OMOP Common Data Model (CDM), sviluppato originariamente nell'ambito del progetto Observational Medical Outcomes Partnership (OMOP), avviato nel 2008 sotto il coordinamento della Food and Drug Administration (FDA) statunitense. L'obiettivo iniziale del progetto era valutare in modo sistematico la sicurezza dei farmaci attraverso l'analisi di grandi database. Dal 2014, il progetto è gestito da una comunità internazionale chiamata Observational Health Data Sciences and Informatics (OHDSI), che ne ha esteso l'ambito applicativo trasformando OMOP in un'infrastruttura open-source per la ricerca collaborativa su dati sanitari osservazionali. Il modello dati e gli strumenti associati sono liberamente accessibili e mantenuti attraverso un processo di sviluppo comunitario che coinvolge istituzioni accademiche, enti di ricerca, industrie farmaceutiche e organizzazioni sanitarie a livello globale.

L'OMOP CDM è attualmente disponibile nella versione 5.4, evoluzione consolidata della serie 5.x. Il modello è progettato per accogliere dati provenienti da fonti eterogenee, quali cartelle cliniche elettroniche, database amministrativi, registri clinici e dati assicurativi, armonizzandoli in una struttura standardizzata che ne consente l'analisi.

Il principio cardine del modello è la trasformazione dei dati sorgente in uno schema relazionale comune, organizzato in tabelle tematiche che rappresentano eventi clinici e informazioni anagrafiche. Ogni evento è associato a un identificativo concettuale standard, che garantisce coerenza semantica indipendentemente dal sistema di codifica originario. Dal punto di vista architetturale, OMOP è implementabile su sistemi di gestione di database tradizionali, come PostgreSQL, Oracle o SQL Server.

Questa logica di integrazione è ben sintetizzata in Figura 3.1, dove diverse sorgenti eterogenee vengono ricondotte a un modello comune attraverso un processo di trasformazione intermedio. In tal modo diventa possibile applicare metodi analitici uniformi su dati originariamente disomogenei, favorendo la comparabilità dei risultati e l'esecuzione di studi su larga scala.

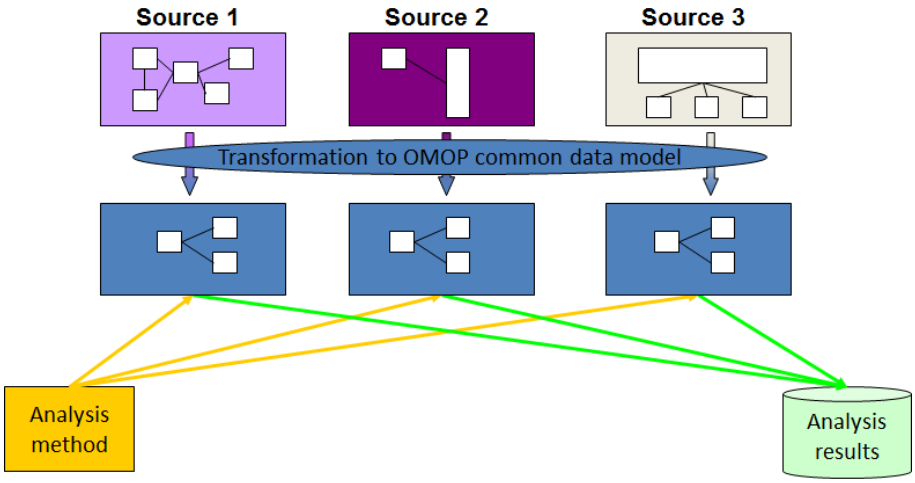


Figura 3.1: Integrazione di sorgenti dati eterogenee nel modello OMOP Common Data Model (CDM).
Fonte: https://www.ohdsi.org/ (The Book of OHDSI, 2019)

Nel presente lavoro, OMOP rappresenta il modello di riferimento per l'armonizzazione dei dati clinici e costituisce la base informativa su cui vengono implementate le procedure di estrazione e trasferimento dati.

3.1.1 Struttura dello schema OMOP CDM

Architettura generale dello schema relazionale

Lo schema del CDM è organizzato secondo una struttura relazionale normalizzata, progettata per rappresentare in modo coerente e uniforme eventi clinici, informazioni anagrafiche e metadati. Nella versione 5.4, attualmente di riferimento, il modello comprende oltre quaranta tabelle relazionali, articolate in domini funzionali distinti che separano i dati clinici dalle componenti semantiche e dai metadati di sistema, secondo una modellazione orientata alla standardizzazione e alla confrontabilità dei dati, come rappresentato in Figura 3.2.

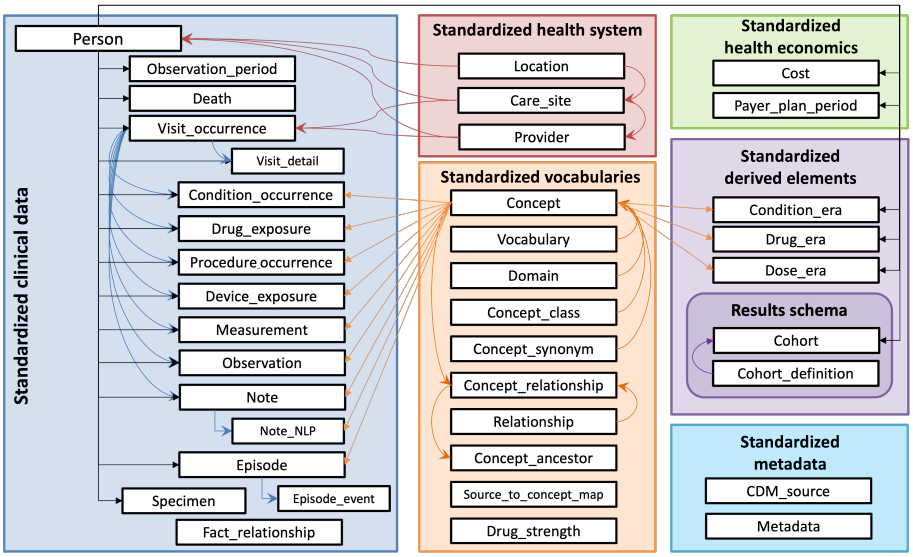


Figura 3.2: Tabelle del OMOP CDM
Fonte: https://www.ohdsi.org/ (The Book of OHDSI, 2019)

Logica event-based e struttura delle tabelle cliniche

Le tabelle cliniche costituiscono il nucleo operativo del modello e rappresentano gli eventi che descrivono il percorso assistenziale del paziente secondo una logica event-based, nella quale ogni evento clinico è registrato come record indipendente. Il punto di ingresso è la tabella person, che contiene un record per ciascun individuo e funge da entità centrale alla quale sono collegate, tramite chiavi esterne, tutte le informazioni relative a diagnosi, esposizioni farmacologiche, procedure, misurazioni, osservazioni e visite sanitarie. La rappresentazione degli eventi clinici avviene in tabelle tematiche distinte, ciascuna dedicata a una specifica categoria di informazione. Tali tabelle condividono una struttura omogenea, caratterizzata dalla presenza di un identificativo univoco dell'evento, dal riferimento al soggetto, dall'associazione obbligatoria a un concetto standard e da attributi temporali standardizzati che descrivono l'inizio e, se prevista, la fine dell'evento. Questa uniformità strutturale favorisce l'esecuzione di analisi longitudinali coerenti tra domini differenti e consente la ricostruzione della storia clinica del paziente nel tempo.

Il ruolo centrale dei concetti standard è ulteriormente evidenziato nella struttura della tabella CONCEPT, la cui organizzazione è riportata in Figura 3.3, che raccoglie le informazioni semantiche necessarie per garantire l'interpretazione univoca dei dati all'interno del modello.

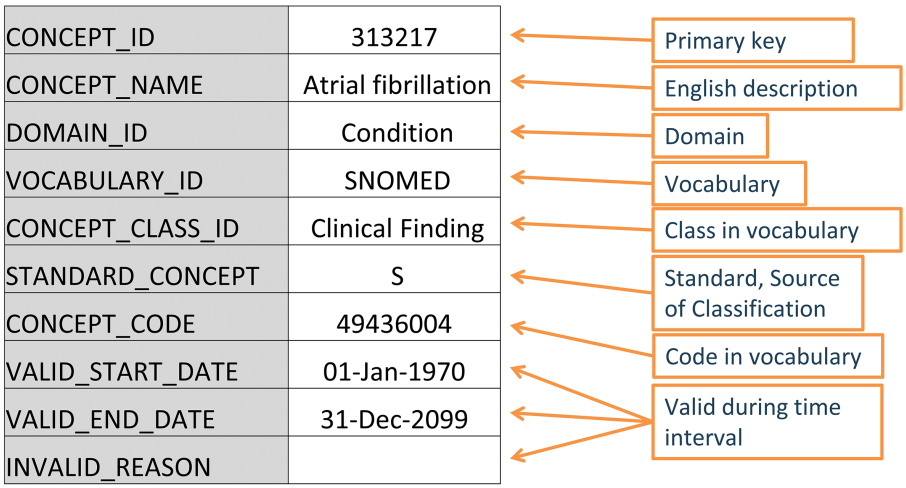


Figura 3.3: Struttura della tabella CONCEPT nel modello OMOP CDM
Fonte: https://www.ohdsi.org/ (The Book of OHDSI, 2019)

Infrastruttura semantica e vocabolari standard

Un elemento architetturale fondamentale del CDM è l'associazione obbligatoria di ogni evento clinico a un identificativo concettuale standard (concept_id). Questo identificativo rimanda alla tabella concept, appartenente alla sezione dei vocabolari standardizzati, che costituisce il nucleo semantico del modello. La tabella dei concetti contiene milioni di record derivati dall'integrazione di terminologie internazionali quali SNOMED CT per le condizioni cliniche, LOINC per le misurazioni di laboratorio, RxNorm per i farmaci e ICD per le classificazioni diagnostiche, ciascuno classificato per dominio, vocabolario di origine e stato di standardizzazione. La distinzione tra il codice sorgente e il corrispondente concetto standard costituisce uno dei principi fondamentali del modello, poiché consente di armonizzare dati provenienti da sistemi eterogenei preservando al contempo la tracciabilità rispetto alla fonte originaria.

Relazioni gerarchiche tra concetti e supporto alle query

Le relazioni tra concetti sono gestite attraverso tabelle dedicate che permettono di rappresentare mapping, gerarchie ontologiche e relazioni di discendenza. In particolare, la presenza di relazioni gerarchiche tra i concetti permette di ampliare le interrogazioni cliniche includendo automaticamente concetti correlati, consentendo analisi su insiemi aggregati e migliorando la completezza e l'affidabilità degli studi osservazionali.

Come evidenziato in Figura 3.4, i concetti sono organizzati secondo strutture gerarchiche in cui ciascun elemento può essere messo in relazione con livelli superiori (ancestor) e inferiori (descendant). Questo consente, ad esempio, di risalire da una condizione specifica a categorie cliniche più generali o, viceversa, di espandere una query includendo tutte le sue specializzazioni.

In questo modo, interrogazioni definite su un singolo concetto possono essere automaticamente estese a tutti i concetti correlati lungo la gerarchia, riducendo il rischio di esclusione di informazioni rilevanti e facilitando la costruzione di coorti e analisi epidemiologiche su larga scala.

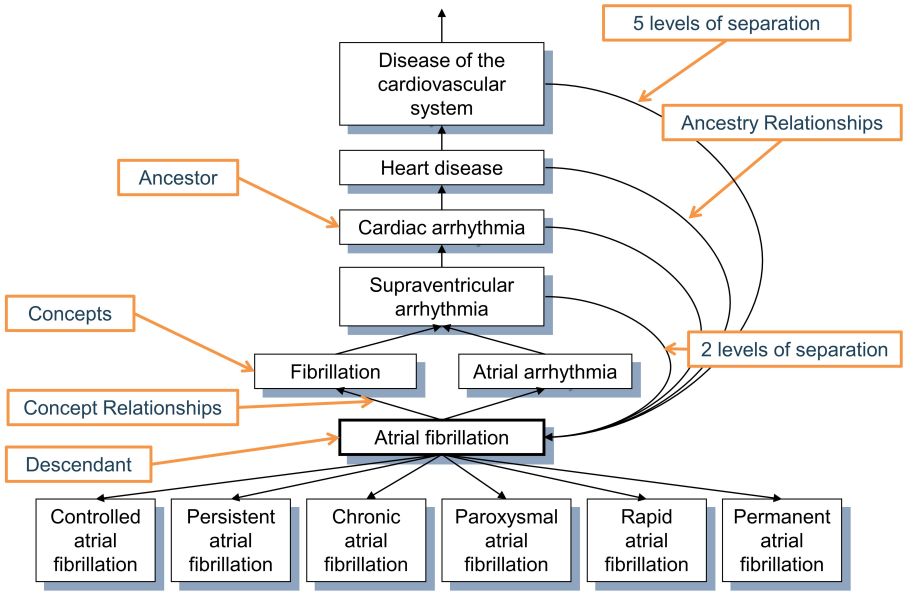


Figura 3.4: Relazioni gerarchiche tra concetti nel modello OMOP Common Data Model (CDM).
Fonte: https://www.ohdsi.org/ (The Book of OHDSI, 2019)

Tabelle di contesto e metadati

Accanto alle tabelle cliniche e ai vocabolari, lo schema include entità dedicate alla descrizione del contesto organizzativo, quali strutture sanitarie e operatori, nonché tabelle per la rappresentazione di informazioni economiche e assicurative. Completano il modello le tabelle di metadati, che documentano l'origine dell'istanza locale del CDM, la versione implementata e le caratteristiche della trasformazione dei dati, contribuendo alla trasparenza e alla riproducibilità delle analisi.

Nel contesto del presente lavoro di tesi, la comprensione dettagliata della struttura tecnica dello schema CDM è risultata fondamentale per la progettazione delle proce-

dure di estrazione e trasformazione dei dati e per l'implementazione dell'architettura applicativa che interagisce con il database conforme al modello OMOP.

3.1.2 Strumenti dell'ecosistema OHDSI

L'ecosistema OHDSI non si limita alla definizione del CDM, ma comprende un insieme integrato di strumenti open-source che supportano l'intero ciclo di vita del dato armonizzato, dalla gestione dei vocabolari fino all'analisi. Tuttavia, nell'ambito di questo lavoro di tesi, tali strumenti non vengono impiegati, in quanto l'attenzione è focalizzata esclusivamente sull'utilizzo del modello dati OMOP come base standardizzata per l'estrazione e l'integrazione delle informazioni cliniche. In questo contesto, come sintetizzato in Figura 3.5, la piattaforma mette a disposizione componenti dedicati alle diverse fasi del processo, che spaziano dalla standardizzazione dei dati fino alla loro esplorazione e visualizzazione.

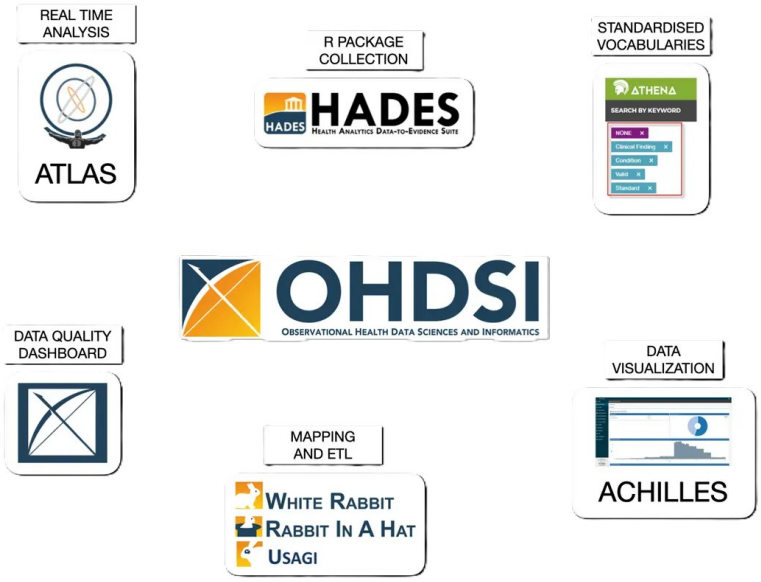


Figura 3.5: Ecosistema OHDSI e principali strumenti per l'analisi dei dati nel modello OMOP.
Fonte: https://www.ohdsi.org/ (The Book of OHDSI, 2019)

L'interfaccia web ATLAS consente la definizione e la gestione di coorti di pazienti, la progettazione di studi osservazionali e la caratterizzazione delle popolazioni in modo standardizzato e riproducibile. Essa si basa sul backend WebAPI, che espone servizi REST per l'interrogazione del database conforme al modello OMOP. Per l'analisi statistica sono disponibili diversi pacchetti sviluppati in ambiente R, tra

cui CohortMethod, dedicato al confronto tra coorti, e PatientLevelPrediction, per la costruzione di modelli predittivi basati su dati individuali.

La caratterizzazione e la valutazione preliminare del database possono essere effettuate mediante strumenti quali Achilles, che genera report descrittivi sulla distribuzione dei dati e Data Quality Dashboard, che consente di verificare la conformità del database alle specifiche del Common Data Model identificando eventuali anomalie o incoerenze.

La gestione e l'aggiornamento dei vocabolari standardizzati avviene attraverso il portale Athena, che mette a disposizione le versioni ufficiali dei concetti e delle relazioni semantiche integrate nel modello. L'integrazione tra struttura dati, vocabolari centralizzati e strumenti analitici interoperabili rappresenta il principale punto di forza dell'ecosistema OHDSI.

La diffusione globale di tale infrastruttura è evidenziata in Figura 3.6, che mostra l'adozione del modello OMOP e la presenza della comunità OHDSI in numerosi Paesi, a testimonianza della sua rilevanza nel contesto della ricerca collaborativa internazionale. Tale struttura consente la realizzazione di studi multicentrici distribuiti secondo un modello federato, nel quale ciascuna istituzione mantiene il controllo locale dei propri dati.

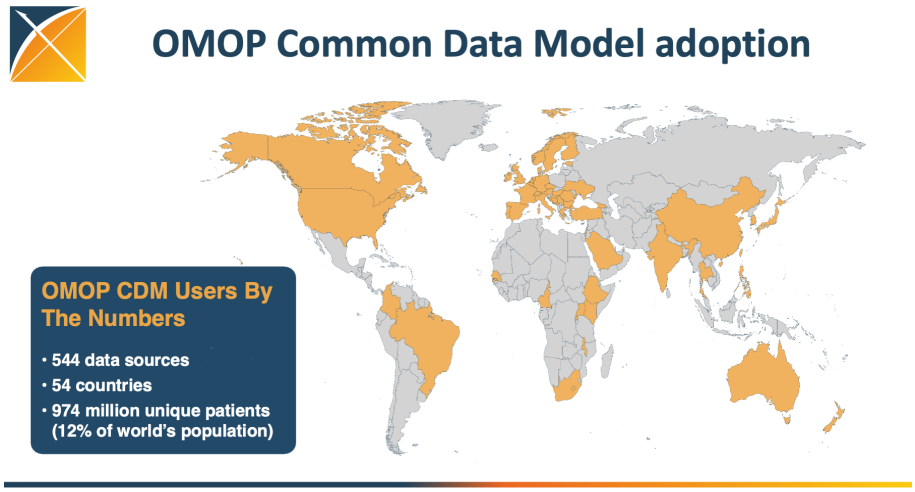


Figura 3.6: Adozione globale del modello OMOP Common Data Model (CDM) e distribuzione dei collaboratori OHDSI.
Fonte: https://www.ohdsi.org/

3.2 REDCap

3.2.1 eCRF

Raccolta strutturata dei dati nella ricerca clinica

Nell'ambito della ricerca clinica e osservazionale, la raccolta sistematica e strutturata dei dati rappresenta un elemento centrale per garantire qualità, affidabilità e riproducibilità degli studi. A tale scopo vengono impiegate le Case Report Form (CRF), strumenti progettati per la raccolta standardizzata delle informazioni relative ai soggetti arruolati in uno studio. Le CRF includono variabili anagrafiche, parametri clinici, esiti, trattamenti e dati di follow-up.

Evoluzione digitale: dalle CRF cartacee alle eCRF

Le eCRF rappresentano l'evoluzione digitale delle tradizionali schede cartacee di raccolta dati. La transizione dalla modalità cartacea a quella elettronica non rappresenta un semplice cambiamento di supporto, ma implica una riorganizzazione strutturale del processo di raccolta del dato, orientata alla standardizzazione preventiva delle variabili e all'integrazione di meccanismi automatici di controllo della qualità.

Struttura e configurazione delle eCRF

Le eCRF sono strutturate in moduli digitali organizzati per sezioni tematiche, ciascuna dedicata a uno specifico insieme di informazioni cliniche. I campi che compongono tali moduli sono tipizzati e configurabili secondo regole predefinite.

In fase di progettazione vengono definiti vincoli di formato, controlli di obbligatorietà e logiche condizionali che regolano la visualizzazione e la compilazione dei dati. Questa impostazione riduce il rischio di errori di inserimento, minimizza la presenza di valori mancanti e favorisce la coerenza del dataset raccolto.

Un esempio di interfaccia di eCRF è riportato in Figura 3.7, dove è possibile osservare la presenza di campi strutturati per l'inserimento di parametri clinici, valori numerici e selezioni guidate. L'organizzazione per sezioni e l'utilizzo di controlli

predefiniti evidenziano come la raccolta del dato sia orientata fin dall'origine alla standardizzazione e alla qualità delle informazioni acquisite.

Dal punto di vista metodologico, l'utilizzo di eCRF consente di acquisire sin dall'origine dati strutturati e digitalizzati, immediatamente disponibili per analisi statistiche, esportazione in formati interoperabili o integrazione con altri sistemi informativi. Tale impostazione agevola eventuali processi di trasformazione o mappatura e riduce la necessità di interventi successivi di data cleaning. La progettazione della scheda assume pertanto un ruolo cruciale, in quanto determina la qualità e la riusabilità del dataset nel tempo.

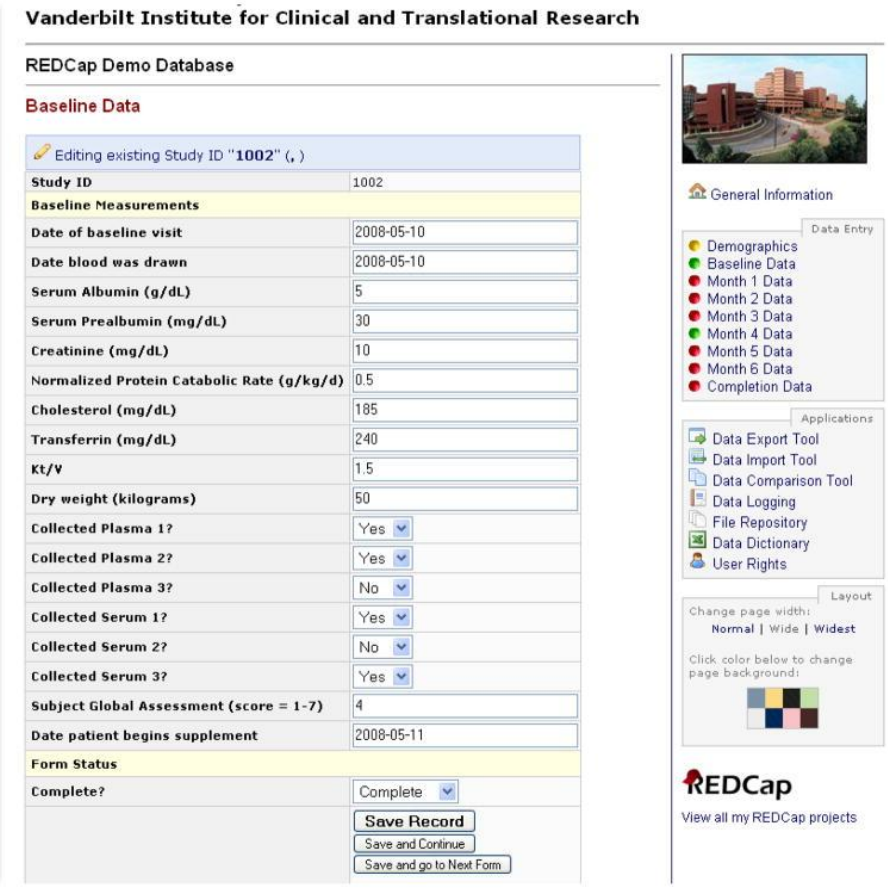


Figura 3.7: Esempio di interfaccia di eCRF per la raccolta strutturata di dati clinici.
Fonte: https://www.project-redcap.org/

REDCap come piattaforma per la gestione delle eCRF

Tra le piattaforme maggiormente diffuse per la progettazione e gestione di eCRF figura REDCap [1] (Research Electronic Data Capture), un'applicazione web sviluppata presso la Vanderbilt University e oggi adottata a livello internazionale in ambito accademico e ospedaliero. REDCap è stato concepito come un sistema di data capture orientato alla ricerca clinica, in grado di supportare la creazione, la gestione e l'esportazione di database strutturati, con particolare attenzione agli aspetti di sicurezza, tracciabilità e controllo della qualità del dato.

In questo contesto, il passaggio dalla raccolta dei dati tramite eCRF alla loro gestione all'interno della piattaforma avviene attraverso un flusso strutturato, illustrato in Figura 3.8, che evidenzia le fasi di acquisizione, elaborazione e trasferimento verso il sistema centrale.

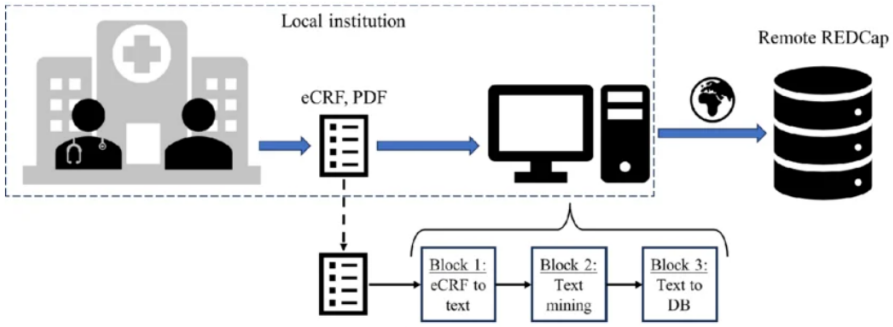


Figura 3.8: Flusso dati da eCRF a REDCap

Dal punto di vista architetturale, REDCap è una piattaforma interamente web-based, accessibile tramite browser senza necessità di installazione locale. Questa impostazione semplifica la distribuzione e la manutenzione del sistema, consentendo aggiornamenti centralizzati e garantendo uniformità tra le diverse sedi coinvolte in uno studio. L'interfaccia grafica è progettata per essere intuitiva e facilmente utilizzabile anche da personale clinico privo di competenze informatiche avanzate, favorendone l'adozione all'interno dei contesti ospedalieri e di ricerca.

La piattaforma dispone inoltre di un'applicazione mobile dedicata che consente la raccolta dei dati anche in modalità offline, con successiva sincronizzazione sul server centrale. Tale funzionalità risulta particolarmente utile in studi multicentrici o diffusi sul territorio.

3.2.2 Studio REDCap

In REDCap ogni attività di raccolta dati è organizzata all'interno di un progetto, questo rappresenta l'unità logico-operativa fondamentale della piattaforma. Il progetto definisce l'ambiente nel quale vengono configurati gli strumenti di raccolta, impostate le regole di accesso e archiviati i record relativi ai soggetti arruolati nello studio. Esso costituisce quindi il contenitore strutturato di tutte le informazioni cliniche raccolte nell'ambito di un protocollo di ricerca.

La struttura di un progetto REDCap è illustrata in Figura 3.9, dove sono visibili le principali sezioni di configurazione. In particolare, l'ambiente di setup consente di definire gli strumenti di raccolta dati (instruments), gestire gli eventi temporali negli studi longitudinali e configurare moduli aggiuntivi come la ripetizione degli strumenti o la pianificazione delle visite.

Tra le funzionalità più rilevanti si distinguono l'Online Designer, che permette la creazione e modifica grafica dei moduli eCRF, il Data Dictionary, utilizzato per la definizione strutturata delle variabili tramite file tabellari, e la sezione dedicata alla gestione degli eventi, che consente di associare gli strumenti ai diversi momenti dello studio. Questi strumenti permettono una configurazione flessibile e standardizzata del progetto, adattabile alle diverse esigenze di raccolta dati in ambito clinico e osservazionale.

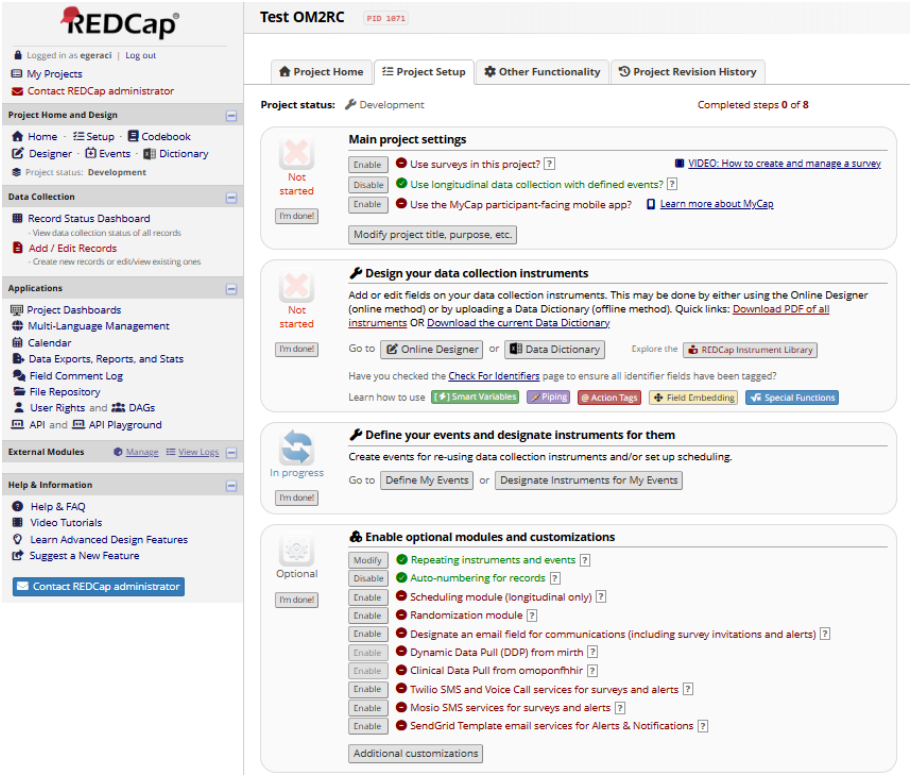


Figura 3.9: Interfaccia di configurazione di un progetto REDCap

In fase di creazione del progetto viene definita la tipologia dello studio, che può essere trasversale oppure longitudinale. Nel caso di uno studio trasversale, ciascun record è compilato una sola volta e rappresenta una fotografia statica del soggetto in un determinato momento, senza una struttura temporale articolata. In uno studio longitudinale, invece, il progetto è configurato per raccogliere dati in più momenti distinti nel tempo, associando gli strumenti a specifici eventi temporali, come visita basale, controlli periodici o follow-up. Questa impostazione consente di monitorare l'evoluzione clinica del soggetto lungo un arco temporale definito, mantenendo separati ma correlati i dati raccolti nei diversi momenti dello studio. Un esempio di configurazione longitudinale è mostrato in Figura 3.10, dove è possibile osservare la distribuzione degli strumenti di raccolta (data collection instruments) sui diversi eventi temporali (baseline e visite successive). La struttura a matrice consente di visualizzare in modo immediato lo stato di compilazione dei dati per ciascun evento, facilitando il monitoraggio del progresso dello studio e la gestione delle informazioni nel tempo.

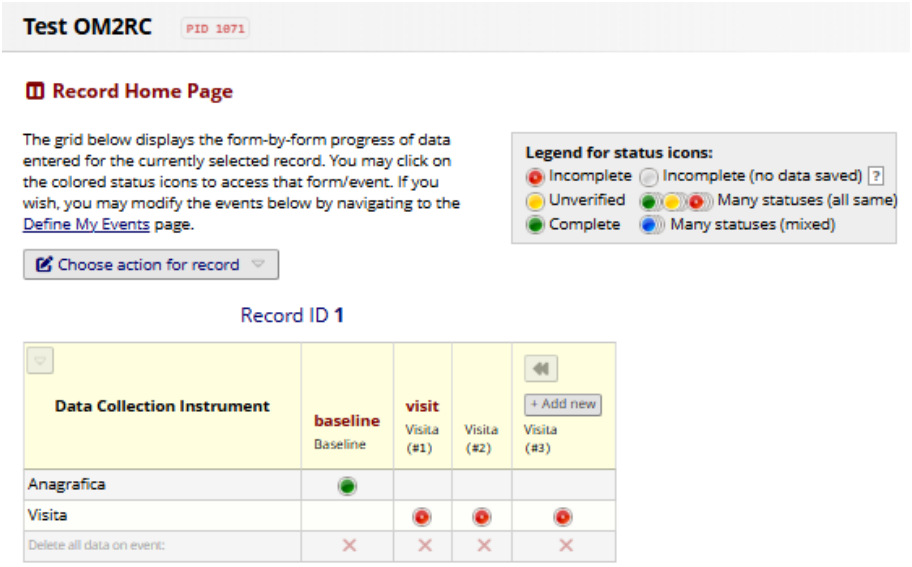


Figura 3.10: Esempio di configurazione longitudinale in REDCap

Per ciascun progetto vengono definiti gli strumenti di raccolta dati, detti instruments, che corrispondono ai moduli della eCRF. Ogni strumento è composto da un insieme di variabili configurabili e rappresenta una sezione tematica dello studio. La progettazione degli strumenti può avvenire tramite un editor grafico oppure attraverso il caricamento di un data dictionary in formato tabellare, che consente di definire in modo strutturato le proprietà di ciascun campo.

L'unità informativa fondamentale all'interno del progetto è il record, che rappresenta l'insieme delle informazioni relative a un singolo soggetto. A ciascun record è associato un identificativo univoco, definito in fase di configurazione del progetto. I dati inseriti nei diversi strumenti confluiscono nel record corrispondente, permettendo una visione integrata delle informazioni relative al paziente lungo l'intero percorso di studio.

Sicurezza e tracciabilità delle operazioni

L'architettura di REDCap consente l'accesso controllato tramite autenticazione degli utenti e la definizione di ruoli differenziati, garantendo così la separazione delle responsabilità e la protezione delle informazioni sensibili. Ogni operazione effettuata all'interno del sistema viene registrata attraverso un meccanismo di audit trail che traccia modifiche, inserimenti ed esportazioni dei dati.

REDCap API

Un elemento di particolare rilievo è la disponibilità di un'API (Application Programming Interface), che consente l'automatizzazione delle operazioni di importazione ed esportazione dei dati e l'integrazione con sistemi esterni. Attraverso le API è possibile sviluppare flussi di trasferimento automatizzati, riducendo l'intervento manuale e migliorando l'affidabilità dei processi di scambio informativo. L'accesso programmato ai dati tramite API richiede l'ottenimento di un API token, una chiave di autenticazione associata a un utente e a uno specifico progetto; tale token garantisce che le richieste siano autorizzate e che l'ambiente rispetti i permessi configurati nel profilo utente. Una volta autenticata, un'applicazione esterna può inviare richieste HTTP al server REDCap per effettuare operazioni di esportazione o importazione dei dati.

Per quanto riguarda l'esportazione dei dati, l'API consente di richiedere l'estrazione di record in formati standard quali CSV, JSON o XML, restituendo il contenuto delle variabili raccolte nel progetto secondo i criteri specificati nella richiesta. Le chiamate API di esportazione possono includere parametri per selezionare sottogruppi di record, specifici campi o forme di aggregazione, e possono essere automatizzate all'interno di pipeline di analisi o di integrazione con altri sistemi.

L'importazione dei record avviene tramite specifiche richieste API nelle quali vengono forniti i valori da inserire o aggiornare nel progetto REDCap. In questa modalità si possono creare nuovi record o aggiornare record esistenti. La struttura dei dati inviati deve rispettare le definizioni presenti nel progetto, compresi nomi di variabile, formati di data e regole di validazione.

Nel contesto del presente lavoro di tesi, il progetto REDCap costituisce il punto di destinazione dei dati estratti dal database OMOP. La corretta configurazione degli strumenti, dei campi e degli eventi rappresenta pertanto una fase determinante per garantire coerenza strutturale tra i dati armonizzati nel Common Data Model e la loro rappresentazione all'interno della eCRF. In questo senso, REDCap non rappresenta soltanto uno strumento di inserimento dati, ma un'infrastruttura che contribuisce in modo determinante alla qualità, tracciabilità e riusabilità del dato clinico nell'ambito dei processi di integrazione per la ricerca.

Capitolo 4

Stato dell'arte

Negli ultimi anni sono state proposte diverse soluzioni per supportare lo scambio di dati tra OMOP e REDCap, sia in modalità unidirezionale sia bidirezionale. Tali approcci si basano generalmente su processi di tipo ETL, su tecniche di mapping semantico tra variabili e concetti standardizzati, oppure sull'utilizzo di standard intermedi, come HL7 FHIR.

Questo capitolo presenta una panoramica delle principali soluzioni già esistenti in letteratura e in ambito applicativo per l'integrazione tra OMOP e REDCap. In particolare, verrà descritto un tool sviluppato per il trasferimento dei dati da REDCap a OMOP, verranno analizzate le funzionalità di Dynamic Data Pull e Clinical Data Pull offerte da REDCap, e verranno infine esaminati approcci che prevedono l'integrazione con altri standard e sistemi, evidenziandone caratteristiche, vantaggi e limiti.

4.1 Integrazione tra OMOP e REDCap

4.1.1 Da OMOP a REDCap

Un primo prototipo di trasferimento automatico dei dati dal modello OMOP CDM verso REDCap[2] è stato sviluppato nel 2021 da Biomeris, spin-off accademico dell'Università degli Studi di Pavia[10]. L'obiettivo principale di questo lavoro è la definizione di un framework in grado di supportare il trasferimento automatico dei dati da OMOP CDM alle eCRF di REDCap, riducendo l'inserimento manuale e migliorando la qualità e la coerenza delle informazioni raccolte.

L'architettura del sistema si basa su due componenti principali: da un lato un framework di annotazione semantica, utilizzato per estendere i metadati di uno studio REDCap, e dall'altro un componente software responsabile dell'esecuzione del trasferimento dei dati. L'elemento centrale dell'approccio è rappresentato proprio dall'annotazione semantica, che viene integrata all'interno del REDCap Data Dictionary, ovvero la struttura che definisce le variabili e i campi dello studio.

Le annotazioni sono espresse in formato JSON e organizzate in diverse sezioni, ciascuna dedicata a un aspetto specifico del processo di integrazione. In primo luogo, viene definita la corrispondenza tra le variabili REDCap e i concetti presenti nei vocabolari standardizzati di OMOP. Questo permette di associare a ciascun campo della eCRF il concetto clinico più appropriato, ad esempio utilizzando codici SNOMED per rappresentare variabili come il genere del paziente o i relativi valori possibili.

Oltre alla definizione dei concetti, l'annotazione semantica include informazioni relative alle modalità con cui il trasferimento dei dati deve essere effettuato. In particolare, viene specificata la data di riferimento associata a ciascun elemento tempodipendente, come nel caso dei valori di laboratorio, al fine di garantire un recupero accurato dei dati dal database OMOP. A questa informazione si affianca la possibilità di definire una tolleranza temporale, utile per gestire eventuali discrepanze tra le date registrate nei diversi sistemi e rendere le query più flessibili.

Ulteriori aspetti gestiti attraverso le annotazioni riguardano il trattamento delle dipendenze tra variabili, ad esempio in presenza di logiche di ramificazione tipiche delle eCRF, e la definizione dell'ordine con cui devono essere eseguiti i tentativi di trasferimento dei dati, sia per singole variabili sia per gruppi di variabili correlate. Questo consente di controllare in modo fine il flusso di esecuzione e di gestire correttamente eventuali dipendenze tra i dati.

Il secondo componente del sistema è rappresentato da uno strumento software sviluppato in linguaggio Java, che ha il compito di interpretare le annotazioni semantiche e di orchestrare l'intero processo di trasferimento. Il software accede al REDCap Data Dictionary tramite le API standard della piattaforma, legge le informazioni di configurazione e utilizza una tabella di lookup esterna per associare gli identificativi dei record REDCap con quelli corrispondenti nel database OMOP.

Sulla base di queste informazioni, il sistema esegue il trasferimento dei dati secondo diverse modalità operative. Il processo può essere pianificato su base periodica, ad esempio con esecuzioni giornaliere automatiche, oppure può essere attivato in rispo-

sta a eventi specifici, come la creazione di un nuovo paziente in REDCap attraverso meccanismi di trigger. In alternativa, è possibile eseguire il trasferimento in modalità on-demand, lasciando all'utente la possibilità di avviare manualmente il processo quando necessario.

Questo approccio consente di ottenere un elevato grado di automazione e flessibilità, grazie alla separazione tra configurazione e logica applicativa. Tuttavia, l'efficacia del sistema dipende fortemente dalla qualità delle annotazioni semantiche e dalla corretta definizione delle regole di mapping, che richiedono una conoscenza approfondita sia del modello OMOP sia della struttura degli studi REDCap.

4.1.2 Da REDCap a OMOP

Nel verso opposto, da REDCap verso OMOP, uno dei contributi più rilevanti è stato sviluppato nel 2021 nell'ambito del COVID-19 and Cancer Consortium (CCC19). Il CCC19 rappresenta un registro di ricerca longitudinale multi-istituzionale, nato con l'obiettivo di raccogliere e analizzare dati relativi a pazienti adulti affetti da neoplasie invasive, attuali o pregresse, a cui sia stato diagnosticato il COVID-19.

Per supportare la raccolta dati su larga scala e garantire uniformità tra le diverse istituzioni partecipanti, il consorzio utilizza un progetto REDCap centralizzato, attraverso il quale vengono inserite informazioni strutturate relative a diagnosi, trattamenti ed esiti clinici. Tuttavia, al fine di rendere tali dati pienamente sfruttabili per analisi avanzate e confrontabili con altri dataset clinici, è emersa la necessità di convertirli nel formato standardizzato del modello OMOP CDM.

Per rispondere a questa esigenza è stata sviluppata la piattaforma REDCap2OMOP[3], un sistema open source rilasciato con licenza MIT, progettato per gestire in modo sistematico e generalizzabile la trasformazione dei dati da REDCap a OMOP. L'obiettivo principale è quello di abilitare l'utilizzo delle librerie metodologiche disponibili nell'ecosistema OHDSI, costituite da pacchetti software, prevalentemente in linguaggio R, che implementano metodi statistici standardizzati per l'analisi dei dati osservazionali.

L'architettura complessiva del sistema è rappresentata in Figura 4.1 e si basa su una pipeline di tipo ETL articolata in più componenti, che consentono la gestione del data dictionary, delle mappature semantiche e del processo di trasformazione dei dati.

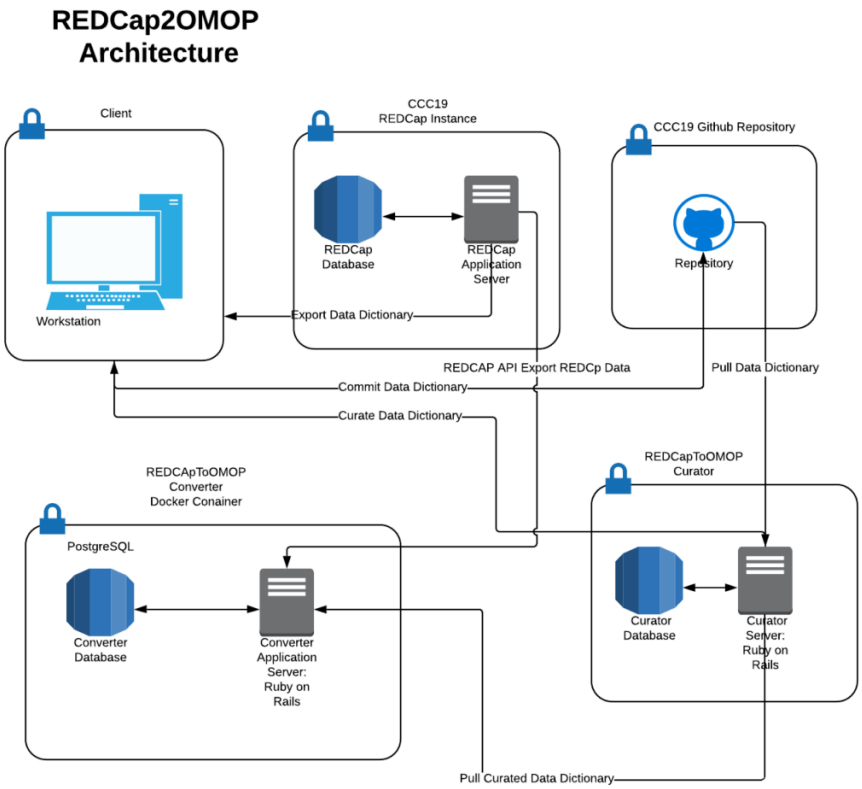


Figura 4.1: Architettura del sistema REDCap2OMOP.
Fonte: REDCap2OMOP (repository GitHub)

Dal punto di vista architetturale, REDCap2OMOP si basa su una pipeline di tipo ETL ed è costituito da due componenti principali. Il primo è rappresentato dal Curator, un'interfaccia web che consente di gestire in maniera centralizzata le versioni del data dictionary REDCap, le mappature tra variabili REDCap e concetti del vocabolario OMOP e le informazioni temporali associate ai dati. Il Curator svolge un ruolo fondamentale nel garantire la coerenza semantica del processo, permettendo di tracciare le modifiche nel tempo e di aggiornare automaticamente le mappature in funzione delle evoluzioni del data dictionary.

In particolare, il sistema estrae periodicamente il data dictionary dal repository GitHub del progetto CCC19, confronta la versione corrente con quella precedente e, se necessario, genera una nuova versione aggiornata, mantenendo allineate anche le mappature del vocabolario OMOP e le designazioni temporali già definite. Per facilitare la selezione dei concetti più appropriati, il Curator integra una copia completa del vocabolario OMOP (versione 5.3.1), popolata a partire dalle risorse disponibili tramite Athena, consentendo agli utenti di effettuare ricerche e associazioni in modo guidato.

Il secondo componente è il Converter, responsabile dell'esecuzione del processo ETL vero e proprio. Attraverso un'interfaccia semplificata, l'amministratore può selezionare una specifica versione del data dictionary REDCap, acquisire i dati completi del progetto tramite le API ufficiali e avviare il processo di trasformazione. Prima di procedere al caricamento, il sistema verifica che i dati esportati siano conformi alla versione del data dictionary selezionata; in caso contrario, il processo viene interrotto e le eventuali incongruenze vengono segnalate all'utente.

Una volta superata la fase di validazione, il Converter esegue il caricamento dei dati nel database OMOP, tipicamente tramite un'operazione di tipo "truncate and load", applicando le mappature semantiche e le regole temporali definite in precedenza. I dati vengono quindi distribuiti nelle diverse tabelle del CDM. Al termine dell'esecuzione, il sistema restituisce un report che consente di monitorare l'esito del processo e di identificare eventuali criticità.

Nel complesso, REDCap2OMOP rappresenta una soluzione strutturata e riutilizzabile per l'integrazione tra REDCap e OMOP, in grado di supportare contesti multi-istituzionali e dataset complessi. Tuttavia, la qualità del risultato finale dipende fortemente dall'accuratezza delle mappature semantiche e dalla corretta gestione delle evoluzioni del data dictionary, elementi che richiedono competenze specifiche sia sul modello OMOP sia sulla progettazione degli studi REDCap.

4.2 REDCap Dynamic Data Pull e Clinical Data Pull

L'integrazione tra sistemi clinici e piattaforme di raccolta dati rappresenta una delle principali sfide nei contesti di ricerca basata su dati real-world. Tradizionalmente, tale integrazione è stata realizzata attraverso processi manuali o pipeline basate su esportazione e riformattazione dei dati, che risultano spesso inefficienti e soggette a errori.

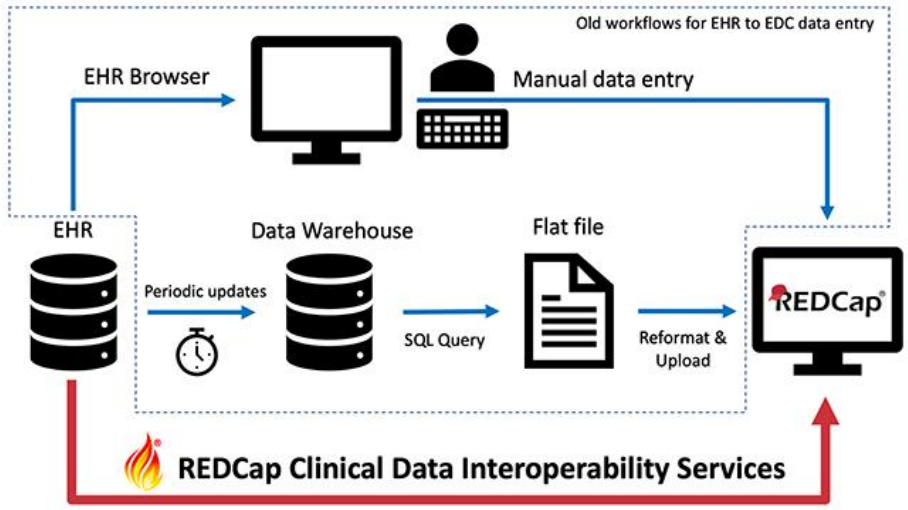


Figura 4.2: Workflow tradizionali di integrazione dei dati tra sistemi EHR e REDCap.
Fonte: https://www.project-redcap.org/

Come illustrato in Figura 4.2, i dati possono essere inseriti manualmente oppure trasferiti tramite passaggi intermedi, come data warehouse e file di esportazione, che richiedono ulteriori operazioni di trasformazione e caricamento. Questi approcci introducono ritardi, aumentano il rischio di errori e limitano la scalabilità del processo.

Per superare tali limitazioni, REDCap mette a disposizione funzionalità avanzate di integrazione, tra cui il Dynamic Data Pull (DDP) e il Clinical Data Pull (CDP), che consentono un accesso più diretto e controllato ai dati provenienti da sistemi esterni.

4.2.1 Dynamic Data Pull

Il Dynamic Data Pull (DDP) rappresenta una funzionalità nativa di REDCap che consente l'importazione in modo controllato e configurabile di dati da sorgenti esterne. Questo meccanismo permette di inserire informazioni provenienti da sistemi clinici o database esterni direttamente all'interno di un progetto REDCap, sia in modo manuale in tempo reale sia con modalità automatizzate basate su interrogazioni periodiche.

Affinché il processo di importazione possa essere eseguito, è necessario che il record sia già presente nel progetto REDCap e che sia identificato in modo univoco. Il DDP utilizza infatti tale identificativo per interrogare la sorgente esterna e recuperare i dati corrispondenti. Una volta stabilita la corrispondenza, il sistema può aggiornare i dati in tempo reale oppure verificarne periodicamente la disponibilità, consentendo l'aggiunta progressiva di nuove informazioni.

L'attivazione del DDP viene consentita solo a utenti con profili amministrativi. Questi possono accedere a un'interfaccia di configurazione che consente di definire il mapping tra i campi del progetto REDCap e quelli della sorgente esterna. Questo processo di mappatura è centrale per il corretto funzionamento del sistema, stabilisce la corrispondenza tra le variabili e determina le modalità con cui i dati vengono recuperati e integrati.

I dati gestiti tramite DDP possono essere distinti in due categorie principali. Da un lato vi sono i dati statici, che vengono acquisiti una sola volta e non sono soggetti a ulteriori aggiornamenti; dall'altro vi sono i dati temporali, che possono essere raccolti ripetutamente nel tempo. Nel secondo caso, è necessario definire una finestra temporale di interesse, specificando una data di riferimento e un intervallo entro il quale i dati devono essere considerati validi. Informazioni che non rientrano in tale intervallo vengono automaticamente scartate dal processo di importazione.

Durante la fase di configurazione, REDCap mette a disposizione strumenti di preview che permettono di verificare la correttezza del mapping e dell'identificazione dei record così da verificare il controllo della qualità dei dati prima della loro effettiva integrazione. Una volta completata la configurazione, i dati recuperati non vengono inseriti automaticamente nel database del progetto, ma devono essere esplicitamente approvati dall'utente. Questo meccanismo introduce un ulteriore livello di controllo, riducendo il rischio di inserimento di dati errati o non coerenti. L'approvazione dei dati può avvenire attraverso diverse interfacce della piattaforma, come il pannello di

monitoraggio dello stato dei record o direttamente all'interno dei form di inserimento dati, dove viene segnalata la presenza di nuove informazioni disponibili.

Dal punto di vista architetturale, il DDP si basa su un sistema di comunicazione tramite servizi web esterni, che devono essere implementati dall'host e non sono forniti nativamente da REDCap. La comunicazione tra REDCap e i servizi web avviene tramite richieste HTTP(S) standardizzate, tipicamente in formato JSON, e può essere protetta attraverso meccanismi di sicurezza come whitelist di indirizzi IP o l'utilizzo di chiavi segrete.

In particolare, il funzionamento del DDP prevede l'utilizzo di diversi servizi web con ruoli distinti. Un primo servizio è dedicato alla gestione dei metadati e viene interrogato durante la fase di configurazione del mapping. Un secondo servizio si occupa del recupero effettivo dei dati per ciascun record, riceve in input le richieste formulate da REDCap e restituisce i dati nel formato previsto. Infine, è possibile prevedere un servizio opzionale per la gestione dei permessi utente, che consente di verificare se un determinato utente sia autorizzato ad accedere ai dati della sorgente esterna.

Nel complesso, il Dynamic Data Pull rappresenta uno strumento potente per l'integrazione dei dati in REDCap, in grado di supportare scenari di aggiornamento continuo e di ridurre il carico di inserimento manuale. Tuttavia, la sua efficacia dipende dalla corretta configurazione dei servizi web e delle regole di mapping, oltre che dalla qualità e dalla coerenza dei dati disponibili nella sorgente esterna.

4.2.2 Clinical Data Pull

Il Clinical Data Pull (CDP) rappresenta un'estensione delle funzionalità di integrazione offerte da REDCap, progettata specificamente per l'importazione di dati provenienti da sistemi di cartelle cliniche elettroniche. A differenza del DDP, che si basa su servizi web generici configurati dall'host, il CDP è stato concepito per operare direttamente in ambienti clinici, sfruttando standard di interoperabilità consolidati. Un'altra differenza rilevante rispetto al DDP è la possibilità di creare nuovi record direttamente all'interno di un progetto REDCap a partire dall'interfaccia del sistema EHR (Electronic Health Record). In questo modo, il paziente visualizzato nel sistema clinico può essere associato immediatamente a un record di ricerca, rendendo più diretto il collegamento tra attività assistenziale e raccolta strutturata dei dati.

Dal punto di vista funzionale, il CDP presenta diverse analogie con il DDP. Anche in questo caso, l'utilizzo della funzionalità richiede l'abilitazione del modulo a livello di progetto da parte di un amministratore REDCap. Il sistema supporta sia dati statici acquisiti una sola volta sia dati temporali che possono essere aggiornati nel tempo, e mette a disposizione strumenti di preview per verificare la correttezza delle informazioni prima della loro integrazione. Inoltre, i dati recuperati devono essere approvati dall'utente prima di essere definitivamente inseriti nel progetto, garantendo così un controllo sulla qualità e sulla coerenza delle informazioni. Il recupero dei dati può essere effettuato sia in modalità manuale sia attraverso meccanismi automatici basati su interrogazioni periodiche.

Dal punto di vista architetturale, il CDP si basa sull'infrastruttura Clinical Data Interoperability Services (CDIS) di REDCap, che consente l'interfacciamento con sistemi EHR compatibili con lo standard HL7 FHIR. In particolare, CDIS utilizza lo stack tecnologico SMART on FHIR, che permette lo scambio di dati clinici attraverso servizi web HTTP standardizzati, utilizzando un formato interoperabile basato su risorse FHIR. Questo approccio consente di superare le limitazioni dei sistemi proprietari e di favorire l'integrazione tra piattaforme eterogenee.

L'utilizzo del CDP prevede un flusso operativo articolato. In primo luogo, è necessario avviare un processo di EHR Launch, che consiste nell'apertura di una finestra REDCap direttamente dall'interfaccia del sistema EHR. Questo passaggio consente di avviare il processo di autenticazione e autorizzazione ai servizi FHIR. All'interno di questa interfaccia, l'utente può selezionare uno dei progetti REDCap abilitati al CDP e accedere al record del paziente oppure crearne uno nuovo.

Una volta completata la fase di autorizzazione, è possibile procedere con l'estrazione dei dati clinici del paziente. Se l'accesso avviene tramite EHR Launch, il sistema è già in possesso del contesto del paziente e indirizza automaticamente l'utente al relativo record. In alternativa, è possibile operare direttamente all'interno di REDCap, selezionando un record esistente oppure creandone uno nuovo. In quest'ultimo caso, l'identificazione del paziente avviene tipicamente tramite l'inserimento dell'MRN (Medical Record Number), che consente al sistema di recuperare in tempo reale i dati dal sistema EHR.

Nel complesso, il CDP rappresenta una soluzione avanzata per l'integrazione tra REDCap e sistemi clinici, in grado di supportare flussi di lavoro più integrati rispetto al DDP. L'utilizzo di standard come FHIR e SMART on FHIR consente di garantire interoperabilità e scalabilità, rendendo il CDP particolarmente adatto a

contesti ospedalieri e multi-istituzionali. Tuttavia, la sua implementazione richiede la disponibilità di infrastrutture compatibili e l'integrazione con sistemi EHR che supportino tali standard. Risulta quindi essere una soluzione più complessa rispetto al DDP ma al contempo più potente e strutturata.

4.3 Integrazione di OMOP e REDCap con altri formati

4.3.1 Da i2b2 a REDCap

Un'ulteriore linea di sviluppo nell'ambito dell'integrazione tra piattaforme per la ricerca clinica riguarda il collegamento tra i2b2 e REDCap. i2b2[5][13], acronimo di Informatics for Integrating Biology and the Bedside, è una piattaforma open source introdotta nel 2004 nell'ambito di un progetto promosso dal National Institutes of Health (NIH). È nata per supportare la costruzione di data warehouse clinici orientati all'integrazione, all'esplorazione e al riutilizzo di dati eterogenei provenienti da contesti assistenziali e di ricerca. La piattaforma è stata progettata per favorire attività di cohort discovery, analisi retrospettive e supporto alla ricerca traslazionale, affermandosi nel tempo come una delle soluzioni più diffuse per l'organizzazione di repository clinici interoperabili. In modo analogo a OMOP, anche i2b2 consente l'integrazione e l'analisi di dati clinici eterogenei.
In questo contesto, nel 2019 Biomeris[10] ha sviluppato un'estensione della piattaforma i2b2 compatibile con il meccanismo di DDP di REDCap. L'idea alla base del progetto è che REDCap, pur mettendo a disposizione un modulo standard per l'importazione di dati da sorgenti esterne, non richiede che tali sorgenti abbiano una struttura specifica, ma stabilisce unicamente le modalità con cui devono comunicare con la piattaforma. A partire da questa impostazione, è stata quindi realizzata un'estensione in grado di adattare i2b2 alle specifiche richieste da REDCap, così da consentirne l'utilizzo come sorgente dati esterna per la compilazione automatica delle eCRF.

Dal punto di vista architetturale, l'estensione sviluppata per i2b2 si basa su due servizi web accessibili da REDCap e connessi al database del data warehouse. Tali servizi corrispondono ai due componenti minimi richiesti dal DDP: il Metadata Web Service, incaricato di restituire a REDCap le informazioni necessarie alla configurazione del mapping, e il Data Web Service, responsabile del recupero vero e

proprio dei dati relativi a un singolo record. In questo modo, REDCap può interrogare i2b2 secondo il protocollo previsto dal DDP senza necessità di modificare il funzionamento interno della piattaforma. L'architettura complessiva del processo è rappresentata in Figura 4.3, che illustra il flusso di integrazione tra i due sistemi attraverso i servizi web e il meccanismo di interrogazione basato su identificativo del record.

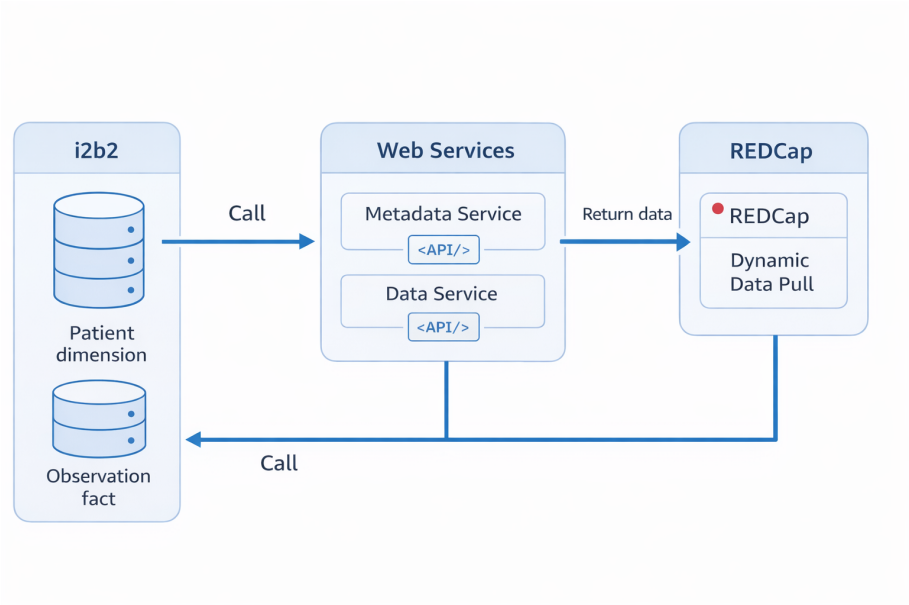


Figura 4.3: Schema di integrazione tra i2b2 e REDCap tramite Dynamic Data Pull

Quando REDCap invia una chiamata al Data Web Service, il sistema riceve un identificativo relativo al record da aggiornare. Il primo passaggio consiste quindi nell'associare questo identificativo con l'ID corrispondente del paziente all'interno di i2b2. Una volta stabilita la corrispondenza tra i due sistemi, l'applicazione seleziona gli elementi richiesti da REDCap tra quelli disponibili nel data warehouse e li trasforma secondo le modalità previste da un file di configurazione. Tale file, espresso in formato JSON, contiene sia la definizione dei campi importabili sia le regole necessarie per convertirli nel formato richiesto dal progetto REDCap.

Nel caso specifico del lavoro sviluppato da Biomeris, le sorgenti dati considerate all'interno di i2b2 sono due. La prima è la tabella patient dimension, che contiene le informazioni demografiche del soggetto, come data di nascita, sesso, razza o data di morte. La seconda è la tabella observation fact, che rappresenta il nucleo informativo del modello i2b2 e memorizza i fatti clinici associati ai pazienti, tra cui diagnosi, procedure, esami di laboratorio e trattamenti farmacologici. Poiché queste due sorgenti hanno caratteristiche strutturali differenti, anche il processo di recupero

dei dati varia in funzione della tabella da cui l'informazione viene estratta. Quando il dato richiesto proviene da patient dimension, il recupero è relativamente diretto, in quanto consiste nella selezione di uno specifico attributo demografico associato al paziente. Successivamente, il valore può essere trasformato per adattarsi al formato previsto nello studio REDCap, ad esempio tramite una semplice tabella di lookup. Questo consente di gestire eventuali differenze di codifica tra la sorgente i2b2 e il progetto REDCap, mantenendo la coerenza del dato in fase di importazione.

Nel complesso, questa estensione dimostra come l'integrazione tra i2b2 e REDCap possa essere realizzata sfruttando il modulo DDP come meccanismo di interoperabilità applicativa. L'approccio risulta particolarmente interessante perché valorizza un'infrastruttura di data warehouse già esistente, rendendola direttamente utilizzabile per la raccolta dati in contesti di ricerca. Allo stesso tempo, la necessità di definire mapping, regole di trasformazione e associazioni tra identificativi evidenzia come anche in questo caso l'interoperabilità non dipenda esclusivamente dalla connessione tecnica tra sistemi, ma richieda un attento lavoro di armonizzazione semantica e strutturale.

4.3.2 Da openEHR a FHIR e OMOP

Tra le soluzioni proposte per l'integrazione tra standard differenti rientra il progetto HiGHmed, un consorzio tedesco composto da otto ospedali universitari, nato con l'obiettivo di favorire lo scambio di dati clinici tra istituzioni attraverso soluzioni avanzate di informatica medica. Nell'ambito di questo lavoro, il gruppo dedicato al caso d'uso Infection Control ha modellato un insieme di dati relativi alle infezioni utilizzando lo standard openEHR[6].

I dati, inizialmente rappresentati in openEHR, vengono successivamente convertiti nel formato FHIR (Fast Healthcare Interoperability Resources) per garantire l'interoperabilità con gli altri consorzi coinvolti nella stessa iniziativa nazionale. In una fase successiva, al fine di sfruttare le capacità analitiche offerte dalla comunità OHDSI, le informazioni espresse in FHIR vengono mappate nel modello OMOP CDM. Questo articolato processo di trasformazione consente di combinare i vantaggi dei diversi standard, favorendo sia lo scambio dei dati sia la loro analisi.

Come evidenziato in Figura 4.4, i modelli FHIR e OMOP CDM rispondono a esigenze differenti e complementari. FHIR è progettato per lo scambio di dati tra sistemi sanitari in tempo reale, attraverso strutture flessibili basate su risorse e interfacce

API, mentre OMOP è orientato all'organizzazione e all'analisi di dati osservazionali su larga scala all'interno di uno schema relazionale standardizzato. Questa distinzione evidenzia come il passaggio da FHIR a OMOP non rappresenti una semplice conversione di formato, ma una trasformazione concettuale che consente di passare da un contesto di interoperabilità a uno di analisi dei dati.

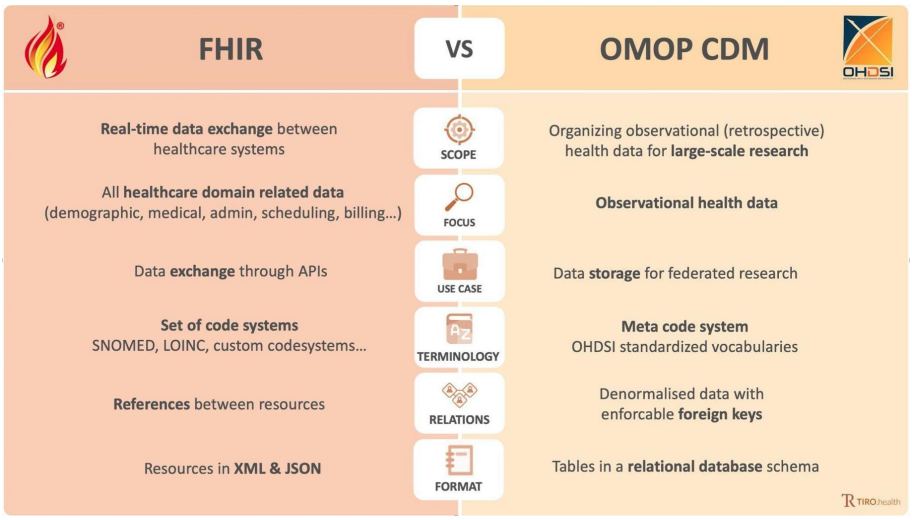


Figura 4.4: Confronto tra FHIR e OMOP Common Data Model (CDM).
Fonte: https://www.ohdsi.org/

Sia openEHR che FHIR, sebbene con approcci differenti, si basano su modelli riutilizzabili per la rappresentazione dei dati clinici. In openEHR tali modelli prendono il nome di archetipi e descrivono in modo completo un concetto clinico, mentre in FHIR le informazioni sono organizzate in risorse, progettate per essere più modulari e facilmente estendibili. Questa differenza strutturale ha implicazioni importanti nella fase di mapping tra i due standard.

Nel caso d'uso analizzato, i membri del consorzio HiGHmed hanno definito un dataset minimo condiviso, comprendente dati amministrativi, informazioni sui movimenti dei pazienti e dati microbiologici. Il gruppo di modellazione ha quindi rappresentato tali informazioni in openEHR utilizzando sia archetipi esistenti sia nuovi modelli, per poi effettuare la conversione verso FHIR attraverso l'impiego dello standard HL7 (Health Level 7) v2.

Durante questa fase è emersa una criticità legata al diverso grado di dettaglio dei due modelli: gli archetipi openEHR tendono a fornire una rappresentazione completa e dettagliata di un concetto clinico, mentre le risorse FHIR presentano una struttura più minimale, che può essere estesa secondo necessità. Tuttavia, nel caso specifico del dataset considerato, non è stato necessario introdurre estensioni, poiché la corri-

spondenza tra gli elementi openEHR e le risorse FHIR ha raggiunto una copertura completa per i dati microbiologici.

La successiva trasformazione da FHIR a OMOP CDM introduce ulteriori complessità dovute alle differenze tra il modello orientato alle risorse di FHIR e la struttura relazionale e altamente normalizzata di OMOP. In particolare, una singola risorsa FHIR può contenere informazioni che, in OMOP, devono essere distribuite tra diverse tabelle in base alla loro semantica e al tipo di dato.

Un esempio significativo riguarda la gestione degli esami microbiologici. In FHIR, tali informazioni sono generalmente rappresentate tramite la risorsa Observation, che include sia dati strutturati sia descrizioni dell'esame. In OMOP, invece, è necessario distinguere tra test standardizzati, che producono risultati numerici e devono essere inseriti nella tabella Measurement, e osservazioni più generiche, che vengono inserite nella tabella Observation. Per risolvere questa ambiguità, è stato adottato un criterio basato sull'utilizzo di codici LOINC: tutte le osservazioni associate a un codice LOINC vengono mappate nella tabella Measurement, mentre le altre vengono assegnate alla tabella Observation.

Nel complesso, il caso d'uso Infection Control dimostra la fattibilità di un'integrazione tra openEHR, FHIR e OMOP attraverso una pipeline di trasformazione progressiva. Tuttavia, l'efficacia di questo approccio dipende fortemente dalla dimensione e dalla complessità del dataset: mentre per un insieme di dati limitato la mappatura risulta gestibile, l'estensione a dataset più ampi e articolati potrebbe introdurre ulteriori difficoltà, sia dal punto di vista semantico sia da quello computazionale.

4.3.3 FHIRCap: da REDCap a FHIR

FHIRCap[7][8] è un sistema progettato per consentire l'esportazione dei dati raccolti in REDCap in forma di risorse conformi allo standard HL7 FHIR. Rappresenta il componente centrale di un'architettura più ampia, concepita per raccogliere dati fenotipici eterogenei e integrarli all'interno di un repository centralizzato e standardizzato. Il sistema complessivo è costituito da più istanze REDCap, dal motore di trasformazione FHIRCap, da un server FHIR (in questo caso il server CSIRO) e da un server terminologico, Ontoserver, utilizzato per la gestione delle codifiche standard.

Un esempio di flusso di integrazione tra sistemi clinici e REDCap è illustrato in Figura 4.5, che mostra il passaggio dai sistemi EHR a un data warehouse interme-

dio e successivamente a REDCap attraverso processi di estrazione, trasformazione e caricamento dei dati basati su query SQL e file intermedi. In tale contesto, l'introduzione dello standard HL7 FHIR consente di superare questo approccio, abilitando uno scambio più diretto, strutturato e interoperabile delle informazioni tra i diversi sistemi.

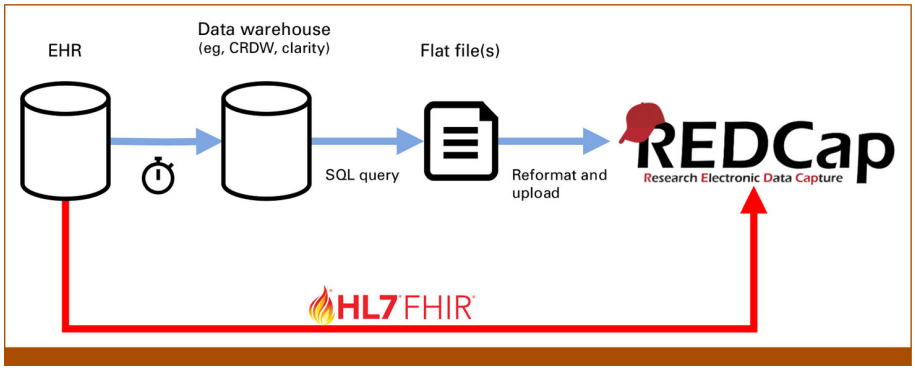


Figura 4.5: Flusso di integrazione dei dati tra sistemi clinici e REDCap.
Fonte: Extracting Electronic Health Record Neuroblastoma Treatment Data With High Fidelity Using the REDCap Clinical Data Interoperability Services Module[9]

L'implementazione di questo sistema presenta diverse sfide, tra cui la definizione di un linguaggio di trasformazione adeguato, la gestione dei dati non strutturati e il mantenimento della coerenza tra i diversi componenti coinvolti. In particolare, la trasformazione dei dati si basa su un insieme di regole formali che specificano le condizioni di attivazione, il tipo di risorsa FHIR da generare, l'identificativo della risorsa e gli attributi da valorizzare. In questo modo, i dati raccolti in REDCap vengono resi compatibili con il modello FHIR.

Una criticità rilevante riguarda la gestione dei campi di testo libero. Mentre i campi strutturati di REDCap, come menu a tendina o selezioni multiple, possono essere facilmente associati a codifiche standard, i campi testuali introducono ambiguità e difficoltà di interpretazione. Per affrontare questo problema, FHIRCap adotta una strategia che consiste nella creazione di concetti codificabili contenenti direttamente il testo inserito dall'utente, consentendo così di preservare l'informazione anche in assenza di una codifica standard.
Un ulteriore aspetto critico è rappresentato dalla sincronizzazione dei dati tra REDCap e il sistema di trasformazione. Poiché i dati e le strutture dei moduli possono variare nel tempo, è necessario garantire che tali modifiche vengano correttamente propagate. A questo scopo, FHIRCap implementa due meccanismi complementari: da un lato, un sistema di notifiche basato su endpoint, attraverso il quale REDCap

segnala le modifiche avvenute; dall'altro, un processo di interrogazione periodica delle API REDCap, finalizzato a individuare eventuali aggiornamenti.

Nel complesso, FHIRCap si configura come una soluzione efficace per l'esportazione dei dati da REDCap verso una rappresentazione FHIR standardizzata, consentendo l'integrazione con terminologie cliniche come SNOMED CT e LOINC. Tuttavia, la qualità del risultato dipende in modo significativo dalla progettazione dei moduli REDCap.

In sintesi, le soluzioni analizzate evidenziano come l'integrazione dei dati clinici non possa essere ricondotta a un unico standard o a un singolo approccio, ma richieda l'utilizzo coordinato di modelli e tecnologie differenti, ciascuno ottimizzato per specifiche finalità. In particolare, standard come openEHR e FHIR supportano rispettivamente la modellazione e l'interoperabilità dei dati, mentre modelli come OMOP e piattaforme come i2b2 sono orientati alla loro organizzazione e analisi su larga scala. In questo contesto, strumenti come REDCap e sistemi di integrazione quali FHIRCap e le pipeline ETL rappresentano elementi chiave per abilitare il passaggio tra questi diversi livelli, evidenziando come l'interoperabilità non sia solo un problema tecnico, ma un processo complesso che richiede armonizzazione semantica, strutturale e organizzativa.

Capitolo 5

Il Framework

Il framework sviluppato si inserisce in questo contesto con l'obiettivo di supportare l'integrazione tra il modello dati OMOP CDM e la piattaforma REDCap, permettendo il trasferimento automatico e strutturato di dati clinici tra i due sistemi. In particolare, consente di collegare un sistema orientato all'analisi e alla standardizzazione dei dati, come OMOP, con uno orientato alla raccolta dei dati per studi clinici, come REDCap, mantenendo coerenza tra le informazioni.

La soluzione è progettata per gestire tutto il processo di estrazione, trasformazione e caricamento dei dati attraverso una pipeline ETL configurabile. A differenza di approcci più rigidi, in cui le trasformazioni sono scritte direttamente nel codice, in questo caso le regole di mapping sono definite in file di configurazione esterni. Questi file descrivono come i dati devono essere trasformati e trasferiti, includono conversioni di formato, condizioni logiche e associazioni tra variabili. Tale impostazione rende il framework più flessibile e facilmente adattabile a diversi casi d'uso, come nuovi studi clinici o modifiche nelle variabili da gestire.

Il framework si basa su una separazione netta delle responsabilità tra i diversi componenti, secondo un'architettura modulare. In particolare, l'architettura comprende moduli specifici per l'interazione con i sistemi OMOP e REDCap, utilizzati rispettivamente come sorgente e destinazione dei dati. Il modulo OMOP è responsabile dell'estrazione dei dati clinici dal database conforme al Common Data Model, mentre il modulo REDCap si occupa della comunicazione con la piattaforma di raccolta dati, gestendo l'invio e l'aggiornamento delle informazioni tramite le API. Nel contesto di questo lavoro, vengono utilizzati due distinti progetti REDCap: uno iniziale, da cui vengono recuperati i file di configurazione, e uno di destinazione, su cui ven-

gono salvati i valori elaborati. I dettagli relativi alla loro progettazione e utilizzo verranno approfonditi nelle sezioni successive.

Il motore ETL rappresenta il nucleo logico del sistema ed è responsabile dell'elaborazione e della trasformazione dei dati secondo le regole definite in configurazione. A completare l'architettura è presente un modulo orchestratore, incaricato di coordinare l'esecuzione del processo e gestire gli input, assicurando che le diverse componenti del sistema interagiscano in modo coerente e che il flusso di dati venga eseguito correttamente dall'inizio alla fine. Una rappresentazione complessiva del flusso operativo del framework è riportata in Figura 5.1. Come illustrato nella figura, il processo viene avviato dal modulo di orchestrazione (Execute), che inizializza l'esecuzione e delega al modulo ETL la gestione della pipeline. Il sistema legge la configurazione e itera sulle variabili definite, eseguendo per ciascuna di esse un ulteriore ciclo sui pazienti inclusi nel mapping.

Per ogni paziente, il modulo ETL recupera inizialmente gli input da REDCap tramite l'apposita utility, ottenendo i valori già presenti nel sistema di destinazione, come ad esempio le date delle visite. Successivamente, utilizzando tali informazioni come riferimento, interroga il database OMOP attraverso il relativo modulo, recuperando i dati clinici associati. I dati provenienti dai due sistemi vengono quindi elaborati e integrati secondo le regole definite nella configurazione, combinando le informazioni disponibili e producendo i valori finali. Infine, i risultati vengono inviati nuovamente a REDCap, dove vengono scritti o aggiornati solo in caso di variazioni rispetto ai dati già presenti. L'intero processo è organizzato in modo iterativo e si ripete per ogni variabile e per ogni paziente, garantendo un flusso strutturato e coerente di integrazione dei dati.

Questa suddivisione consente di ottenere un'elevata modularità, in cui ciascun componente è indipendente e può essere sviluppato, testato e modificato separatamente. Inoltre, favorisce la riusabilità dei moduli, impiegabili anche in altri contesti di integrazione dati, e l'estendibilità del sistema.

Nel complesso, il framework rappresenta una soluzione flessibile e scalabile per l'integrazione di sistemi eterogenei in ambito sanitario, facilitando la gestione e l'utilizzo dei dati clinici. Al fine di implementare le funzionalità descritte, il sistema si basa su un insieme di tecnologie che supportano l'accesso ai dati, la loro elaborazione e la comunicazione tra i diversi componenti. La scelta di tali strumenti è stata guidata dalla necessità di garantire affidabilità, interoperabilità e facilità di integrazione con i sistemi coinvolti.

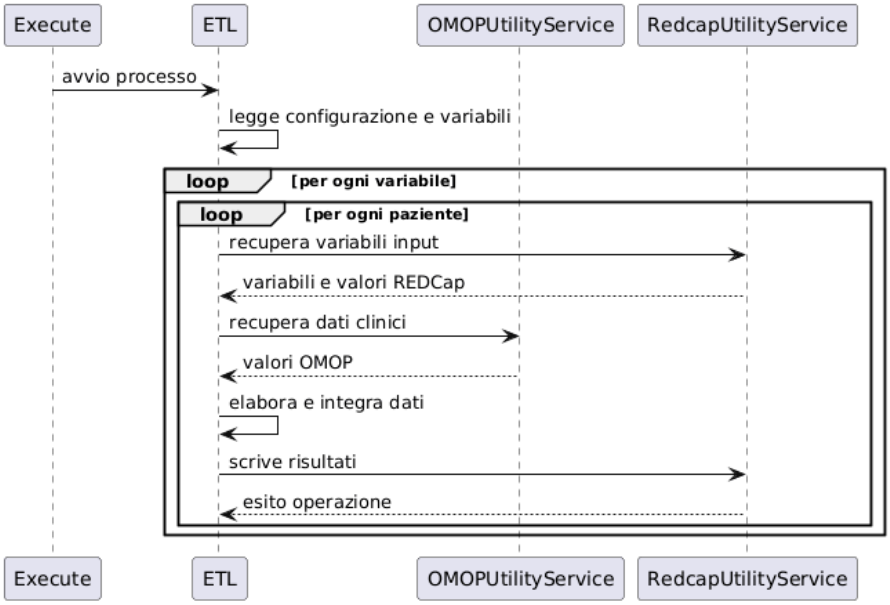


Figura 5.1: Schema semplificato del flusso di esecuzione del framework

5.1 Tecnologie di supporto

5.1.1 Sistema di gestione del database e linguaggio SQL

La gestione dei dati clinici all'interno del sistema sviluppato si basa su un database relazionale, interrogato tramite il linguaggio SQL (Structured Query Language), standard per l'interazione con basi di dati strutturate.

Nel modello relazionale le informazioni sono organizzate in tabelle composte da righe e colonne, dove ogni riga rappresenta un record e ogni colonna un attributo specifico. Le relazioni tra le tabelle sono definite attraverso chiavi primarie e chiavi esterne, che permettono di collegare tra loro le diverse entità del dominio informativo e di garantire coerenza e integrità dei dati.

Nel contesto di questo lavoro, il database ospita i dati clinici armonizzati secondo il modello OMOP Common Data Model. Le informazioni relative a pazienti, condizioni cliniche, procedure, farmaci e misurazioni sono memorizzate in tabelle distinte ma collegate tra loro secondo una struttura normalizzata che facilita la gestione e l'interrogazione dei dati.

Il linguaggio SQL consente di definire la struttura delle tabelle, inserire, aggiornare e interrogare i dati memorizzati. Le query di selezione assumono un ruolo centrale

nel flusso applicativo sviluppato, poiché permettono di estrarre sottoinsiemi specifici di informazioni sulla base di criteri clinici, temporali o concettuali.

Dal punto di vista architetturale, il database è il componente in cui i dati clinici vengono memorizzati in modo stabile e permanente. I moduli applicativi sviluppati in Java accedono a tali dati per interrogarli, elaborarli e trasferirli verso sistemi esterni.

5.1.2 Linguaggio di programmazione Java

Lo sviluppo del sistema applicativo è stato realizzato utilizzando il linguaggio di programmazione Java. Java è un linguaggio orientato agli oggetti, caratterizzato da un' ampia disponibilità di librerie e framework. La sua architettura consente di sviluppare applicazioni modulari e facilmente manutenibili, aspetti particolarmente rilevanti in contesti in cui il sistema deve interagire con basi di dati strutturate e servizi esterni.

Nel progetto sviluppato, Java è stato utilizzato per implementare la logica applicativa responsabile dell'estrazione dei dati dal database OMOP, della loro elaborazione secondo le regole definite nella configurazione delle variabili e della successiva preparazione per l'invio verso REDCap. Attraverso la definizione di classi e metodi specifici, il sistema organizza le diverse operazioni in modo strutturato, separando il livello di accesso ai dati dal livello di elaborazione logica. Questa impostazione facilita la leggibilità del codice e consente di intervenire in modo mirato su singole componenti senza compromettere l'intero sistema. L'utilizzo di Java consente inoltre una gestione efficiente delle connessioni al database e delle chiamate verso servizi esterni, garantendo stabilità e controllo degli errori durante l'esecuzione dei processi di integrazione.

Nell'architettura complessiva, Java rappresenta il livello applicativo che si colloca tra il database, in cui i dati sono memorizzati, e i sistemi esterni con cui avviene lo scambio informativo, svolgendo un ruolo centrale nei flussi di trasformazione e trasferimento dei dati clinici.

5.1.3 Framework Spring Boot

Per realizzare l'applicazione è stato utilizzato il framework Spring Boot, estensione del framework Spring, ampiamente impiegato nello sviluppo di applicazioni Java

lato backend. Spring Boot consente di configurare e avviare rapidamente un'applicazione, riducendo la complessità legata alla gestione manuale delle dipendenze e delle configurazioni di sistema e favorendo lo sviluppo di servizi backend organizzati e facilmente manutenibili.

Nel sistema sviluppato, Spring Boot è stato impiegato per organizzare l'architettura applicativa secondo una struttura a livelli, distinguendo i componenti responsabili dell'esposizione dei servizi, della logica di elaborazione e dell'accesso ai dati. In particolare, le classi annotate con @Service implementano la logica applicativa, mentre i componenti contrassegnati con @Repository gestiscono l'interazione con il database. Le entità persistenti sono invece definite tramite classi annotate con @Entity, che rappresentano la struttura dei dati e consentono la loro mappatura nel database relazionale tramite JPA/Hibernate. Questa separazione contribuisce a rendere il codice più leggibile e modulare.

Un elemento centrale del framework è il meccanismo di dependency injection, attivato tramite annotazioni come @Autowired, che consente al sistema di istanziare e collegare automaticamente i diversi componenti dell'applicazione. Ciò riduce l'accoppiamento tra le classi e migliora la flessibilità complessiva del progetto.

Spring Boot facilita inoltre l'integrazione con servizi esterni tramite l'utilizzo di client HTTP e la gestione di richieste verso API, consentendo lo scambio di dati in formato JSON. Questa funzionalità risulta particolarmente rilevante nel contesto del presente lavoro, in cui l'applicazione interagisce con API esterne, come quelle messe a disposizione da REDCap, per l'importazione automatizzata dei record.

Dal punto di vista architetturale, Spring Boot fornisce quindi l'infrastruttura che consente all'applicazione Java di operare in modo strutturato, integrando il livello di persistenza dei dati con i meccanismi di comunicazione verso sistemi esterni e garantendo coerenza nell'organizzazione complessiva del progetto software.

5.1.4 JPA e Hibernate

Per gestire l'interazione tra applicazione e database è stata utilizzata la Java Persistence API (JPA), specifica standard che definisce le modalità con cui gli oggetti Java possono essere mappati su tabelle relazionali.

JPA non è un'implementazione concreta, ma un insieme di regole e interfacce che consentono di definire la corrispondenza tra classi dell'applicazione e strutture del

database. Nel presente progetto, l'implementazione utilizzata è Hibernate[12] , uno dei framework ORM (Object-Relational Mapping) più diffusi in ambiente Java.

L'Object-Relational Mapping consente di rappresentare le tabelle del database come classi Java, nelle quali ogni oggetto corrisponde a un record e ogni attributo a una colonna della tabella. Questa impostazione permette di lavorare con oggetti e metodi del linguaggio Java, evitando la gestione diretta delle query SQL nella maggior parte delle operazioni standard. In Figura 5.2 viene illustrata la corrispondenza tra una classe Java annotata con JPA e la relativa tabella nel database.

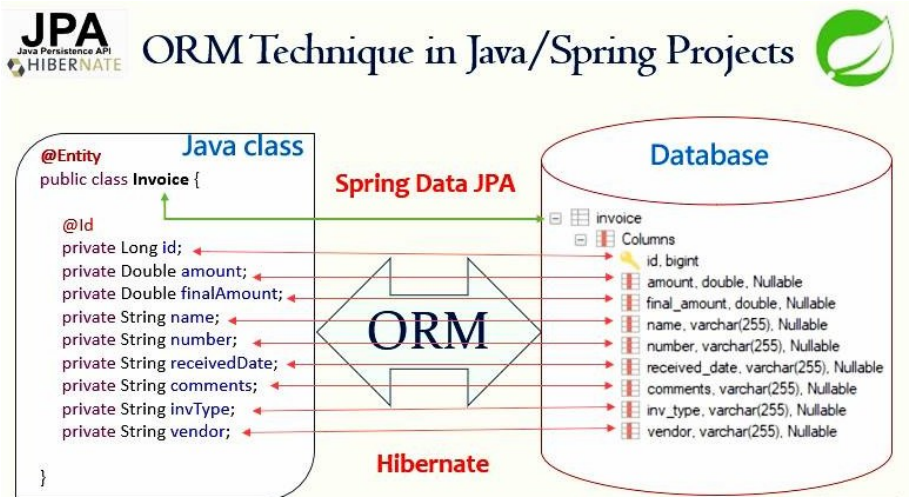


Figura 5.2: Corrispondenza tra entità Java e tabelle di database tramite ORM.
Fonte: https://javatechonline.org/

Mediante annotazioni JPA come @Entity per identificare una classe persistente, @Table per specificare la tabella di riferimento, @Id per la chiave primaria, @Column per la mappatura degli attributi e @ManyToOne o @OneToMany per definire le relazioni tra entità, le classi Java vengono collegate in modo esplicito alla struttura relazionale del database.

Nel flusso applicativo sviluppato, JPA e Hibernate svolgono un ruolo fondamentale come livello di astrazione tra l'applicazione e il database, consentendo di gestire i dati in modo strutturato senza interagire direttamente con il linguaggio SQL. In particolare, è stato utilizzato lo strumento Hibernate Tools per generare automaticamente le classi @Entity a partire dallo schema del database OMOP, evitando di definire manualmente ciascuna classe.

Questo approccio ha permesso di velocizzare significativamente la fase di sviluppo, soprattutto considerando la complessità e l'elevato numero di tabelle del modello OMOP, garantendo al contempo una piena coerenza tra la struttura del database e

la sua rappresentazione nel codice applicativo. Le entità generate riflettono infatti in modo fedele lo schema relazionale, facilitando la navigazione dei dati e la gestione delle relazioni tra le diverse tabelle.

Inoltre, l'utilizzo combinato di JPA e Hibernate consente di eseguire operazioni di accesso e manipolazione dei dati attraverso oggetti Java, riducendo la necessità di scrivere query SQL manuali. Questo semplifica la logica applicativa, migliora la leggibilità del codice e ne aumenta la manutenibilità, rendendo più agevole l'evoluzione del sistema e l'adattamento a eventuali modifiche del modello dati.

Questa integrazione permette di mantenere una chiara separazione tra logica applicativa e gestione della persistenza, garantendo al contempo coerenza con la struttura del modello OMOP e stabilità nei processi di accesso ai dati.

5.1.5 Formato di scambio dati JSON

Per lo scambio di informazioni tra l'applicazione e i sistemi esterni è stato utilizzato il formato JSON (JavaScript Object Notation), uno standard ampiamente adottato per la rappresentazione e trasmissione dei dati strutturati. JSON organizza le informazioni secondo una struttura a coppie chiave–valore, facilmente leggibile sia dalle macchine che dagli sviluppatori. Questa caratteristica lo rende particolarmente adatto allo scambio di dati tra applicazioni e alla rappresentazione di informazioni strutturate in contesti eterogenei.

Nel contesto del presente lavoro, JSON è impiegato sia come formato di comunicazione con le API di REDCap, sia come formato per la definizione dei file di configurazione del sistema. In particolare, le regole di mappatura tra variabili REDCap e concetti OMOP, i criteri di filtraggio e le impostazioni operative dell'applicazione sono descritti attraverso file di configurazione strutturati in formato JSON. Questa scelta consente di separare la logica applicativa dal livello di configurazione, rendendo il sistema più flessibile e adattabile. Eventuali modifiche alle variabili, ai filtri o ai parametri di estrazione possono essere effettuate intervenendo sui file JSON, senza necessità di modificare direttamente il codice sorgente.

Inoltre, durante il processo di trasferimento dei dati, le informazioni estratte dal database OMOP e rielaborate vengono convertite in oggetti JSON conformi alla struttura richiesta dalle API di REDCap. Analogamente, le risposte ricevute vengono deserializzate in oggetti Java per essere gestite internamente dal sistema.

L'integrazione tra Spring Boot e JSON è resa possibile dall'utilizzo della libreria Jackson, inclusa automaticamente nel framework. Essa consente di convertire in modo trasparente oggetti Java in rappresentazioni JSON e viceversa, attraverso meccanismi di serializzazione e deserializzazione automatica. Questo approccio semplifica lo sviluppo delle funzionalità di scambio dati, riducendo la necessità di gestire manualmente la costruzione delle strutture testuali e permettendo di operare direttamente su oggetti tipizzati.

L'utilizzo di JSON si colloca quindi su due livelli dell'architettura: da un lato come formato di configurazione interna del sistema, dall'altro come formato standard di scambio dati con servizi esterni. Questa doppia funzione contribuisce a garantire interoperabilità, modularità e semplicità di integrazione nei processi di trasferimento dei dati clinici.

5.2 Struttura del file di configurazione JSON

Il funzionamento del framework si basa sull'utilizzo di file di configurazione in formato JSON (vedi Appendice B), che descrivono in modo dichiarativo come i dati devono essere estratti da OMOP e trasferiti in REDCap.

La struttura dei file di configurazione riprende e generalizza quella proposta in lavori precedenti, introducendo alcune estensioni per supportare casi d'uso più complessi. In particolare, il formato è stato adattato per gestire più variabili all'interno dello stesso file, valori numerici con intervalli di riferimento e mapping verso valori categoriali (campi a scelta multipla in REDCap).

Il file è organizzato come una lista di variabili, ciascuna delle quali rappresenta un'informazione clinica da estrarre e trasformare. Ogni variabile è definita come un blocco indipendente, il che permette di aggiungere, modificare o rimuovere elementi senza impattare sul resto della configurazione.

All'interno di ogni variabile, ci sono tre sezioni principali:

- semantic
- context
- outputs

5.2.1 Sezione Semantic

La sezione semantica definisce il significato clinico della variabile:

- omop_concept_id: identifica il concetto OMOP
- omop_domain: specifica il dominio (es. Measurement)

Queste informazioni guidano l'estrazione dei dati dal database, permettono al sistema di individuare correttamente le informazioni da recuperare.

5.2.2 Sezione Context

La sezione successiva, denominata context, definisce il contesto operativo della variabile. Qui vengono specificati gli input necessari, i criteri di selezione dei dati e le regole per identificare il valore da scrivere in REDCap.

Input

Gli input rappresentano valori provenienti da REDCap, utilizzati per contestualizzare l'estrazione:

- name: nome dell'input (es. data_visita)
- ref_date_ref: riferimento alla variabile
- repeating: indica se l'input è ripetuto

Gli input sono fondamentali per collegare i dati OMOP ad un esatto evento clinico, ad esempio una specifica visita. Il fatto che un input possa essere marcato come ripetuto permette di gestire correttamente i casi in cui uno stesso paziente ha più visite.

Facts

All'interno del contesto viene poi definita la sezione dei facts, che descrive come recuperare i dati da OMOP. Ogni fatto rappresenta un insieme di istruzioni utilizzate

per selezionare una o più righe all'interno delle tabelle del modello ed è costituito da due elementi principali: un insieme di filtri, che determinano i criteri di ricerca, e un parametro refval, utilizzato per risolvere eventuali ambiguità nei risultati.

I filtri costituiscono il meccanismo attraverso cui viene costruita la query sul database e possono essere di tre tipi:

- DOMAIN, utilizzato per limitare la ricerca a uno specifico dominio OMOP. In questo caso è necessario specificare l'attributo domain_ref, che indica la variabile di riferimento da cui ottenere il dominio;
- CONCEPT, che consente di selezionare i dati associati a un determinato concetto. Richiede l'attributo concept_id_ref, che identifica la variabile da cui ricavare il concept ID; DATE, che permette di introdurre vincoli temporali. In questo caso devono essere specificati:
  - date_ref, ovvero la data di riferimento;
  - filter_type, che definisce la modalità di confronto tra le date;
  - filter_value, che indica l'ampiezza dell'intervallo temporale, espressa in giorni.

Il parametro filter_type può assumere i seguenti valori:

- FIX, che prevede un confronto esatto tra le due date;
- AFTER, che considera un intervallo temporale successivo alla data di riferimento, esteso per un numero di giorni definito da filter_value;
- BEFORE, che considera un intervallo temporale precedente alla data di riferimento, anch'esso definito da filter_value;
- BOTH, che considera un intervallo temporale simmetrico rispetto alla data di riferimento, includendo sia i giorni precedenti sia quelli successivi.

Nel caso in cui la query restituisca più risultati, il parametro refval consente di selezionare il valore più appropriato tra quelli disponibili. In particolare, può assumere i seguenti valori:

- newest, che seleziona il dato più recente;

- latest, che seleziona il dato più lontano nel tempo rispetto alla data di riferimento.

Questo meccanismo consente di gestire in modo controllato la presenza di più misurazioni per lo stesso paziente, garantendo coerenza nella selezione dei dati utilizzati nel processo di trasformazione.

5.2.3 Sezione Outputs

L'ultima sezione è quella degli outputs, che definisce come i dati devono essere trasformati e scritti in REDCap. Per ogni output viene specificata la variabile di destinazione, inclusi il nome del campo e l'eventuale evento longitudinale, e la logica di trasformazione da applicare.

La logica può essere di tipo:

- col: si scrive all'interno del campo di REDCap il contenuto della cella estratta da OMOP
- col_num: in questo caso il contenuto della colonna è un numero decimale, si può decidere di modificarne il formato e eventualmente convertirlo grazie a un fattore di conversione
- col_lookup: questa sezione del file json contiene un mapping tra valori di OMOP e valori REDCap nel caso di campi categoriali a scelta multipla

Nel complesso, il file di configurazione rappresenta il cuore del framework e definisce in modo esplicito l'intero processo di trasformazione dei dati. Grazie a questa impostazione, è possibile modificare il comportamento del sistema intervenendo unicamente sulla configurazione, senza dover modificare il codice, aumentando così flessibilità e riusabilità.

5.3 Architettura generale

L'architettura del framework è stata progettata secondo un approccio modulare, con l'obiettivo di separare le diverse responsabilità e rendere il sistema flessibile e facilmente estendibile. I diversi componenti collaborano per realizzare le fasi del processo di integrazione, mantenendo una distinzione chiara tra accesso ai dati, logiche di trasformazione e gestione del flusso operativo. Di seguito vengono descritti i principali moduli che compongono il sistema e il loro ruolo all'interno dell'architettura.

5.3.1 Utility OMOP

Il modulo dedicato a OMOP si occupa di gestire l'accesso al database strutturato secondo il Common Data Model, mettendo a disposizione i dati clinici necessari al processo. In particolare, consente di recuperare informazioni relative ai pazienti, come sesso, data di nascita, razza o altre caratteristiche demografiche, oltre a misurazioni, osservazioni ed eventi clinici registrati nel sistema. I dati vengono restituiti in una forma già elaborata. In questo modo, non è necessario che il resto dell'applicazione conosca la struttura delle tabelle o le query SQL utilizzate per l'estrazione dei dati, poiché l'accesso avviene tramite un'interfaccia uniforme che espone direttamente le informazioni necessarie.

All'interno di questo modulo è concentrata tutta la logica di accesso ai dati, incluse le query SQL necessarie per interrogare il database OMOP. Il modulo rappresenta quindi un livello intermedio tra il database e il codice applicativo in Java, fungendo da ponte tra il modello relazionale e gli oggetti utilizzati all'interno del sistema. L'utility OMOP è stata progettata per supportare sia operazioni di lettura sia di scrittura dei dati, rendendola riutilizzabile in diversi contesti di integrazione e processi ETL. Nel presente lavoro, tuttavia, essa viene impiegata esclusivamente per la fase di lettura dei dati clinici necessari al processo di estrazione. In tal modo viene garantita una separazione chiara tra il livello persistente e quello applicativo, contribuendo a rendere il sistema più modulare e facilmente manutenibile. Per facilitare questa integrazione, le entità del database sono state mappate in classi Java tramite Hibernate, adottando il paradigma dell'Object-Relational Mapping (ORM). Le principali tabelle del Common Data Model, tra cui Person, Measurement, ConditionOccurrence, DrugExposure, ProcedureOccurrence e VisitOccurrence, sono rappresentate come classi che modellano sia gli attributi sia le relazioni tra i diversi domini clinici. Questo mapping consente di tradurre automaticamente le

operazioni sul database in operazioni su oggetti, riducendo la complessità del codice e migliorando la coerenza tra il modello dati e il modello applicativo.

Le operazioni di accesso ai dati sono incapsulate all'interno di repository basati su Spring Data JPA, che forniscono un'interfaccia ad alto livello per l'interrogazione del database. Ogni entità è associata a un repository dedicato, come nel caso di MeasurementRepository, che espone metodi per il recupero dei dati tramite query personalizzate. Questo consente di definire interrogazioni mirate in base a criteri specifici, come l'identificativo del paziente, il concetto clinico associato, il tipo di misura, l'unità di misura o intervalli temporali e numerici. Nel MeasurementRepository riportato di seguito, ad esempio, sono presenti metodi come findByPersonId, che consente di recuperare tutte le misurazioni associate a un determinato paziente, findByMeasurementConceptId, utilizzato per filtrare i dati in base a uno specifico concetto OMOP, oppure findByMeasurementDatetimeBetween e findByValueAsNumberBetween, che permettono di introdurre rispettivamente vincoli temporali e numerici. Questi metodi riflettono direttamente le esigenze di filtraggio espresse nella configurazione del sistema e vengono utilizzati per implementare la logica di estrazione dei dati. Le query sono espresse in forma dichiarativa e permettono di ottenere selezioni complesse mantenendo il codice indipendente dalla sintassi SQL del database sottostante. Inoltre, l'utilizzo di annotazioni come @EntityGraph consente di controllare in modo esplicito il caricamento delle relazioni tra entità, evitando problemi di lazy loading e migliorando le prestazioni complessive delle interrogazioni. In questo modo, è possibile ottenere in un'unica operazione tutte le informazioni necessarie, inclusi i riferimenti ai concetti standard associati ai dati clinici.

Di seguito il MeasurementRepository con alcuni metodi da esempio:

```
public interface MeasurementRepository
extends JpaRepository<Measurement, Integer>{
...
 @EntityGraph(value = "Measurement.withConcepts")
 List<Measurement>
    findByConceptByMeasurementConceptId(Concept measurementConcept);

 @EntityGraph(value = "Measurement.withConcepts")
 @Query("SELECT m FROM Measurement m
    WHERE m.person.personId = :personId")
 List<Measurement> findByPersonId(@Param("personId") Integer personId);
```

```
 @EntityGraph(value = "Measurement.withConcepts")
 @Query("SELECT m FROM Measurement m
    WHERE m.measurementDatetime
    BETWEEN :start AND :end")
 List<Measurement> findByMeasurementDatetimeBetween(
         @Param("start") Timestamp start,
         @Param("end") Timestamp end);

 @EntityGraph(value = "Measurement.withConcepts")
 @Query("""SELECT m
        FROM Measurement m
        WHERE m.person.personId = :personId
          AND m.conceptByMeasurementConceptId.conceptId = :conceptId
        """)
 List<Measurement> findByPersonIdAndMeasurementConceptId(
         @Param("personId") Integer personId,
         @Param("conceptId") Integer conceptId
 );
}
```

A un livello superiore, i servizi utilizzano i repository per orchestrare le interrogazioni e combinare i risultati, producendo dati già strutturati e pronti per essere utilizzati nelle fasi successive della pipeline ETL. L'organizzazione del modulo segue quindi una struttura a livelli composta da entità, repository e servizi, che consente di separare in modo netto la rappresentazione dei dati, la logica di accesso e la logica applicativa. Tale separazione favorisce la modularità del sistema e ne semplifica sia la comprensione sia la manutenzione. Una rappresentazione più dettagliata del flusso di interrogazione dei dati clinici è riportata in Figura 5.3, che evidenzia l'interazione tra i diversi livelli del modulo.

Questo livello di astrazione risulta particolarmente utile in quanto consente di isolare completamente la logica di accesso ai dati dal resto dell'applicazione. Come mostrato in Figura 5.3, una richiesta proveniente dal livello applicativo (ETL Service) viene gestita dall'OmopUtilityService, che delega l'accesso ai dati al repository. Quest'ultimo traduce la richiesta in una query SQL eseguita sul database OMOP e restituisce i risultati sotto forma di entità, mantenendo completamente trasparente il dettaglio implementativo al resto del sistema. Di conseguenza, eventuali

modifiche alla struttura del database OMOP, all'organizzazione delle entità o alle modalità di interrogazione possono essere gestite localmente all'interno del modulo, senza influire sugli altri componenti del sistema.

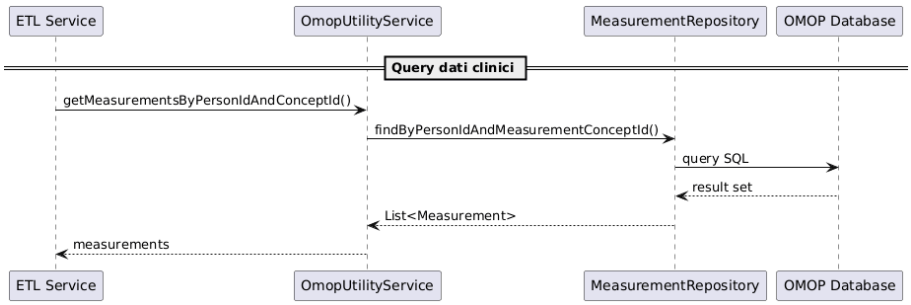


Figura 5.3: Flusso di interrogazione dei dati clinici nel modulo OMOP

5.3.2 Utility REDCap

Il modulo REDCap svolge una funzione speculare rispetto a quello OMOP, rappresenta il punto di accesso alla piattaforma di destinazione e ha il compito di gestire tutte le interazioni con le API di REDCap. Attraverso questo componente, il framework è in grado sia di leggere informazioni dai progetti associati agli studi REDCap, come metadati e dati già presenti, sia di scrivere nuovi record o aggiornare quelli esistenti sulla base dei risultati prodotti dalla pipeline ETL.

Dal punto di vista operativo, il modulo si occupa di costruire correttamente le richieste verso le API, gestendo parametri, formati e strutture richieste dalla piattaforma. Le API di REDCap si basano infatti su richieste HTTP di tipo POST, in cui il comportamento dell'operazione è determinato da un insieme di parametri che descrivono il contenuto della richiesta e il formato dei dati. In questo contesto, due operazioni fondamentali sono l'export e l'import dei record. Una rappresentazione del flusso di interazione con le API REDCap è riportata in Figura 5.4.

Come mostrato in Figura 5.4, nel caso dell'export il servizio applicativo invia una richiesta all'utility REDCap, che a sua volta delega al client la costruzione della chiamata HTTP verso le API. La richiesta viene inviata utilizzando il parametro *content=record* e i dati restituiti dalla piattaforma vengono convertiti in oggetti strutturati utilizzabili all'interno del sistema. In modo analogo, durante l'import, i record da scrivere vengono serializzati in formato JSON e inviati tramite una richiesta POST, ricevendo in risposta un esito che indica il risultato dell'operazione.

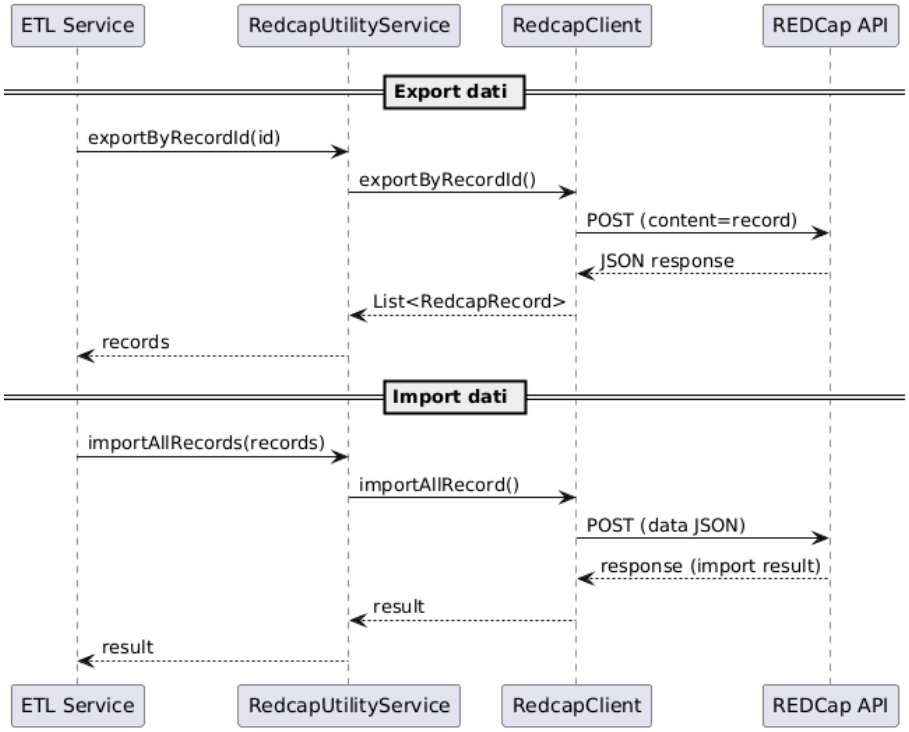


Figura 5.4: Flusso di export e import dei dati tramite API REDCap

Export

L'operazione di export consente di recuperare i dati già presenti su REDCap e viene utilizzata, ad esempio, per verificare lo stato corrente dei record o per aggiornarli. La richiesta viene costruita specificando i parametri principali:

- "content=record": specifica che si vogliono esportare i record
- "format=json": definisce il formato della risposta
- "type=flat": indica che i dati devono essere restituiti in formato tabellare, una riga per record-evento

A questi possono essere aggiunti parametri opzionali, come l'identificativo dei record da estrarre. Il risultato della chiamata è una stringa JSON che rappresenta i dati esportati. Questa viene successivamente convertita in una struttura applicativa (List<RedcapRecord>), che permette al framework di manipolare i dati in modo tipizzato.

La classe RedcapRecord rappresenta l'unità informativa di base utilizzata per modellare i dati provenienti da REDCap. Ogni istanza corrisponde a un record del

progetto ed è identificata da un recordId. Successivamente, possono essere associati ulteriori attributi, come il nome dell'evento (redcapEventName) e, nel caso di strumenti ripetuti, le informazioni relative allo strumento (redcapRepeatInstrument) e all'istanza (redcapRepeatInstance). I valori delle variabili della eCRF non sono rappresentati come attributi fissi della classe, ma sono memorizzati in una struttura dinamica (Map<String, String>), che consente di gestire in modo flessibile campi eterogenei e variabili configurabili. Questo approccio permette di adattare facilmente la struttura del record a progetti REDCap diversi, senza dover modificare il modello applicativo. Infine, il metodo toMap() consente di convertire il record in una rappresentazione chiave–valore compatibile con il formato richiesto dalle API di REDCap, facilitando le operazioni di esportazione e importazione dei dati.

Import

L'operazione di import rappresenta invece la fase finale della pipeline ETL, in cui i dati trasformati vengono scritti su REDCap. In questo caso, i record vengono prima organizzati come coppie chiave-valore (List <Map <String,String> >), dove ogni chiave corrisponde a un campo REDCap, e successivamente serializzati in formato JSON. Ogni mappa rappresenta un record REDCap, le chiavi corrispondono ai nomi dei campi REDCap e i valori rappresentano i dati da inserire. La richiesta HTTP include i seguenti parametri principali:

- "content=record": indica che si stanno importando record
- "format=json": formato dei dati inviati
- "type=flat": struttura tabellare dei dati
- "data=jsonData": payload contenente i record
- "overwriteBehavior"=overwrite: sovrascrittura dei valori esistenti
- "forceAutoNumber=false": mantiene eventuali ID già specificati
- "returnContent=count": restituisce il numero di record importati
- "returnFormat=json": formato della risposta

Queste operazioni sono completamente incapsulate all'interno dell'utility REDCap e non sono esposte agli altri moduli del framework. In questo modo, la logica

di comunicazione con le API rimane isolata, consentendo agli altri componenti di lavorare su una rappresentazione astratta dei dati senza doversi occupare dei dettagli implementativi.

Oltre alla comunicazione di base con le API, il modulo gestisce anche alcune caratteristiche specifiche di REDCap che rendono il processo di integrazione più complesso. In particolare, si occupa della gestione degli eventi longitudinali, che permettono di associare ad ogni record, quindi ad ogni paziente, un'unica anagrafica ma più visite nel tempo. Questo significa che, quando si inseriscono i dati, non è sufficiente indicare il paziente, ma è necessario specificare anche a quale visita si riferiscono. Ad esempio, se un paziente ha più visite, come una iniziale e una di follow-up, il sistema deve garantire che i dati della seconda visita non vengano salvati nella prima. Questa associazione avviene in modo automatico e permette di collegare ogni valore all'evento corretto. Un altro caso riguarda i repeat instruments, in cui uno stesso tipo di informazione può essere registrato più volte. Per esempio, se per uno stesso paziente vengono raccolte più misurazioni di laboratorio nella stessa visita, ogni valore deve essere salvato separatamente. Il modulo gestisce automaticamente queste ripetizioni, assegnando a ciascun dato la propria istanza e evitando che i valori si sovrascrivano tra loro.

Questi aspetti, se affrontati direttamente nella pipeline ETL, renderebbero il sistema più complesso e meno manutenibile. Per questo motivo, vengono completamente incapsulati all'interno dell'utility REDCap, che fornisce un'interfaccia più semplice e uniforme verso il resto del framework. In questo modo, gli altri componenti possono lavorare su una rappresentazione logica dei dati, senza doversi occupare dei dettagli operativi legati alla piattaforma.

5.3.3 Pipeline ETL

La pipeline ETL rappresenta il nucleo operativo del framework e costituisce il componente responsabile della gestione del flusso dei dati tra il sistema sorgente OMOP e la piattaforma di destinazione REDCap. Il suo obiettivo è quello di trasformare dati clinici strutturati secondo il Common Data Model in una rappresentazione compatibile con il modello logico e i vincoli imposti da REDCap.

A differenza di una semplice fase di trasferimento dati, la pipeline introduce un livello di elaborazione intermedio in cui i dati vengono selezionati, trasformati e adattati in base a regole specifiche indicate nel file JSON di configurazione.

La pipeline è progettata secondo una logica modulare e sequenziale, in cui ogni fase svolge un compito ben definito all'interno del processo complessivo. Le operazioni vengono eseguite in modo controllato e ripetibile, garantendo coerenza nei risultati e facilitando il tracciamento delle trasformazioni applicate ai dati. Come rappresentato nel diagramma di flusso in Appendice E.2, la pipeline ETL si articola in una sequenza di operazioni che coinvolgono diversi componenti del sistema, dalla lettura della configurazione fino alla scrittura dei dati su REDCap. Di seguito vengono descritte nel dettaglio le tre fasi principali che la compongono: Extract, Transform e Load.

Extract

La fase di Extract è il primo passaggio operativo della pipeline ETL e ha il compito di raccogliere tutti i dati necessari per avviare il processo di trasformazione. Nel framework sviluppato, questa fase non coincide semplicemente con una singola interrogazione al database OMOP, ma con una sequenza ordinata di operazioni che parte dalla lettura degli input in REDCap e arriva all'estrazione dei dati clinici corrispondenti dal database.

Il processo inizia dalla lettura degli input presenti in REDCap. Questi input sono valori già inseriti nel progetto, come ad esempio le date delle visite o altri campi utili, e servono per capire quali dati andare a cercare in OMOP. L'utility REDCap recupera questi valori e li organizza in una struttura dati che mantiene anche il contesto, come il paziente, l'evento e l'eventuale istanza ripetuta. Questo passaggio è cruciale perché, come enunciato precedentemente, lo stesso tipo di dato può comparire più volte per uno stesso paziente, e deve quindi essere sempre collegato alla visita corretta.

Nel caso di variabili caratterizzate da input ripetuti, come le date delle visite, i dati vengono inizialmente raccolti come una lista di record associati allo stesso campo, ciascuno identificato da una specifica istanza (repeat instance). Come mostrato nel log riportato in Appendice C.1, l'input *data_visita* risulta infatti associato a più record per lo stesso paziente, ognuno rappresentante una visita distinta. Per garantire la correttezza del processo, il sistema non elabora l'intera lista simultaneamente, ma separatamente, considerando una sola visita alla volta. In questo modo, la pipeline ETL viene eseguita separatamente per ciascuna istanza, mantenendo il corretto allineamento tra input REDCap e dati clinici estratti da OMOP.

Una volta ottenuti gli input, il sistema li utilizza come riferimento per guidare l'interrogazione e l'estrazione dei dati dal database. A questo punto viene chiamata l'utility OMOP. In base al tipo di variabile da elaborare, il modulo OMOP recupera i dati dal dominio corretto, come measurement, person o altri. La scelta del dominio dipende dalla configurazione e dal tipo di informazione richiesta: per una variabile clinica si accede alle misurazioni, mentre per dati anagrafici si utilizza la tabella dei pazienti.

I dati estratti non vengono restituiti in forma grezza, ma già organizzati in oggetti che rappresentano i fatti clinici in modo uniforme. Questo permette alla fase successiva di lavorare sempre con lo stesso tipo di struttura, senza doversi occupare delle differenze tra le varie tabelle del database. Durante tutta questa fase di estrazione, i dati recuperati da OMOP rimangono collegati agli input da cui è partita la richiesta. Questo collegamento viene mantenuto durante tutta la fase di Extract e permette di sapere sempre a quale visita e a quale paziente si riferisce ogni valore. Questo flusso di acquisizione degli input e interrogazione del database è illustrato nel diagramma di flusso in Appendice E.2, che evidenzia la sequenza delle operazioni e le interazioni tra i servizi coinvolti.

In sintesi, la fase di Extract segue un flusso chiaro: prima legge gli input da REDCap, poi usa questi input per interrogare OMOP e infine restituisce i dati in una forma strutturata e coerente. Il risultato di questa fase non è ancora pronto per essere scritto su REDCap, ma rappresenta un insieme di informazioni organizzate che devono essere interpretate e adattate secondo le regole definite nella configurazione durante la fase successiva.

Transform

La fase di Transform rappresenta il cuore della pipeline ETL e ha il compito di convertire i dati estratti da OMOP in una forma compatibile con REDCap. A differenza della fase di Extract, che si limita a recuperare i dati, in questa fase avviene l'elaborazione vera e propria, in cui i valori vengono interpretati, trasformati e adattati secondo le regole definite nel file JSON di configurazione.

Il processo prende in ingresso due elementi principali: da un lato gli input provenienti da REDCap, che definiscono il contesto (ad esempio paziente, visita e istanza), dall'altro i dati clinici estratti da OMOP, già organizzati in una struttura uniforme. La fase di Transform combina queste due informazioni per produrre i valori finali da

scrivere nella piattaforma di destinazione. L'intero comportamento della trasformazione è guidato dal file JSON di configurazione. Per ogni variabile, il sistema legge la sezione outputs e applica le logiche definite per ciascun campo di destinazione. Il codice non contiene regole specifiche per ogni variabile, ma esegue in modo generico le istruzioni descritte nella configurazione.

Il primo passaggio consiste nell'identificare, per ogni variabile, il dato di partenza. Questo avviene tramite riferimenti ai facts definiti nella configurazione, ad esempio fact_1.ValueAsNumber oppure fact_1.RangeLow. In questo modo, durante questa fase, non si accede direttamente al database, ma vengono utilizzati i dati già estratti e selezionati nella fase precedente. Prima della trasformazione vera e propria, viene inoltre applicata una fase di filtraggio e selezione del dato clinico secondo quanto indicato nel file JSON (Appendice B). Come mostrato nel log riportato in Appendice C.2, il sistema parte da un insieme di eventi clinici (più misurazioni disponibili per uno stesso paziente), applica i filtri definiti nella configurazione, in particolare il filtro temporale, e seleziona infine il valore. Nel caso mostrato, a partire da 10 eventi iniziali, il filtro temporale riduce il risultato a una singola osservazione, che viene poi scelta come valore finale da utilizzare nella trasformazione. Una volta individuato il valore di input, il sistema applica la logica di trasformazione specificata. Le logiche supportate sono diverse e permettono di coprire i principali casi d'uso. Il caso più semplice è quello dei valori numerici, gestito tramite la logica col_num. In questo caso, il valore estratto da OMOP viene convertito nel formato richiesto da REDCap. È possibile, ad esempio, definire il separatore decimale oppure applicare un fattore di conversione. Come evidenziato nel log (Appendice C.2), il valore numerico 4.9 viene convertito nel formato 4,9, in accordo con le specifiche della configurazione.

Un caso più articolato riguarda la gestione dei range, cioè dei valori minimi e massimi associati a una misurazione. OMOP mette a disposizione campi come RangeLow e RangeHigh, che possono essere utilizzati per valorizzare campi distinti in REDCap. La fase di Transform consente quindi di generare più valori di output a partire dallo stesso fatto clinico, mantenendo la coerenza tra le informazioni derivate. Un'ulteriore tipologia di trasformazione riguarda le lookup, gestite tramite la logica col_lookup. In questo caso, il valore estratto da OMOP non viene utilizzato direttamente, ma convertito in un altro valore secondo una mappatura definita nella configurazione. Questo è particolarmente utile per i campi categoriali di REDCap, come quelli a scelta multipla o a tendina, in cui i valori devono essere codificati secondo un insieme finito di opzioni. Un esempio di configurazione delle lookup è riportato nel file JSON in Appendice B, dove sono definite le corrispondenze tra i

valori di input e quelli di output. La loro applicazione durante l'esecuzione è invece documentata nei log (Appendice C.2). Durante la trasformazione, il sistema mantiene sempre il collegamento con il contesto originale, cioè paziente, evento e istanza. Questo garantisce che ogni valore venga associato correttamente alla visita di riferimento e, se necessario, alla corretta ripetizione.

Dal punto di vista implementativo, la fase di trasformazione è progettata per essere generica e riutilizzabile. Il sistema non contiene logiche specifiche per determinate variabili ma interpreta dinamicamente il contenuto del file di configurazione. Questo consente di aggiungere nuove variabili o modificare quelle esistenti senza intervenire sul codice. Un aspetto importante riguarda la gestione dei casi in cui i dati non sono disponibili o sono già presenti in REDCap. In particolare, se un valore è già stato inserito, il sistema evita di sovrascriverlo, se invece non viene trovato alcun valore in fase di estrazione, il campo viene lasciato vuoto. Al termine di questa fase, i dati non sono più espressi secondo il modello OMOP, ma sono stati completamente trasformati nel formato richiesto da REDCap. Ogni valore è associato a un campo specifico, a un evento e a un contesto ben definito. Come mostrato nel diagramma di flusso in Appendice E.2, la fase di trasformazione si inserisce in un processo iterativo che, per ogni variabile e per ogni paziente, applica le regole definite nella configurazione per produrre i valori finali. In sintesi, la fase di trasformazione ha il compito di tradurre i dati da un modello all'altro, applicando regole configurabili e mantenendo coerenza con il contesto clinico. Il risultato è un insieme di dati pronti per essere inseriti nella piattaforma di destinazione, che verranno gestiti nella fase successiva di Load.

Load

La fase di Load rappresenta l'ultimo passaggio della pipeline ETL e ha il compito di trasferire i dati trasformati nella piattaforma REDCap. In questa fase, i valori prodotti dalla fase precedente vengono organizzati nella struttura richiesta e inviati tramite le API, completando il processo di integrazione.

Il processo prende in ingresso i dati già trasformati, che sono stati costruiti in modo coerente con il modello di REDCap. Ogni valore è infatti già associato a un campo specifico, a un paziente e a un evento, e non richiede ulteriori elaborazioni logiche. La fase di Load si occupa quindi principalmente della costruzione dei record e della loro scrittura. Per ogni insieme di dati, il sistema crea un record REDCap rispettando la struttura della piattaforma. Questo significa che vengono valorizzati correttamente

gli identificativi del paziente, l'evento longitudinale e, se presente, l'eventuale istanza ripetuta. In questo modo, i dati vengono inseriti nella posizione corretta, evitando errori di associazione.

Una volta costruiti i record, il modulo REDCap si occupa di inviarli tramite le API. Il sistema prepara la richiesta nel formato richiesto dalla piattaforma e gestisce la comunicazione, assicurando che i dati vengano trasmessi correttamente. Durante questa fase, i dati non vengono modificati, ma semplicemente trasferiti. Inoltre, viene rispettato il comportamento già definito nella Transform: i valori già presenti non vengono sovrascritti e i campi senza dati restano vuoti. L'intero processo, dalla lettura degli input fino al caricamento finale dei dati, è riassunto nel diagramma di flusso in Appendice E.2.

In sintesi, la fase di Load si occupa di completare il processo ETL trasferendo i dati in REDCap in modo coerente e strutturato. Dopo l'estrazione e la trasformazione, questa fase garantisce che le informazioni vengano effettivamente salvate nella piattaforma, rendendole disponibili per l'utilizzo nel contesto dello studio clinico.

5.3.4 Progetto Execute

Come illustrato nel diagramma di flusso riportato in Appendice E.1, il progetto Execute è il componente responsabile del coordinamento dell'intero framework. Mentre i moduli dedicati a OMOP e REDCap gestiscono rispettivamente l'accesso ai sistemi esterni e la pipeline ETL implementa le logiche di estrazione, trasformazione e caricamento dei dati, questo modulo ha il compito di orchestrare l'intero flusso operativo e avviarne l'esecuzione. La sua implementazione consente di disaccoppiare la logica di orchestrazione dalle componenti applicative, facilitando l'evoluzione e l'estensione del sistema. In prospettiva, questo livello rappresenta anche il punto ideale per integrare meccanismi di scheduling, permettendo l'esecuzione automatica e pianificata dei processi di trasferimento dei dati.

Dal punto di vista architetturale, Execute è il modulo che governa il flusso complessivo. Non contiene la logica di accesso al database OMOP, né quella di comunicazione con REDCap, e non implementa direttamente le regole di trasformazione dei dati. Il suo ruolo è invece quello di leggere la configurazione, interpretarla, raccogliere gli input necessari, richiamare i moduli corretti e garantire che le diverse fasi vengano eseguite nella corretta sequenza. In questo senso, costituisce il punto di ingresso dell'intera pipeline.

L'elemento centrale del funzionamento di Execute è il progetto REDCap di integrazione. A differenza del REDCap di destinazione, che ospita i dati clinici dello studio, questo contiene tutte le informazioni necessarie per configurare ed eseguire il processo. In particolare:

- il token API per l'accesso al progetto REDCap di destinazione;
- il file JSON di configurazione della pipeline;
- il file di mapping degli identificativi dei pazienti.

Queste informazioni sono organizzate all'interno di un form dedicato che funge da contenitore della configurazione. In questo modo, l'intero comportamento del sistema può essere gestito direttamente tramite l'interfaccia di REDCap, senza dover intervenire sul codice. Il REDCap di integrazione svolge un ruolo diverso rispetto al REDCap di destinazione dei dati clinici. Mentre quest'ultimo contiene i dati raccolti nello studio, il progetto di integrazione ha una funzione puramente operativa.

All'avvio, Execute recupera queste informazioni interrogando il REDCap di integrazione tramite il modulo REDCap Utility. Il file JSON viene deserializzato per costruire dinamicamente la struttura della pipeline, mentre il file di mapping consente di stabilire la corrispondenza tra gli identificativi dei pazienti nei due sistemi. Questo approccio rende il sistema completamente configurabile: la logica di esecuzione non è fissata staticamente nel codice, ma dipende dai contenuti forniti in fase di configurazione. Particolare rilevanza assume proprio la gestione del mapping tra pazienti, necessaria per garantire la corretta associazione dei dati tra OMOP e REDCap, che adottano schemi di identificazione differenti. Il mapping, fornito tramite file CSV e utilizzato come lookup table, viene caricato e interpretato da Execute all'inizio del processo e utilizzato lungo tutta la pipeline per tradurre gli identificativi, assicurando la coerenza e l'integrità del trasferimento dei dati. Nel dettaglio, la sequenza delle operazioni, dalla fase di download della configurazione fino all'avvio del processo ETL, è rappresentata nel diagramma di flusso in Appendice E.1, che evidenzia l'interazione tra i diversi componenti del sistema.

Nel complesso, il progetto Execute consente di separare in modo netto la definizione del processo dalla sua esecuzione, contribuendo a rendere il framework flessibile, riutilizzabile e facilmente adattabile a contesti applicativi diversi.

Capitolo 6

Test e risultati

La fase di test ha l'obiettivo di validare il corretto funzionamento del framework sviluppato per il trasferimento automatico di dati clinici da un database conforme al modello OMOP a un progetto REDCap. In particolare, sono stati verificati la correttezza del mapping tra pazienti, la coerenza dei dati trasferiti e la capacità del sistema di gestire configurazioni dinamiche attraverso file JSON. L'analisi si è concentrata in particolare sui dati di laboratorio, scelti come caso di studio principale del progetto. Le misurazioni di laboratorio sono caratterizzate da una componente temporale rilevante, dalla presenza di valori numerici e dalla possibilità di più rilevazioni per lo stesso paziente. I test hanno quindi verificato la corretta applicazione dei filtri temporali, la selezione del valore appropriato in presenza di misurazioni multiple e la coerenza tra i concetti OMOP e le variabili REDCap corrispondenti.

A tal fine sono stati utilizzati dati sintetici generati tramite Synthea, opportunamente strutturati secondo il modello OMOP, e due distinti progetti REDCap: uno dedicato alla configurazione del processo ETL e uno destinato alla raccolta dei dati clinici risultanti.

6.1 Studio REDCap di integrazione

Come indicato precedentemente, per consentire un'elevata flessibilità e riusabilità del sistema è stato progettato uno specifico studio REDCap dedicato alla configurazione del processo ETL. Questo studio rappresenta il punto di ingresso per tutti i parametri necessari all'esecuzione del framework.

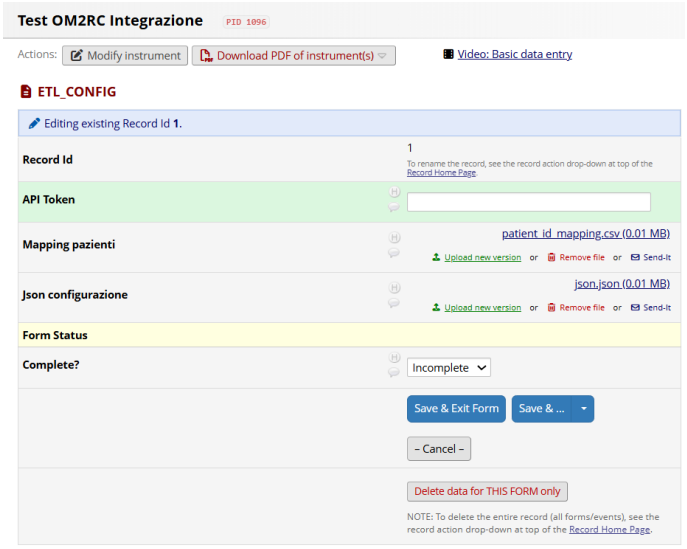


Figura 6.1: Studio REDCap di integrazione

All'interno dello studio sono stati definiti tre elementi principali:

- **API Token**: necessario per l'autenticazione e la comunicazione con lo studio REDCap target;
- **file di mapping dei pazienti**: definisce la corrispondenza tra identificativi OMOP e record REDCap;
- **file JSON di configurazione**: contiene le regole di estrazione, trasformazione e caricamento dei dati.

Il mapping dei pazienti è stato implementato tramite un file CSV, in cui ogni riga rappresenta una corrispondenza tra un identificativo OMOP e un record REDCap.

redcap_id	omop_id
1	4
2	6
3	7
4	9

Figura 6.2: CSV con mapping dei pazienti

Il file JSON di configurazione (vedi Appendice B) rappresenta invece il cuore logico del sistema: esso definisce quali variabili estrarre da OMOP, i filtri da applicare (ad esempio per dominio o intervallo temporale) e le modalità di scrittura su REDCap. In questo modo, il comportamento del framework può essere modificato senza intervenire sul codice, ma semplicemente aggiornando la configurazione.

6.2 Studio REDCap target

Il secondo progetto REDCap è stato progettato come ambiente di destinazione dei dati clinici elaborati dal framework. A differenza dello studio di configurazione, che ha una funzione puramente parametrica, questo studio rappresenta il livello applicativo finale, ovvero il punto in cui i dati estratti da OMOP vengono resi disponibili in forma strutturata e interpretabile. Lo studio è stato configurato secondo un modello longitudinale basato su eventi, in cui ciascun paziente (identificato da un record ID) può essere associato a una o più visite cliniche. In particolare, è stato definito un evento denominato visita, implementato come strumento ripetibile (repeating instrument), al fine di rappresentare correttamente la natura temporale delle osservazioni cliniche. Questa scelta progettuale consente di modellare sequenze di misurazioni nel tempo per lo stesso paziente, mantenere la coerenza con la struttura paziente-centrica del modello OMOP e supportare eventuali estensioni future dello studio (ad esempio follow-up o più eventi clinici).

Come mostrato in figura 6.3, all'interno di ciascuna istanza dell'evento visita viene registrata la data della visita, utilizzata come riferimento temporale per tutte le osservazioni associate. Lo studio include tre variabili cliniche selezionate per i test: albumina, creatinina e calcio. Per ogni variabile è stata adottata una rappresentazione strutturata che separa:

- valore osservato (misura reale estratta da OMOP);
- valore minimo di riferimento;
- valore massimo di riferimento.

Ad esempio, per la variabile albumina sono presenti i seguenti campi:

- albumina: valore misurato;
- albumina_min: limite inferiore del range;
- albumina_max: limite superiore del range.

Questo approccio consente di arricchire il dato clinico grezzo con informazioni interpretative, rendendo possibile una valutazione immediata dello stato del paziente rispetto ai valori di normalità. Le variabili sono state implementate utilizzando diverse tipologie di campo REDCap, in funzione della natura del dato:

- campi numerici per i valori osservati e per i range di albumina e creatinina;
- campi a selezione (dropdown) per i valori di riferimento del calcio.

La scelta di utilizzare campi dropdown per il calcio è legata alla presenza di una logica di lookup predefinita nel file di configurazione, che associa valori discreti a specifici range clinici. Questo consente di standardizzare l'inserimento dei dati ed evitare ambiguità o errori dovuti all'inserimento manuale.

Inoltre, tutti i campi sono stati configurati per essere compatibili con l'import automatico tramite API, rispettando i vincoli richiesti da REDCap in termini di naming e struttura dei record.

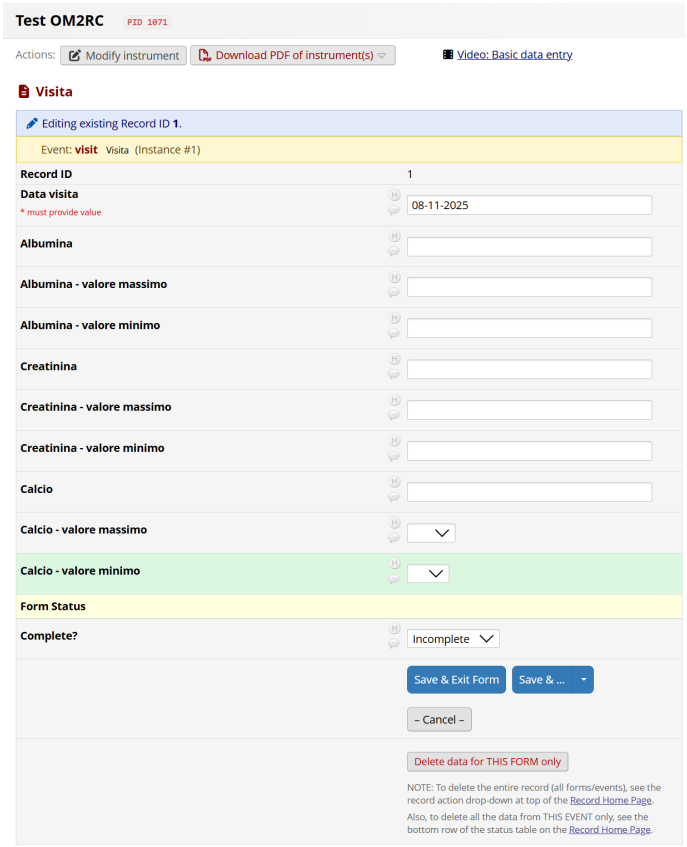


Figura 6.3: Struttura dell'evento visita dello studio REDCap target

6.3 Database OMOP CDM Synthea

Come sorgente dei dati clinici è stato utilizzato un database conforme al modello OMOP Common Data Model (CDM), popolato tramite lo strumento open-source

Synthea, un simulatore sviluppato dal MITRE che genera dati sanitari sintetici a partire da modelli di malattia e percorsi clinici realistici. I dati prodotti seguono standard interoperabili e possono essere facilmente convertiti in diversi formati, tra cui OMOP CDM. Synthea consente la generazione di dati clinici sintetici realistici, riproducendo il comportamento di pazienti nel tempo senza includere informazioni reali. Questo approccio permette di effettuare test in un contesto controllato, evitando problematiche legate alla privacy e garantendo al contempo una buona rappresentatività dei dati clinici.

Per l'esplorazione e la manipolazione del database è stato utilizzato DBeaver, un ambiente di sviluppo SQL che permette di eseguire query, visualizzare le tabelle e gestire in modo efficiente database relazionali. L'utilizzo di questo strumento ha facilitato le operazioni di analisi dei dati e di aggiornamento dei valori necessari ai test.

All'interno del database OMOP, le informazioni utilizzate nel processo di validazione provengono principalmente dalla tabella measurement, che contiene le osservazioni quantitative associate ai pazienti (ad esempio esami di laboratorio). Questa tabella rappresenta il punto di partenza per l'estrazione dei dati da trasferire verso REDCap.

6.3.1 Arricchimento dei dati

Per rendere i dati clinici maggiormente interpretabili, è stato necessario arricchire le misurazioni con i relativi valori di riferimento (range_low e range_high). A tal fine, sono state utilizzate due diverse strategie, a seconda della variabile considerata. Albumina: range dinamici basati sul valore osservato

Range dinamici basati sul valore

Per le variabili albumina (measurement_concept_id = 3024561) e creatinina (measurement_concept_id = 3051825) , è stata adottata una strategia basata su una variazione percentuale del valore osservato.

In particolare, il valore minimo è stato definito come il 90% del valore misurato e il valore massimo come il 110%.

La query utilizzata per arricchire in database è la seguente:

```
UPDATE measurement
```

```
SET
    range_low = value_as_number * 0.9,
    range_high = value_as_number * 1.1
WHERE measurement_concept_id = 3024561;
```

Questo approccio consente di generare range personalizzati per ciascuna osservazione, simulando un intervallo di tolleranza attorno al valore misurato.

Range dipendenti dal genere

Per la variabile calcio (measurement_concept_id = 3032503), i valori di riferimento sono stati assegnati in funzione del sesso del paziente, sfruttando le informazioni presenti nella tabella person.

In particolare:

- per le pazienti di sesso femminile (gender_concept_id = 8532) è stato definito un range pari a 6–9;
- per i pazienti di sesso maschile (gender_concept_id = 8507) è stato definito un range pari a 7–10.

L'aggiornamento è stato effettuato tramite la seguente query SQL:

```
UPDATE measurement
SET
    range_low = CASE
        WHEN person_id IN (SELECT person_id FROM person
        WHERE gender_concept_id = 8532) THEN 6
        WHEN person_id IN (SELECT person_id FROM person
        WHERE gender_concept_id = 8507) THEN 7
    END,
    range_high = CASE
        WHEN person_id IN (SELECT person_id FROM person
        WHERE gender_concept_id = 8532) THEN 9
        WHEN person_id IN (SELECT person_id FROM person
        WHERE gender_concept_id = 8507) THEN 10
    END
WHERE measurement_concept_id = 3032503;
```

Questa logica consente di introdurre una differenziazione clinicamente significativa dei valori di riferimento, rendendo il dataset più aderente a scenari reali.

L'introduzione dei valori di riferimento direttamente nel database OMOP ha permesso di semplificare le fasi successive della pipeline ETL. In particolare, durante l'estrazione dei dati, il framework è in grado di recuperare sia il valore osservato sia i relativi limiti di normalità, evitando la necessità di calcoli aggiuntivi in fase di trasformazione.

Questo arricchimento a monte ha contribuito a rendere il flusso complessivo più efficiente e coerente, facilitando il trasferimento dei dati verso REDCap e la loro successiva interpretazione.

6.4 Esecuzione e risultati

L'esecuzione del processo ETL avviene tramite l'avvio dell'applicazione sviluppata in ambiente Spring Boot, che funge da componente di orchestrazione dell'intero framework. In questo contesto, i moduli dedicati all'accesso ai dati OMOP e alla comunicazione con REDCap, insieme alla logica della pipeline ETL, sono stati sviluppati come componenti indipendenti e integrati sotto forma di librerie JAR. Una volta avviata, l'applicazione carica automaticamente i parametri di configurazione definiti nello studio REDCap di integrazione. Sulla base di queste informazioni, il sistema avvia il flusso di elaborazione senza necessità di ulteriori interventi manuali.

La gestione delle dipendenze tra i diversi moduli è affidata a Maven, uno strumento di build e gestione dei progetti ampiamente utilizzato in ambiente Java. Maven consente di definire in modo dichiarativo le librerie necessarie all'applicazione, automatizzando il processo di risoluzione, download e integrazione delle dipendenze. In particolare, ogni modulo del sistema è dotato di un file di configurazione pom.xml, nel quale sono specificate le librerie richieste, tra cui i componenti Spring per la gestione dell'applicazione, le dipendenze per l'accesso al database OMOP e le librerie dedicate all'interazione con REDCap. Il file pom.xml relativo al modulo ETL (riportato in Appendice D) documenta in modo esplicito le dipendenze utilizzate e la struttura del progetto, evidenziando l'integrazione tra componenti sviluppati internamente e librerie esterne.

La logica di esecuzione è strutturata in modo gerarchico e riflette direttamente l'organizzazione definita nel file JSON. Il sistema opera inizialmente a livello di variabile,

analizzando una alla volta le entità cliniche specificate nella configurazione e definite come main. Per ciascuna variabile vengono interpretati i parametri semantici, come il concept ID e il dominio di appartenenza, e vengono applicati i filtri necessari per l'estrazione dei dati dal database OMOP. Una volta definita la logica di estrazione per una determinata variabile, il sistema procede iterando sui pazienti inclusi nel file di mapping, applicando la stessa configurazione a ciascun soggetto in modo coerente e indipendente. Nel contesto dei test effettuati è stato applicato un vincolo temporale che limita l'estrazione dei dati a un intervallo di 30 giorni rispetto alla data di riferimento della visita. Questo filtro consente di selezionare esclusivamente le osservazioni clinicamente più rilevanti, evitando l'inclusione di dati troppo distanti nel tempo che potrebbero compromettere la coerenza dell'analisi. La finestra temporale viene definita all'interno del file JSON per i test (riportato in Appendice B) e viene applicata automaticamente dal sistema durante la fase di interrogazione del database OMOP.

Al termine dell'estrazione e della trasformazione, i dati vengono trasferiti allo studio REDCap target, mantenendo la struttura definita e garantendo la corretta associazione tra paziente, evento e variabile. I risultati ottenuti, visibili in figura 6.4, mostrano un comportamento coerente con le aspettative: il sistema è in grado di gestire correttamente il flusso variabile per variabile e paziente per paziente, applicando i filtri temporali definiti e producendo dati strutturati pronti per l'analisi all'interno dello studio REDCap.

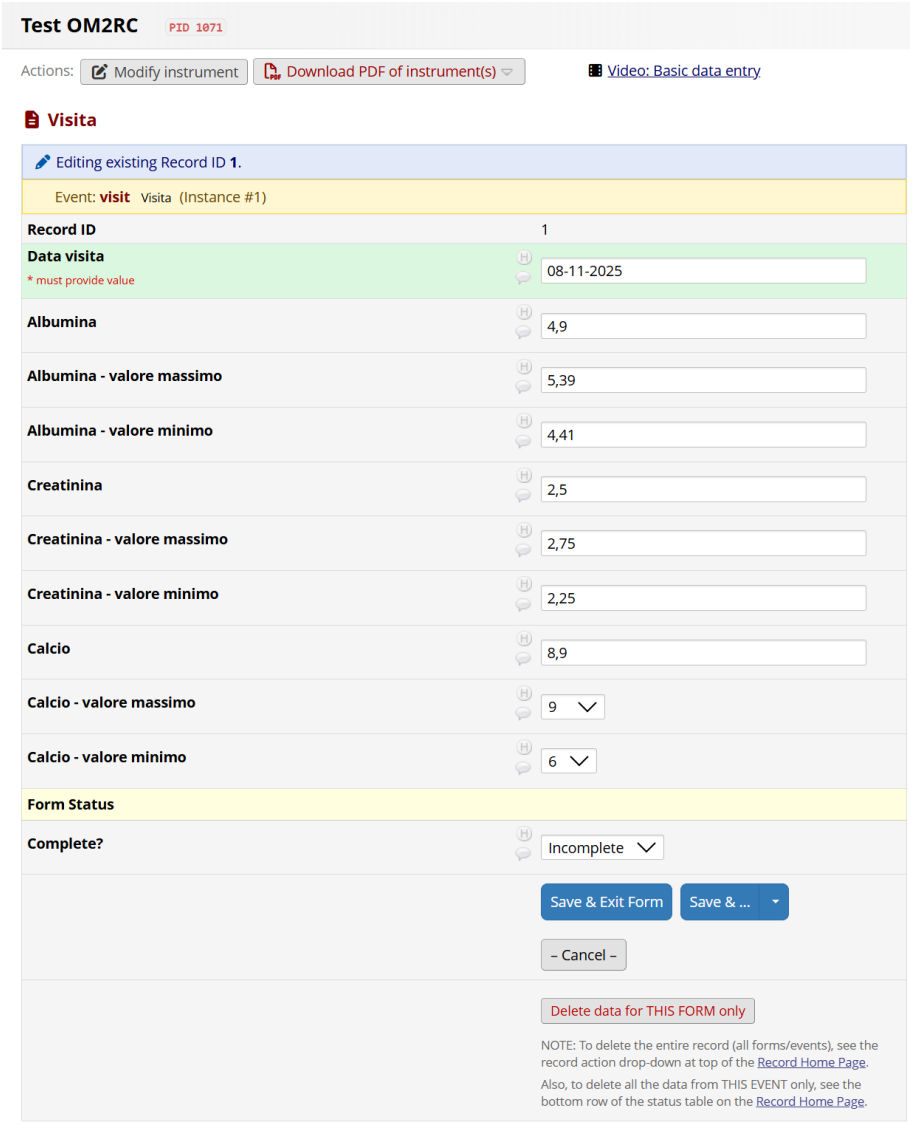


Figura 6.4: Studio REDCap target dopo l'esecuzione del trasferimento

In conclusione, la fase di esecuzione ha confermato la solidità del framework sia dal punto di vista architetturale sia operativo, evidenziando la capacità del sistema di integrare configurazioni dinamiche, gestire dati clinici strutturati e garantire un flusso di trasferimento affidabile tra sistemi eterogenei. Questi risultati costituiscono una base solida per l'estensione del framework a scenari più complessi e per il suo utilizzo in contesti reali di ricerca clinica e sanità digitale.

Capitolo 7

Sviluppi futuri

Tra i possibili sviluppi futuri, un'estensione rilevante riguarda l'introduzione di un meccanismo di attivazione del processo ETL direttamente da REDCap. In particolare, sarebbe possibile implementare delle chiamate REST che consentano di avviare l'esecuzione del framework a partire dalla piattaforma di raccolta dati. Questo approccio consentirebbe, ad esempio, di attivare automaticamente il trasferimento dei dati in seguito alla creazione o all'aggiornamento di un record, rendendo il processo più dinamico e riducendo la necessità di interventi manuali.

Un ulteriore sviluppo potrebbe riguardare l'estensione del framework a un numero maggiore di domini OMOP. Nel lavoro svolto l'attenzione è stata focalizzata principalmente sui dati di laboratorio (measurements), ma l'architettura progettata consentirebbe di generalizzare il processo anche ad altri tipi di informazioni, come condizioni cliniche, procedure o esposizioni farmacologiche, ampliando così le potenzialità applicative del sistema.

Un altro aspetto particolarmente rilevante riguarda il miglioramento dei meccanismi di validazione e controllo qualità dei dati. L'introduzione di verifiche più avanzate, sia in fase di estrazione sia durante il caricamento, permetterebbe di individuare eventuali inconsistenze o anomalie, aumentando l'affidabilità complessiva del processo.

Infine, ulteriori sviluppi potrebbero riguardare l'ottimizzazione delle prestazioni in presenza di dataset di grandi dimensioni e la realizzazione di strumenti a supporto della configurazione, come interfacce grafiche per la definizione dei file JSON o per la gestione delle lookup table. Questo renderebbe il sistema più accessibile anche a utenti non tecnici, facilitandone l'impiego in contesti reali di ricerca clinica.

Capitolo 8

Conclusioni

Il presente lavoro di tesi ha affrontato il tema dell'interoperabilità e dell'integrazione dei dati clinici, attraverso la progettazione e lo sviluppo di un framework software per il trasferimento automatico di informazioni da un database conforme al modello OMOP CDM a un sistema REDCap.

L'obiettivo è stato quello di realizzare una pipeline ETL flessibile e configurabile, in grado di ridurre l'inserimento manuale dei dati, migliorare la qualità dell'informazione e supportare attività di ricerca clinica basate su dati strutturati. A tal fine è stato sviluppato un sistema modulare, in cui le componenti di estrazione, trasformazione e caricamento sono chiaramente separate, e il comportamento del framework è definito tramite file JSON, senza necessità di modifiche al codice.

L'approccio adottato si basa sull'integrazione tra sistemi eterogenei: da un lato il database OMOP, che fornisce una base dati standardizzata e semanticamente coerente, dall'altro REDCap, utilizzato per la raccolta e la gestione dei dati clinici. Il collegamento tra i due sistemi è stato realizzato attraverso un meccanismo di mapping dei pazienti e l'utilizzo delle API di REDCap.

La fase di test, condotta su dati sintetici generati tramite Synthea, ha permesso di validare il corretto funzionamento del framework in un contesto realistico. I risultati hanno evidenziato la capacità del sistema di gestire configurazioni complesse, applicare filtri temporali e trasferire correttamente i dati nello studio REDCap di destinazione. Un aspetto rilevante è rappresentato dall'arricchimento dei dati clinici tramite l'introduzione di valori di riferimento, sia dinamici sia fissi, basati su caratteristiche del paziente, a conferma della flessibilità del sistema.

Nel complesso, il framework sviluppato si è dimostrato robusto, scalabile e facilmente estendibile. La possibilità di configurare il comportamento tramite file esterni rappresenta un vantaggio significativo in termini di manutenzione e adattabilità a nuovi scenari applicativi, mentre l'adozione dello standard OMOP favorisce l'interoperabilità con altri sistemi.

In conclusione, il lavoro svolto dimostra come sia possibile realizzare una soluzione efficace per il trasferimento dei dati clinici, contribuendo a migliorare l'integrazione tra sistemi e a supportare l'utilizzo secondario dei dati in ambito sanitario.

In questo contesto, i dati clinici rappresentano una risorsa di straordinario valore: se opportunamente strutturati e integrati, essi possono trasformarsi da semplice registrazione di eventi sanitari a strumento attivo di conoscenza. Il loro valore non risiede soltanto nella quantità, ma soprattutto nella possibilità di essere messi in relazione, interpretati e riutilizzati in modo consapevole.

Il contributo di questo lavoro si inserisce proprio in questa prospettiva, evidenziando come soluzioni tecnologiche adeguate possano abilitare nuovi scenari di analisi e supportare un utilizzo più efficace e significativo dei dati. In un contesto in cui la sanità è sempre più orientata al dato, la capacità di valorizzare queste informazioni rappresenta una sfida centrale e una grande opportunità per il futuro della ricerca e dell'innovazione in ambito clinico.

Bibliografia

[1] P. A. Harris, R. Taylor, R. Thielke, J. Payne, N. Gonzalez, J. G. Conde. *Research electronic data capture (REDCap) – A metadata-driven methodology and workflow process for providing translational research informatics support*. Journal of Biomedical Informatics, 2009, Volume 42, Numero 2, Pagine 377- 381.

[2] Emanuele Girani, Matteo Gabetta, Anna Alloni, Morena Stuppia, Lucia Sacchi, Nicola Barbarini. *Automatic Data Transfer from OMOP-CDM to REDCap: A Semantically-Enriched Framework*. Department of Electrical, Computer and Biomedical Engineering, University of Pavia, Italy, 2021. Report Tecnico Interno.

[3] Michael J. Gurley, Jeremy Warner, Yulia Bushmanova, Firas Wehbe. *REDCap2OMOP: A platform for ETLing REDCap projects into the OMOP CDM*. Northwestern University, Illinois, USA, 2021. Report Tecnico Interno.

[4] Salvatore G. Volpe, Karthik Natarajan. *REDHot OMOP: Facilitating Semantic Interoperability in REDCap with FHIR and the OMOP CDM*. Department of Biomedical Informatics, Columbia University, New York, USA, 2021. Report Tecnico Interno.

[5] Matteo Gabetta, Marco Mirabelli, Catherine Klersy, Valeria Musella, Gianpiero Rizzo, Paolo Pedrazzoli, Nicola Barbarini, Riccardo Bellazzi, Cristiana Larizza. *An Extension of the i2b2 Data Warehouse to Support REDCap Dynamic Data Pull*. Department of Electrical, Computer and Biomedical Engineering, University of Pavia, Italy, 2019. Report Tecnico Interno.

[6] Eugenia Rinaldi, Sylvia Thun. *From OpenEHR to FHIR and OMOP Data Model for Microbiology Findings*. Charité, University of Berlin, Germany, 2021. Report Tecnico Interno.

[7] Alejandro Metke-Jimenez, David Hansen. *FHIRCap: Transforming REDCap forms into FHIR resources*. The Australian eHealth Research Centre, CSIRO, Herston, Queensland, Australia, 2019. Report Tecnico Interno.

[8] A.C. Cheng, S. N. Duda, R. Taylor, F. Delacqua, A. A. Lewis, T. Bosler, K. B. Johnson, P. A. Harris. *REDCap on FHIR: Clinical Data Interoperability Services*. Vanderbilt University Medical Center, Nashville, Tennessee, USA; UT Southwestern Medical Center, Dallas, Texas, USA, 2021. Report Tecnico Interno.

[9] B. Furner, A. Chen, A. V. Desai, D. J. Benedetti, D. L. Friedman, M. D. Wyatt, M. Watkins, S. L. Volchenboum, S. L. Cohn. *Extracting Electronic Health Record Neuroblastoma Treatment Data With High Fidelity Using the REDCap Clinical Data Interoperability Services Module*. JCO Clinical Cancer Informatics, 2024, Volume 8, e2400009.

[10] *Biomeris S.r.l.*: `https://www.biomeris.it/`.

[11] *REDCap API Documentation*: `https://redcap-new.labmedinfo.org/api/help/`.

[12] *Hibernate ORM*. Disponibile online: `https://hibernate.org/orm/`.

[13] *Servizi Biomeris: i2b2*: `https://www.biomeris.it/servizi/i2b2/`.

Appendice A

Acronimi

API	Application Programming Interface
CDM	Common Data Model
CDP	Clinical Data Pull
CRF	Case Report Form
CSV	Comma-Separated Values
DDP	Dynamic Data Pull
eCRF	Electronic Case Report Form
EHR	Electronic Health Record
ETL	Extract, Transform, Load
FHIR	Fast Healthcare Interoperability Resources
HL7	Health Level Seven
HTTP	HyperText Transfer Protocol
JPA	Java Persistence API
JSON	JavaScript Object Notation
LOINC	Logical Observation Identifiers Names and Codes
OHDSI	Observational Health Data Sciences and Informatics
OMOP	Observational Medical Outcomes Partnership
ORM	Object-Relational Mapping
RDBMS	Relational Database Management System
REDCap	Research Electronic Data Capture
REST	Representational State Transfer
RxNorm	Normalized Naming System for Generic and Branded Drugs
SNOMED CT	Systematized Nomenclature of Medicine – Clinical Terms
SQL	Structured Query Language

Appendice B

File JSON di configurazione

```
{
  "variables": [
    {
      "name": "albumin",
      "semantic": {
        "omop_concept_id": 3024561,
        "omop_domain": "Measurement"
      },
      "context": {
        "main": true,
        "inputs": [
          {
            "name": "data_visita",
            "ref_date_ref": "albumin",
            "repeating": true
          }
        ],
        "facts": {
          "fact_1": {
            "filters": [
              { "type": "DOMAIN", "domain_ref": "albumin" },
              { "type": "CONCEPT", "concept_id_ref": "albumin" },
              {
                "type": "DATE",
```

```
        "date_ref": "data_visita",
        "filter_type": "BOTH",
        "filter_value": 30
      }
    ],
    "refval": "latest"
  }
},
"outputs": {
  "out_albumin": {
    "redcap_var": {
      "name": "albumina",
      "event": "visit_arm_1"
    },
    "logic": {
      "type": "col_num",
      "input_val": "fact_1.ValueAsNumber",
      "output_decimal": ",",
      "conversion_factor": "1"
    }
  },
  "out_albumin_min": {
    "redcap_var": {
      "name": "albumina_min",
      "event": "visit_arm_1"
    },
    "logic": {
      "type": "col_num",
      "input_val": "fact_1.RangeLow",
      "output_decimal": ",",
      "conversion_factor": "1"
    }
  },
  "out_albumin_max": {
    "redcap_var": {
      "name": "albumina_max",
      "event": "visit_arm_1"
```

```
        },
        "logic": {
          "type": "col_num",
          "input_val": "fact_1.RangeHigh",
          "output_decimal": ",",
          "conversion_factor": "1"
        }
      }
    }
  }
},
{
  "name": "creatinine",
  "semantic": {
    "omop_concept_id": 3051825,
    "omop_domain": "Measurement"
  },
  "context": {
    "main": true,
    "inputs": [
      {
        "name": "data_visita",
        "ref_date_ref": "creatinine",
        "repeating": true
      }
    ],
    "facts": {
      "fact_1": {
        "filters": [
          { "type": "DOMAIN", "domain_ref": "creatinine" },
          { "type": "CONCEPT", "concept_id_ref": "creatinine" },
          {
            "type": "DATE",
            "date_ref": "data_visita",
            "filter_type": "BOTH",
            "filter_value": 30
          }
```

```
    ],
    "refval": "latest"
  }
},
"outputs": {
  "out_creatinine": {
    "redcap_var": {
      "name": "creatinina",
      "event": "visit_arm_1"
    },
    "logic": {
      "type": "col_num",
      "input_val": "fact_1.ValueAsNumber",
      "output_decimal": ",",
      "conversion_factor": "1"
    }
  },
  "out_creatinine_min": {
    "redcap_var": {
      "name": "creatinina_min",
      "event": "visit_arm_1"
    },
    "logic": {
      "type": "col_num",
      "input_val": "fact_1.RangeLow",
      "output_decimal": ",",
      "conversion_factor": "1"
    }
  },
  "out_creatinine_max": {
    "redcap_var": {
      "name": "creatinina_max",
      "event": "visit_arm_1"
    },
    "logic": {
      "type": "col_num",
      "input_val": "fact_1.RangeHigh",
```

```
          "output_decimal": ",",
          "conversion_factor": "1"
        }
      }
    }
  }
 },
 {
"name":"calcio",
"semantic":{
 "omop_concept_id":3032503,
 "omop_domain":"Measurement"
},
"context":{
 "main" : true,
 "inputs":[
  {
   "name":"data_visita",
   "ref_date_ref":"calcio",
   "repeating":true
  }
 ],
 "facts":{
  "fact_1":{
   "filters":[
    {
     "type":"DOMAIN",
     "domain_ref":"calcio"
    },
    {
     "type":"CONCEPT",
     "concept_id_ref":"calcio"
    },
    {
     "type":"DATE",
     "date_ref":"data_visita",
     "filter_type":"BOTH",
```

```
    "filter_value":30
   }
  ],
  "refval":"latest"
 }
},
"outputs":{
 "out_calcium":{
  "redcap_var":{
   "name":"calcio",
          "event": "visit_arm_1"
  },
  "logic":{
   "type":"col_num",
   "input_val":"fact_1.ValueAsNumber",
   "output_decimal":",",
   "conversion_factor":"1"
  }
 },
     "out_calcium_min": {
       "redcap_var": {
         "name": "calcio_min",
         "event": "visit_arm_1"
       },
       "logic": {
         "type": "col_lookup",
         "input_val": "fact_1.RangeLow",
         "input_format": [
    "6",
    "7"
   ],
   "output_format": [
    "1",
    "2"
   ]
       }
     },
```

```
          "out_calcium_max": {
            "redcap_var": {
              "name": "calcio_max",
              "event": "visit_arm_1"
            },
            "logic": {
              "type": "col_lookup",
              "input_val": "fact_1.RangeHigh",
              "input_format": [
        "9",
        "10"
       ],
       "output_format": [
        "1",
        "2"
       ]
            }
          }
    }
   }
  }
  ]
}
```

Appendice C

Log di esecuzione

C.1 Log della lettura degli input REDCap

```
{data_visita=[
=== REDCAP RECORD ===
recordId: 1
eventName: visit_arm_1
repeatInstrument:
repeatInstance: 1
fields:
  - calcio = 8,9
  - albumina_max = 5,39
  - cognome =
  - nome =
  - data_visita = 08-11-2025
  - creatinina = 2,5
  - creatinina_max = 2,75
  - visita_complete = 0
  - calcio_max = 1
  - albumina_min = 4,41
  - data_nascita =
  - albumina = 4,9
  - creatinina_min = 2,25
  - calcio_min = 1
  - anagrafica_complete =
```

```
====================
,
=== REDCAP RECORD ===
recordId: 1
eventName: visit_arm_1
repeatInstrument:
repeatInstance: 2
fields:
  - calcio = 9,1
  - albumina_max = 4,29
  - cognome =
  - nome =
  - data_visita = 10-10-2020
  - creatinina = 1,1
  - creatinina_max = 1,21
  - visita_complete = 0
  - calcio_max = 1
  - albumina_min = 3,51
  - data_nascita =
  - albumina = 3,9
  - creatinina_min = 0,99
  - calcio_min = 1
  - anagrafica_complete =
====================
,
=== REDCAP RECORD ===
recordId: 1
eventName: visit_arm_1
repeatInstrument:
repeatInstance: 3
fields:
  - calcio = 8,9
  - albumina_max = 4,62
  - cognome =
  - nome =
  - data_visita = 10-10-2019
  - creatinina = 2,8
```

```
  - creatinina_max = 3,08
  - visita_complete = 0
  - calcio_max = 1
  - albumina_min = 3,78
  - data_nascita =
  - albumina = 4,2
  - creatinina_min = 2,52
  - calcio_min = 1
  - anagrafica_complete =
====================
]}
```

C.2 Log di applicazione dei filtri sui measurements numerici estratti da OMOP

```
=== FILTERS ===
Events size: 10
=== FILTER ===
Filter type: DOMAIN
>>> FILTER NON RICONOSCIUTO: DOMAIN
=== FILTER ===
Filter type: CONCEPT
>>> FILTER NON RICONOSCIUTO: CONCEPT
=== FILTER ===
Filter type: DATE
>>> DATE FILTER ATTIVO
Filtered size: 1
=== FILTERED MEASUREMENTS ===
Value: 4.9
Date: 2025-11-08
=== SELECTED MEASUREMENT ===
Value: 4.9
Date: 2025-11-08
=== CONVERSION ===
Type: col_num
Raw value: 4.9
```

```
Converted value: 4,9
=== READ CURRENT VALUES ===
Field: albumina
Old:
New: 4,9
MODIFICA APPLICATA
```

C.3 Log di applicazione dei filtri sui measurements di tipo tabellare estratti da OMOP

```
=== FILTERS ===
Events size: 20
=== FILTER ===
Filter type: DOMAIN
>>> FILTER NON RICONOSCIUTO: DOMAIN
=== FILTER ===
Filter type: CONCEPT
>>> FILTER NON RICONOSCIUTO: CONCEPT
=== FILTER ===
Filter type: DATE
>>> DATE FILTER ATTIVO
Filtered size: 2
=== FILTERED MEASUREMENTS ===
Value: 9.0
Date: 2021-09-08
Value: 9.0
Date: 2021-09-15
=== SELECTED MEASUREMENT ===
Value: 9.0
Date: 2021-09-15
=== CONVERSION ===
Type: col_num
Raw value: 9.0
Converted value: 9,0
=== READ CURRENT VALUES ===
Field: calcio
```

```
Old: 9,0
New: 9,0
=== CONVERSION ===
Type: col_lookup
Raw value: 7
Lookup match NUM: 7 -> 2
=== READ CURRENT VALUES ===
Field: calcio_min
Old: 2
New: 2
=== CONVERSION ===
Type: col_lookup
Raw value: 10
Lookup match NUM: 10 -> 2
=== READ CURRENT VALUES ===
Field: calcio_max
Old: 2
New: 2
Nessuna modifica, nessuna scrittura
```

Appendice D

File pom.xml del progetto ETL

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    https://maven.apache.org/xsd/maven-4.0.0.xsd">
 <modelVersion>4.0.0</modelVersion>
 <parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>3.5.6</version>
  <relativePath /> <!-- lookup parent from repository -->
 </parent>
 <groupId>it.biomeris</groupId>
 <artifactId>OMOP2REDCapEvolution</artifactId>
 <version>0.0.1</version>
 <name>OMOP2REDCapEvolution</name>
 <description>Evolution of OMOP2REDCap project</description>
 <url />
 <licenses>
  <license />
 </licenses>
 <developers>
  <developer />
 </developers>
```

```
<scm>
 <connection />
 <developerConnection />
 <tag />
 <url />
</scm>
<properties>
 <java.version>21</java.version>
</properties>

<dependencies>
 <dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-data-jpa</artifactId>
 </dependency>
 <dependency>
  <groupId>org.postgresql</groupId>
  <artifactId>postgresql</artifactId>
 </dependency>
 <dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-test</artifactId>
  <scope>test</scope>
 </dependency>
 <dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-configuration-processor</artifactId>
  <optional>true</optional>
 </dependency>
 <dependency>
     <groupId>it.biomeris.omopUtility</groupId>
     <artifactId>omop-utility</artifactId>
     <version>1.0.0</version>
 </dependency>
 <dependency>
   <groupId>it.biomeris</groupId>
   <artifactId>redcap-utility</artifactId>
```

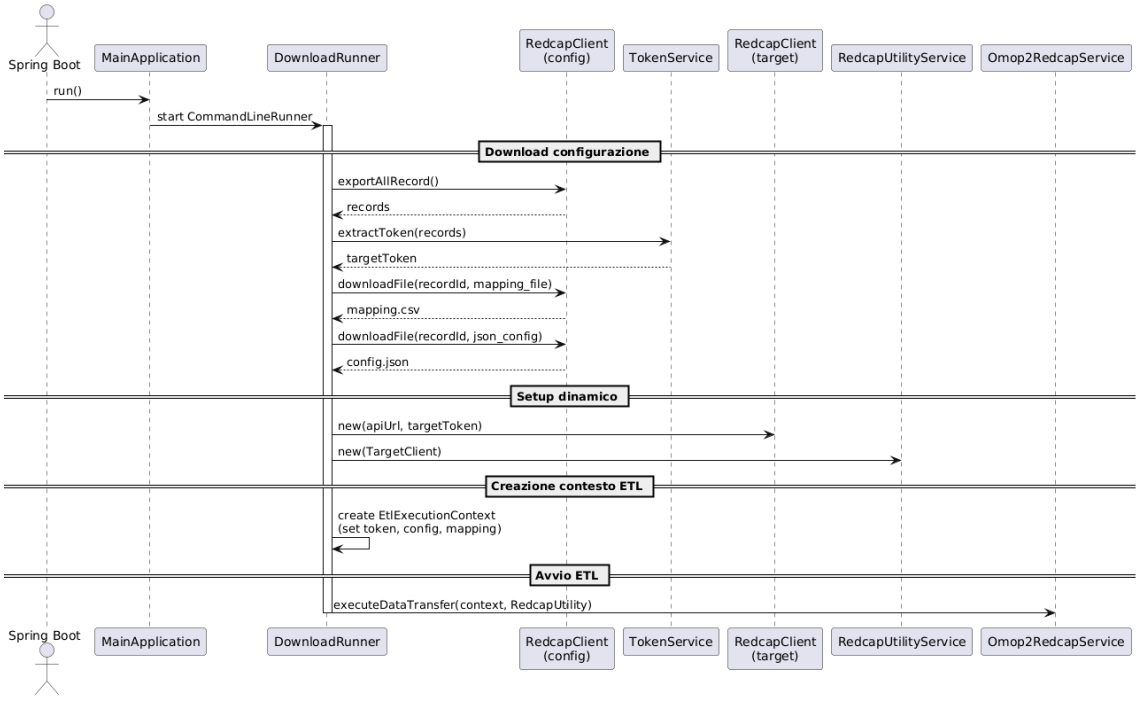
```
    <version>1.0.0</version>
  </dependency>
  <dependency>
       <groupId>com.google.code.gson</groupId>
          <artifactId>gson</artifactId>
  </dependency>

 </dependencies>
 <build>
  <plugins>
   <plugin>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-maven-plugin</artifactId>
    <configuration>
           <skip>true</skip>
       </configuration>
   </plugin>
  </plugins>
 </build>
</project>
```

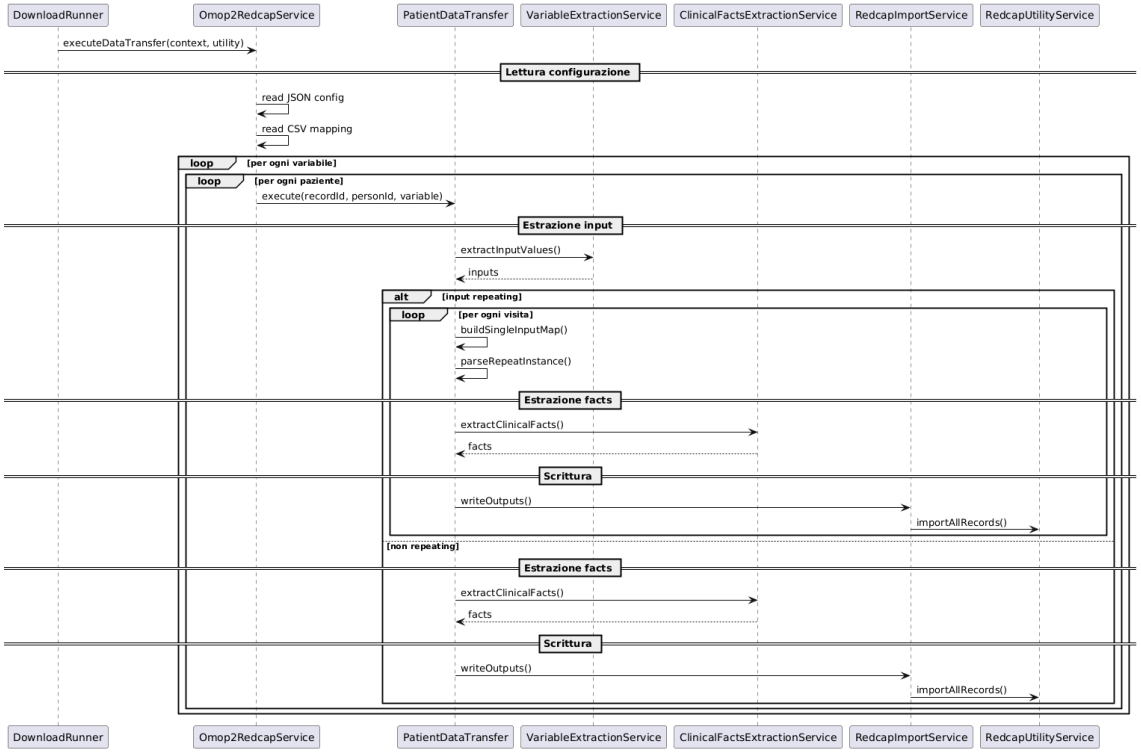
Appendice E

Diagrammi di flusso

E.1 Diagramma di flusso del progetto Execute



E.2 Diagramma di flusso del progetto ETL



Ringraziamenti