UNIVERSITÀ
DEGLI STUDI DI BARI
ALDO MORO

DIPARTIMENTO DI INFORMATICA

CORSO DI LAUREA IN INFORMATICA

TESI DI LAUREA
IN
INGEGNERIA DEL SOFTWARE

Progettazione e sviluppo di un software basato su microservizi per la gestione dei genogrammi sanitari

RELATORI:
Prof. Filippo Lanubile
Prof. Giulio Mallardi

LAUREANDO:
Danilo Santo

ANNO ACCADEMICO 2024 - 2025

Disclaimer

Tutti i marchi, nomi commerciali, prodotti e loghi menzionati in questa tesi sono di proprietà dei rispettivi titolari. L'autore non rivendica alcun diritto di proprietà su tali marchi o nomi commerciali e li utilizza solo a scopo informativo e descrittivo, senza alcun intento di violazione.

Indice

Introduzione 1

Motivazione e Contesto 1
Obiettivi della Tesi 2
Contributo Innovativo 3
Struttura del Documento 3
Metodologia e Tecnologie 4

Introduzione 1

1 La raccolta dei dati sanitari in ambito Healthcare 5

1.1 Introduzione 5
1.2 Tipologie di dati raccolti 6
1.3 Finalità della raccolta dei dati 9
1.4 Qualità e organizzazione dei dati 10
1.5 Ruolo dei dati nella medicina predittiva e personalizzata 13

2 Rappresentazione grafica della storia clinica familiare 16

2.1 Introduzione 16
2.2 Evoluzione storica delle rappresentazioni familiari 18
2.3 Dall'albero genealogico al genogramma clinico 20
2.4 Simbologia, standard e convenzioni nei genogrammi 22
2.5 Utilizzo dei genogrammi in ambito clinico 26
2.5.1 Medicina genetica e consulenza 26
2.5.2 Applicazione nei percorsi clinici 26
2.5.3 Integrazione nei sistemi informativi sanitari 27
2.6 Considerazioni finali 28

3 Standard e strumenti per la gestione di pedigree genetici 30
3.1 BOADICEA: un modello predittivo per il rischio oncologico ereditario 31
3.2 CanRisk: piattaforma avanzata per la valutazione del rischio oncologico . . . 32
3.3 Altri Strumenti Web Esistenti per la Gestione di Pedigree . . . 33
3.4 PedigreeJS: Libreria di Visualizzazione . . . 35

4 Strumenti e tecnologie utilizzate 39
4.1 Ambiente di sviluppo . . . 39
4.2 Architettura generale del sistema . . . 40
4.3 Frontend: tecnologie e librerie utilizzate . . . 42
4.4 Backend: architettura e tecnologie utilizzate . . . 44
4.5 Database e gestione dei dati . . . 46
4.6 Versionamento del codice: Git e GitHub . . . 48

5 Analisi funzionale e comportamento dell'applicazione 50
5.1 Panoramica generale dell'applicazione . . . 50
5.2 Requisiti dell'applicazione . . . 51
5.2.1 Requisiti funzionali . . . 51
5.2.2 Requisiti non funzionali . . . 53
5.3 Modello dati e progettazione del database . . . 54
5.3.1 Struttura delle entità principali . . . 54
5.3.2 Struttura dei dati genealogici . . . 55
5.3.3 Considerazioni progettuali . . . 55
5.4 Interfaccia utente e flusso funzionale . . . 56
5.4.1 Architettura del frontend . . . 56
5.4.2 Comportamento dell'interfaccia utente (UI) . . . 60
5.4.3 Comportamento del sistema e API REST . . . 62
5.5 Criticità e soluzioni adottate . . . 68
5.5.1 Identificazione delle limitazioni della libreria originale . . . 68
5.5.2 Sviluppo della soluzione basata sui nodi nascosti . . . 69
5.5.3 Implementazione della gestione dei partner nascosti . . . 70
5.5.4 Implementazione della gestione dei figli per coppie precedentemente single . . . 71

5.5.5 Impatto clinico e valore innovativo 72

6 Conclusioni e sviluppi futuri 73

Bibliografia 75

Introduzione

Nell'era della medicina digitale e della sanità personalizzata, la gestione efficace dei dati clinici rappresenta un pilastro fondamentale per il miglioramento della qualità dell'assistenza sanitaria. Tra le informazioni più preziose per i professionisti sanitari vi è la storia clinica familiare, che consente di identificare pattern ereditari, valutare il rischio genetico individuale e orientare le decisioni terapeutiche verso approcci sempre più mirati e personalizzati.

I genogrammi e i pedigree genetici sono strumenti utilizzati da decenni per rappresentare i legami familiari. In origine erano semplici alberi genealogici disegnati a mano, ma oggi si sono trasformati in veri e propri strumenti clinici. Questi diagrammi permettono di visualizzare non solo i rapporti di parentela, ma anche informazioni mediche e genetiche importanti per la diagnosi. Con l'arrivo dell'informatica, però, è emersa la necessità di creare software specifici che riescano a gestire tutte queste informazioni complesse in ambito ospedaliero.

Motivazione e Contesto

Il presente lavoro nasce dall'esigenza di colmare il divario esistente tra gli strumenti accademici disponibili per la gestione dei pedigree genetici e le necessità operative quotidiane del personale sanitario. Durante l'analisi dei requisiti con i professionisti medici, sono emerse limitazioni significative negli strumenti esistenti, in particolare nell'impossibilità di rappresentare configurazioni familiari comuni nella pratica clinica, come genitori single con figli o coppie senza progenie.

La crescente importanza della medicina predittiva e della consulenza genetica, supportata da standard internazionali come BOADICEA e CanRisk, richiede strumenti informatici che possano garantire non solo la conformità agli standard medici,

ma anche la flessibilità necessaria per adattarsi alle diverse situazioni cliniche che si presentano nella realtà ospedaliera.

Questo progetto è stato sviluppato lavorando insieme al Centro per le Malattie Neurodegenerative (CMND) dell'Università di Bari, che ha sede presso l'Ospedale Panico di Tricase, in provincia di Lecce. Il CMND è un centro di ricerca importante che si occupa di studiare le malattie neurodegenerative, in particolare analizzando come queste si trasmettono nelle famiglie per valutare il rischio genetico. Grazie a questa collaborazione, ho potuto creare una soluzione software che risponde davvero alle necessità dei medici e degli operatori sanitari, mantenendo allo stesso tempo gli standard scientifici richiesti.

Obiettivi della Tesi

L'obiettivo principale di questo lavoro è lo sviluppo di un sistema informatico completo per la gestione dei pazienti e la visualizzazione interattiva di genogrammi clinici, che risolva le limitazioni identificate negli strumenti esistenti mantenendo la piena compatibilità con gli standard medici internazionali.

Gli obiettivi specifici includono:

- **Analisi del dominio**: Approfondimento delle modalità di raccolta e gestione dei dati sanitari, con particolare focus sulla rappresentazione grafica della storia clinica familiare
- **Progettazione architetturale**: Sviluppo di un'architettura software moderna e scalabile, basata su tecnologie web full-stack
- **Innovazione tecnica**: Implementazione di una soluzione innovativa per superare le limitazioni della libreria PedigreeJS attraverso il sistema dei "nodi nascosti"
- **Integrazione standard**: Garantire la compatibilità con i formati medici standardizzati (BOADICEA v4, CanRisk v4.0) per l'interoperabilità con strumenti esistenti

- **Validazione clinica**: Sviluppo di un sistema utilizzabile in contesti clinici reali, con interfaccia intuitiva e workflow adattati alle esigenze del personale sanitario

Contributo Innovativo

Il principale contributo innovativo di questo lavoro risiede nello sviluppo di una soluzione tecnica originale che permette di superare i vincoli strutturali della libreria PedigreeJS, mantenendone l'integrità e la compatibilità con gli standard esistenti. Il sistema dei "nodi nascosti" rappresenta un approccio elegante che introduce flessibilità nella rappresentazione visiva preservando la coerenza logica della struttura dati sottostante.

Questa soluzione consente, per la prima volta, di rappresentare configurazioni familiari precedentemente impossibili da visualizzare, come genitori single con figli o coppie senza progenie, situazioni frequenti nella pratica clinica ma non gestibili dagli strumenti esistenti.

Struttura del Documento

La tesi è organizzata in sei capitoli principali:

Capitolo 1 introduce il tema della raccolta dei dati sanitari in ambito healthcare, analizzando tipologie, finalità e qualità dei dati, con particolare attenzione al ruolo nella medicina predittiva e personalizzata

Capitolo 2 esplora l'evoluzione storica delle rappresentazioni grafiche familiari, dall'albero genealogico tradizionale al genogramma clinico moderno, analizzandone l'utilizzo in ambito sanitario

Capitolo 3 presenta una panoramica completa degli standard e strumenti esistenti per la gestione di pedigree genetici, con focus su BOADICEA, CanRisk e PedigreeJS

Capitolo 4 descrive dettagliatamente l'architettura tecnologica adottata, gli strumenti di sviluppo e le scelte progettuali effettuate

Capitolo 5 fornisce un'analisi funzionale approfondita dell'applicazione sviluppata, includendo requisiti, modello dati, interfaccia utente e le soluzioni innovative implementate

Capitolo 6 presenta le conclusioni del lavoro, evidenziando i risultati raggiunti e le prospettive future

Metodologia e Tecnologie

Il progetto è stato sviluppato seguendo una metodologia iterativa, con coinvolgimento attivo del personale medico per l'identificazione dei requisiti e la validazione delle soluzioni proposte. L'architettura adottata si basa su tecnologie moderne e consolidate: Angular per il frontend, Spring Boot per il backend, PostgreSQL per la persistenza dei dati.

Per la visualizzazione è stato scelto di utilizzare PedigreeJS, una libreria sviluppata dal Centre for Cancer Genetic Epidemiology dell'Università di Cambridge. Questa scelta è stata dettata dal fatto che rispetta gli standard medici internazionali e proviene da un ambiente accademico affidabile. Il mio lavoro dimostra come l'informatica possa essere utile in campo sanitario: ho cercato di creare uno strumento che fosse sia tecnicamente valido sia praticamente utilizzabile dai medici nel loro lavoro quotidiano.

Capitolo 1

La raccolta dei dati sanitari in ambito Healthcare

1.1 Introduzione

Negli ultimi decenni, il settore sanitario ha assistito a una trasformazione radicale, spinta dall'evoluzione tecnologica e dalla crescente disponibilità di dati clinici. La raccolta, gestione e analisi di questi dati rivestono oggi un ruolo cruciale nel miglioramento della qualità dell'assistenza, nella prevenzione delle patologie e nella personalizzazione dei trattamenti.

I dati sanitari rappresentano il punto di partenza per una medicina basata sull'evidenza: essi consentono di documentare lo stato di salute dei pazienti, di supportare le decisioni cliniche e di monitorare l'efficacia delle terapie nel tempo. La loro raccolta sistematica non solo garantisce la continuità assistenziale, ma fornisce anche le basi per lo sviluppo di nuovi approcci predittivi e proattivi, grazie all'impiego di tecnologie emergenti come l'intelligenza artificiale e l'analisi dei big data.

In questo contesto, la corretta gestione dei dati sanitari non è più una semplice attività amministrativa, ma una componente fondamentale dell'intero ecosistema sanitario. Questo primo capitolo si propone di introdurre il tema della raccolta dei dati in ambito healthcare, analizzando le sue finalità, le tipologie di dati coinvolti, l'importanza della loro qualità e le opportunità che essi offrono nell'ambito della medicina moderna.

Oltre al processo clinico, la raccolta dati assume un ruolo strategico anche su scala di popolazione. Attraverso la sorveglianza epidemiologica, si possono identificare trend di malattie emergenti, monitorare l'andamento di epidemie e valutare l'impatto delle misure sanitarie adottate – come avvenuto con la pandemia di COVID-19 [10]. Queste informazioni alimentano decisioni politiche e investimenti per la salute pubblica.

Parallelamente, alla base di una sanità moderna vi è la necessità di condividere i dati in modo sicuro tra strutture ospedaliere, ambulatori e studi medici. L'impiego di sistemi come l'EHR (Electronic Health Record) rappresenta un primo passo verso interoperabilità e integrazione delle informazioni, permettendo di seguire l'intero percorso del paziente lungo più strutture [8].

Infine, l'utilizzo dei dati contrappone benefici significativi a sfide non trascurabili: privacy, sicurezza, etica e compliance normativa (es. GDPR, HIPAA) richiedono la definizione di regole rigorose e di sistemi robusti di governance [3]. Gestire correttamente questi aspetti è condizione necessaria affinché i dati possano essere davvero uno strumento al servizio della cura, della prevenzione e della ricerca.

1.2 Tipologie di dati raccolti

Le cartelle cliniche elettroniche (Electronic Health Records, EHR) raccolgono un ampio spettro di informazioni sul paziente e sul processo assistenziale. Tali dati possono essere classificati in base alla loro struttura e fonte:

Dati strutturati

Questi dati sono rappresentati da campi predefiniti e codifiche standardizzate, facilmente leggibili e interpretati da software. Comprendono:

- *Informazioni demografiche*: età, sesso, data di nascita, indirizzo, identificativi.
- *Codici diagnostici e procedure* (es. ICD-10, CPT).
- *Prescrizioni e somministrazioni farmacologiche.*
- *Segni vitali* e valori di laboratorio.

Grazie alla struttura tabellare, questi dati sono ideali per il calcolo statistico, report e integrazione nei flussi clinici [1].

Dati non strutturati

Costituiti da documenti di testo libero, immagini, referti e note narrative. Sebbene contengano informazioni dettagliate e contestuali (come la gravità dei sintomi, storie personali, osservazioni cliniche), sono difficili da analizzare automaticamente a causa dell'assenza di formato standard. Sono inclusi:

- *Note mediche*, referti radiologici, referti patologici.
- *Immagini mediche* (radiografie, TAC, risonanze magnetiche).
- *Audio/video delle visite*, dati generati dal paziente.

Questo tipo di dati rappresenta oltre l'80% del contenuto clinico totale e si presta bene a tecniche di NLP e mining per estrarre informazioni strutturate [2].

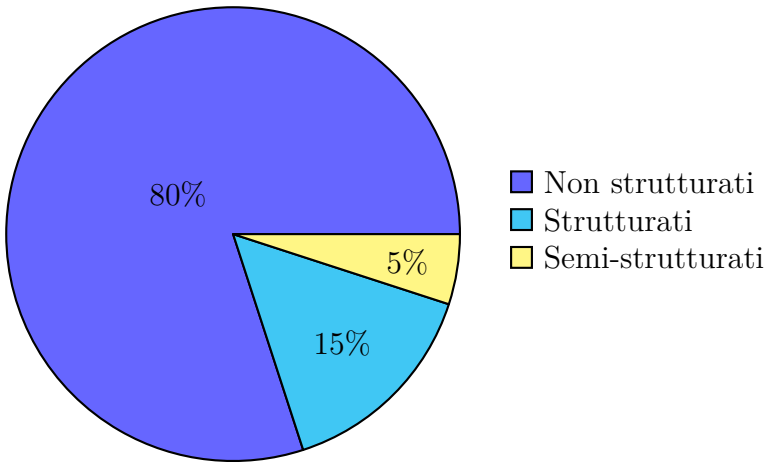

Figura 1.1. Distribuzione stimata dei dati clinici nei sistemi EHR

Dati semi-strutturati e integrazione

Un terzo gruppo comprende dati come XML, JSON, documenti HL7 CDA e FHIR:

- *FHIR*, standard REST/JSON/XML, facilita l'interoperabilità tra sistemi e la condivisione dei dati tra ospedali e applicazioni cliniche [5].

- *HL7 CDA / C-CDA*, formati testuali strutturati molto utilizzati negli USA per lo scambio documentale clinico [4].

Questi standard permettono di unire diversi tipi di informazioni mediche in documenti che possono essere facilmente condivisi tra ospedali, laboratori e reparti.

Esempi e casi d'uso

- In oncologia, i dati strutturati (es. codici TNM, dosaggio farmaci) forniscono elementi per modelli predittivi, mentre le note narrative descrivono sintomi, effetti collaterali e qualità di vita.
- In epidemiologia, i registri demografici "strutturati" abbinati ai dati testuali e geografici permettono il monitoraggio dei focolai e trend di malattia.

Sfide associate

L'eterogeneità dei dati richiede:

- *Processi di normalizzazione e codifica* (es. mapping verso ICD, SNOMED).
- *Uso di tecnologie di NLP*, OCR e audio-processing per rendere i dati non strutturati utilizzabili.
- *Gestione efficace dello storage*, vista l'enorme mole di dati (es. immagini mediche richiedono centinaia di MB cadauna) [2].
- *Governance forte* per garantire sicurezza, privacy e gestione del ciclo di vita dei dati.

Tabella 1.1. Confronto tra tipi di dati sanitari

Caratteristica	**Strutturati**	**Semi-strutturati**	**Non strutturati**
Formato	Tabellare, codificato (es. ICD)	XML, JSON, HL7 CDA/FHIR	Testo libero, immagini, audio/video
Facilità di analisi	Elevata	Media	Bassa
Esempi	Segni vitali, codici diagnostici	Referti XML, FHIR	Note cliniche, referti PDF, immagini
Utilizzo nei modelli predittivi	Diretto	Spesso convertito in strutturato	Necessita di NLP, OCR

1.3 Finalità della raccolta dei dati

La raccolta dei dati sanitari risponde a molteplici obiettivi fondamentali:

1. Supporto alle decisioni cliniche

I sistemi di supporto alle decisioni cliniche (Clinical Decision Support Systems, CDSS) rappresentano uno strumento avanzato che integra la conoscenza clinica con i dati del paziente per suggerire diagnosi, farmaci sicuri, alert su interazioni o condizioni critiche e promemoria terapeutici. Tali sistemi, integrati nel flusso di lavoro quotidiano (point-of-care), migliorano l'accuratezza delle diagnosi, riducono gli errori medici e ottimizzano i tempi di degenza [12]. È stato dimostrato che, quando progettati con attenzione (evitando alert eccessivi e incomprensibili), i CDSS possono aumentare l'aderenza alle linee guida e migliorare gli esiti clinici, come nella gestione di diabete, ipertensione e sepsi.

2. Sorveglianza epidemiologica e sanità pubblica

A livello di popolazione, la raccolta continua e sistematica dei dati — demografici, clinici e geografici — permette di riconoscere rapidamente cambiamenti nelle dinamiche di malattia, identificare focolai, stimare la morbilità e la mortalità, monitorare l'efficacia delle misure sanitarie e guidare interventi coerenti con le evidenze attuali [10]. Un esempio significativo è stato il monitoraggio durante la pandemia di influenza o COVID-19, in cui i sistemi di sorveglianza hanno permesso di attivare lockdown mirati, campagne vaccinali e di contenimento, basate su dati sanitari aggiornati.

Altri scopi della raccolta

- **Ricerca scientifica**: i dati standardizzati permettono la conduzione di clinical trial, studi osservazionali tramite archivi EHR, coorti retrospettive e studi di outcomes real-world.
- **Ottimizzazione delle risorse**: Un altro vantaggio è la possibilità di gestire meglio le risorse ospedaliere. Analizzando i flussi dei pazienti, si riesce a capire dove servono più medici, quali reparti hanno bisogno di più posti letto e come

utilizzare al meglio le apparecchiature, con il risultato di ridurre i tempi di attesa e contenere i costi.

- **Medicina personalizzata**: integrando dati clinici, genetici e comportamentali, è possibile stimare il rischio individuale per malattie e adattare terapie su misura, elemento ora centrale nella gestione di tumori, malattie rare e croniche.

3. Sinergia tra livelli assistenziali

Il patrimonio informativo accumulato facilita il care-coordination e il follow-up tra ospedali, studi medici e assistenza territoriale, garantendo continuità e completezza dell'anamnesi del paziente, fondamentale per evitare duplicazioni di esami, errori comunicativi e per attuare percorsi diagnostico-terapeutici integrati.

1.4 Qualità e organizzazione dei dati

La qualità dei dati sanitari è un prerequisito fondamentale per il loro impiego efficace in ambito clinico, scientifico, gestionale e strategico. I dati clinici devono riflettere accuratamente la realtà medica del paziente, essere completi, aggiornati e adeguatamente contestualizzati. L'assenza di qualità può comportare diagnosi errate, trattamenti inappropriati, errori statistici e decisioni manageriali fuorvianti.

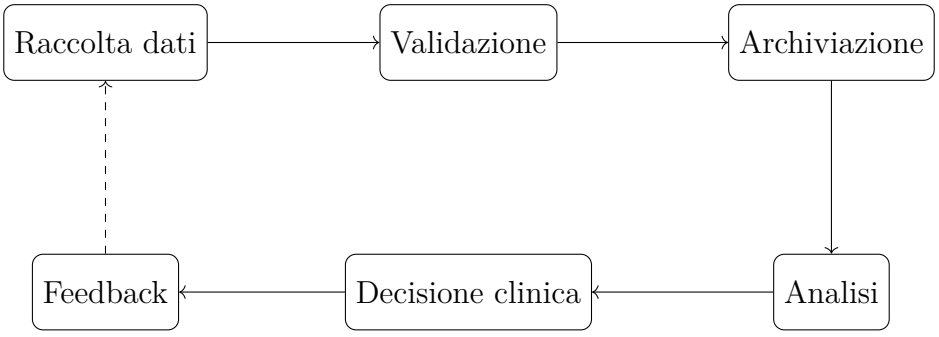

Figura 1.2. Ciclo di vita dei dati sanitari nel contesto clinico

1. Dimensioni della qualità dei dati

Secondo lo studio sistematico di Lewis et al. [7], le dimensioni fondamentali per valutare la qualità dei dati in ambito EHR includono:

- **Completezza:** presenza di tutte le informazioni necessarie.
- **Accuratezza:** correttezza rispetto ai fatti clinici reali.
- **Concordanza:** coerenza tra fonti diverse.
- **Plausibilità:** logicità dei valori registrati (es. temperatura corporea non fuori range).
- **Tempestività:** aggiornamento puntuale e in tempo reale.
- **Conformità normativa:** aderenza a standard e linee guida.
- **Assenza di bias:** neutralità rispetto a fattori socio-demografici o sistemici.

Queste dimensioni sono ulteriormente supportate dalla letteratura recente, come la rassegna pubblicata su *Journal of Medical Internet Research* [6], che propone un framework articolato in:

- **Accessibilità:** facilità di accesso ai dati per gli operatori autorizzati.
- **Consistenza:** uniformità nella rappresentazione e nel formato dei dati.
- **Validità contestuale:** adeguatezza rispetto allo scenario clinico.
- **Attualità:** grado di aggiornamento rispetto all'evoluzione della condizione clinica.

2. Tecniche e strumenti per il controllo qualità

Per garantire che i dati siano "fit for purpose", ovvero idonei all'uso previsto, vengono impiegate numerose tecniche:

- **Validazioni automatiche:** controllo di range, formati, coerenza temporale.

- **Audit clinici:** confronti periodici tra registrazioni digitali e fonti primarie (es. cartelle cartacee).
- **Riconciliazione dei dati:** in particolare per farmaci, allergie, anamnesi.
- **Data profiling e monitoraggio continuo:** tramite dashboard interattive per rilevare anomalie, outlier o trend inattesi.

3. Organizzazione e governance dei dati

Un'efficace organizzazione dei dati non riguarda solo l'archiviazione, ma include:

- **Modelli di governance:** strutture decisionali che definiscono ruoli, responsabilità e flussi di controllo.
- **Catalogazione e metadati:** ogni dato deve essere identificabile, rintracciabile e corredato di contesto.
- **Politiche di data stewardship:** gestione attiva della qualità da parte di referenti clinici e tecnici.
- **Sistemi interoperabili:** che garantiscano l'integrazione tra fonti eterogenee attraverso standard aperti (es. HL7, FHIR).

Con la digitalizzazione della sanità, diventa sempre più importante saper organizzare e gestire correttamente i dati. Solo così si possono portare avanti progetti che usano l'intelligenza artificiale, fare valutazioni sui costi, verificare la qualità dei servizi e condurre ricerche mediche.

4. Esempi pratici

- **Nel monitoraggio glicemico:** una registrazione incompleta o non coerente delle dosi insuliniche o dei pasti può compromettere il calcolo del fabbisogno e mettere a rischio il paziente.
- **Nei reparti di emergenza:** la tempestività dell'aggiornamento dei dati (es. segni vitali) può fare la differenza nella triage decision.

- **Nella ricerca clinica:** servono dati di buona qualità per ottenere risultati affidabili e ridurre gli errori negli studi. Avere dati precisi non è solo una questione tecnica, ma è importante anche dal punto di vista medico ed etico. Quando i dati rappresentano davvero quello che succede ai pazienti, si possono prendere decisioni migliori per le cure e garantire maggiore sicurezza.

La qualità dei dati non è solo un requisito tecnico, ma un valore clinico ed etico: garantire che ogni dato rifletta fedelmente la realtà sanitaria significa aumentare la sicurezza del paziente, l'efficacia delle cure e l'affidabilità delle decisioni basate su tali informazioni.

1.5 Ruolo dei dati nella medicina predittiva e personalizzata

Nell'ambito della medicina moderna, i dati sanitari assumono un ruolo centrale nella predizione di eventi clinici e nella personalizzazione delle cure.

1. Predizione precoce di eventi clinici

L'uso di modelli predittivi avanzati su serie storiche di dati clinici ha dimostrato notevoli benefici. In particolare, algoritmi basati su machine learning riescono a identificare la sepsi diverse ore prima dell'insorgenza dei primi sintomi, permettendo interventi tempestivi e riducendo mortalità e tempo di degenza ospedaliera [9]. Studi recenti hanno registrato una riduzione del 20–30 % nei casi gravi quando i modelli vengono integrati nel flusso operativo.

2. Analisi predittiva e prescrittiva

Gli Analytics sanitari generano previsioni (predictive) e suggerimenti operativi (prescriptive) combinando dati clinici, genomica, parametri da dispositivi medici e stili di vita. Queste informazioni sono utili per pianificare il ricovero, ottimizzare risorse e personalizzare il piano terapeutico [11]. Un esempio concreto: nei reparti cardiologici, l'analisi predittiva aiuta a identificare pazienti a rischio di scompenso, consentendo piani di intervento personalizzati.

3. Medicina di precisione

I dati genetici, ambientali e comportamentali permettono di sviluppare terapie mirate per ogni paziente. In oncologia questo significa abbandonare i protocolli standard a favore di trattamenti basati sulle caratteristiche molecolari del singolo tumore, con farmaci mirati e dosaggi calibrati sul paziente. Il risultato è una riduzione della tossicità e un miglioramento dell'efficacia terapeutica.

4. Sfide e limiti

Nonostante i vantaggi, l'applicazione di approcci predittivi è vincolata a:

- *Bias nei dati*: se i dati originali non sono rappresentativi, i modelli possono amplificare discriminazioni.
- *Trasferibilità limitata*: modelli addestrati in un contesto possono performare male in altri centri o popolazioni.
- *Privacy e etica*: l'utilizzo di dati sensibili richiede tecniche avanzate di anonimizzazione e consenso informato.

Tuttavia, la continua evoluzione delle tecnologie e la validazione clinica in trial real-world spingono questo campo verso una diffusione sempre maggiore.

> *"Il farmaco giusto, alla dose giusta, al paziente giusto, nel momento giusto."*

Conclusioni

La raccolta e l'elaborazione dei dati sanitari rappresentano oggi un pilastro fondamentale per un sistema sanitario moderno, efficiente e personalizzato. Attraverso l'utilizzo strutturato di dati clinici, epidemiologici e genomici, è possibile migliorare i percorsi di cura, anticipare scenari critici e promuovere una medicina sempre più orientata al paziente.

L'analisi dei dati sanitari però non riguarda solo il presente. Nel prossimo capitolo vedremo come, nel corso del tempo, le informazioni familiari e genetiche sono state rappresentate usando strumenti grafici come gli alberi genealogici e i genogrammi. Questi diagrammi sono molto utili per capire quali malattie possono

essere ereditarie e per prevenire alcune patologie, perché permettono di avere una visione completa della storia medica della famiglia.

Capitolo 2

Rappresentazione grafica della storia clinica familiare

2.1 Introduzione

La conoscenza della storia clinica familiare rappresenta da sempre un elemento fondamentale nella comprensione delle condizioni ereditarie e nella valutazione del rischio individuale di sviluppare determinate patologie. Fin dall'antichità, l'essere umano ha cercato di tracciare le proprie origini e relazioni familiari attraverso rappresentazioni grafiche note come **alberi genealogici**.

Con il tempo, queste rappresentazioni si sono evolute: da schemi semplici e lineari, focalizzati esclusivamente sui legami biologici, si è passati a strumenti sempre più articolati e ricchi di informazioni, come i **genogrammi**, capaci di includere dati clinici, psicosociali e dinamiche interpersonali.

In ambito sanitario, l'uso di tali strumenti consente ai professionisti di:

- visualizzare rapidamente le relazioni familiari;
- individuare pattern di ereditarietà genetica;
- riconoscere condizioni cliniche ricorrenti o comorbidità familiari;
- integrare queste informazioni nei percorsi di diagnosi e prevenzione.

Negli ultimi decenni, l'avvento delle tecnologie informatiche ha reso possibile la digitalizzazione e l'interattività di queste rappresentazioni, facilitando la loro

integrazione nei sistemi informativi clinici (EHR) e la loro personalizzazione rispetto al caso specifico del paziente.

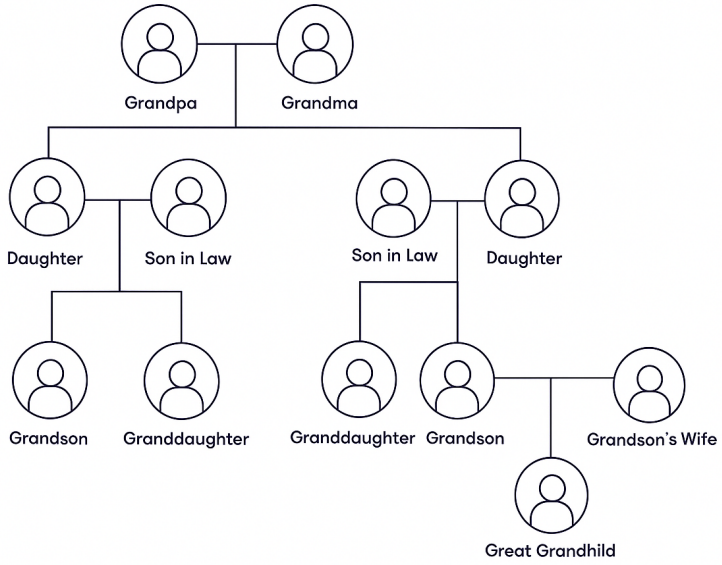

Figura 2.1. Esempio di albero genealogico tradizionale con struttura semplice.

L'immagine sopra rappresenta un tipico albero genealogico, come veniva utilizzato storicamente: una struttura lineare, incentrata esclusivamente su relazioni di parentela, senza alcuna informazione clinica.

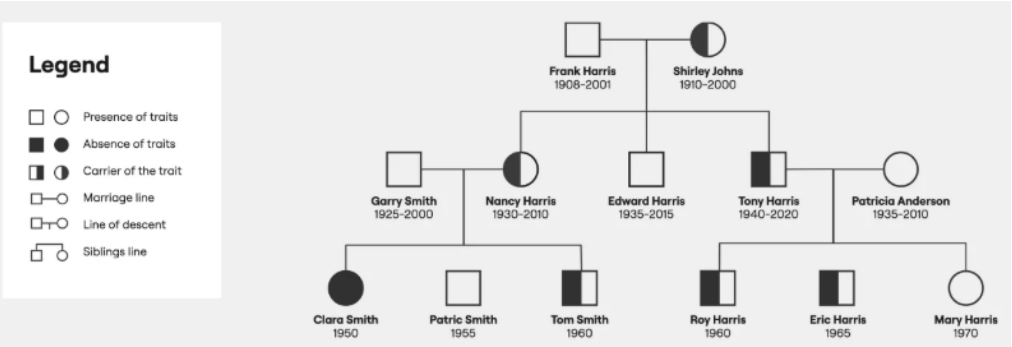

Figura 2.2. Esempio di rappresentazione moderna: genogramma clinico con legenda associata.

Nell'immagine successiva è visibile un modello moderno ispirato a sistemi come *PedigreeJS*, che integra metadati clinici e simboli standard per indicare presenza o assenza di tratti genetici, portatori sani, linee di matrimonio, discendenza e fratellanza.

Questa evoluzione grafica riflette un cambiamento di paradigma: l'albero genealogico non è più solo uno strumento genealogico, ma un **dispositivo clinico** a supporto della medicina predittiva, della genetica medica e della prevenzione personalizzata.

2.2 Evoluzione storica delle rappresentazioni familiari

La genealogia ha sempre affascinato l'umanità. Già nell'antichità, le persone sentivano l'esigenza di ricostruire le proprie origini, documentare l'appartenenza a specifiche dinastie e mantenere traccia dei rapporti familiari. Queste attività avevano molteplici finalità: preservare la memoria delle famiglie, consolidare l'identità sociale e garantire la trasmissione di titoli e beni.

Le prime rappresentazioni genealogiche si trovano nell'antico Egitto e nelle genealogie bibliche, inizialmente trasmesse oralmente o incise su pietra e papiro. L'introduzione della scrittura permise di documentare con maggiore accuratezza le linee familiari, come evidenziano i documenti della dinastia Han cinese e i rotoli ebraici che registravano le genealogie levitiche.

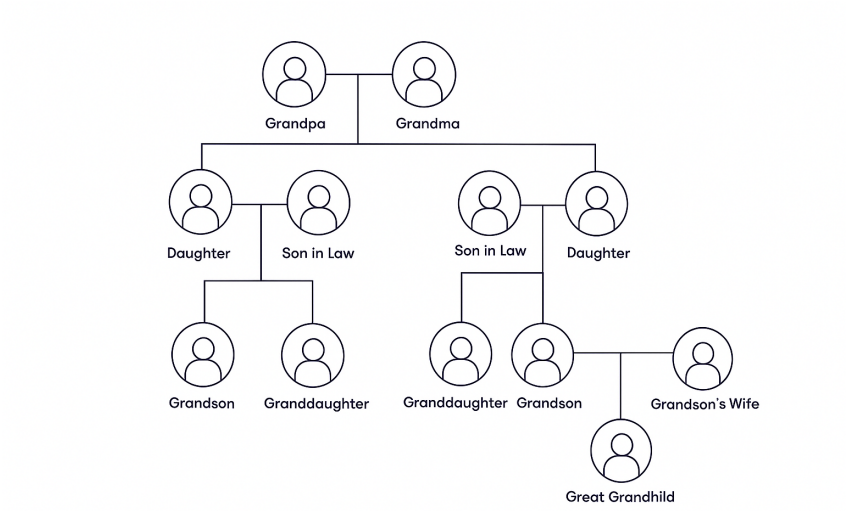

Figura 2.3. Esempio di albero genealogico tradizionale con struttura discendente.

A partire dal Rinascimento, la genealogia divenne una vera e propria materia di studio per storici e giuristi. Si iniziarono a definire regole più precise per disegnare gli alberi genealogici: linee per i matrimoni, caselle per le persone, simboli per distinguere maschi e femmine. Ma questi schemi mostravano solo i legami familiari e matrimoniali, senza alcun riferimento alle condizioni di salute o alle malattie.

È solo tra XIX e XX secolo, con l'avvento della medicina scientifica e della genetica mendeliana, che la rappresentazione delle famiglie assume un nuovo significato: non più (solo) mappatura sociale, ma strumento di rilevanza clinica. Le prime analisi di trasmissione ereditaria di malattie — come l'emofilia nelle case reali europee — si avvalsero di alberi genealogici annotati a mano, in cui i soggetti affetti venivano evidenziati con colori o simboli speciali.

Nel tempo, queste annotazioni si sono evolute in veri e propri schemi clinici, aprendo la strada allo sviluppo dei **genogrammi**, che verranno approfonditi nella sezione successiva.

2.3 Dall'albero genealogico al genogramma clinico

L'albero genealogico tradizionale ha rappresentato per secoli la base visiva per la comprensione delle relazioni familiari. Tuttavia, con la crescente importanza delle malattie ereditarie e multifattoriali, è emersa l'esigenza di strumenti più ricchi e informativi. In questo contesto nasce il **genogramma clinico**, una rappresentazione grafica evoluta, che unisce le relazioni familiari a una serie di informazioni cliniche, psicologiche e sociali dei soggetti rappresentati.

A differenza dell'albero genealogico classico, che si limita a indicare l'ascendenza e la discendenza biologica, il genogramma:

- riporta lo stato di salute o le patologie diagnosticate (es. diabete, depressione, neoplasie);
- utilizza simboli standardizzati per rappresentare sesso, relazioni affettive, divorzi, gravidanze interrotte, adozioni;
- indica con linee codificate i tipi di legami: emotivi, conflittuali, di supporto o assenza di relazione;
- può evidenziare tratti genetici specifici (portatori sani, manifestazione della malattia, predisposizione).

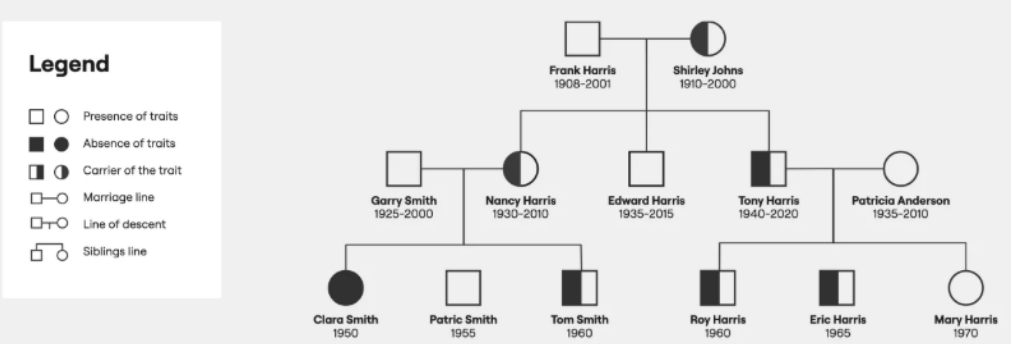

Figura 2.4. Esempio di genogramma clinico moderno con legenda integrata.

Questa rappresentazione è nata negli anni '80 nell'ambito della terapia familiare e della genetica medica, trovando ampie applicazioni nella consulenza genetica, nella

medicina preventiva e nei percorsi oncologici e psichiatrici. La sua struttura si basa su convenzioni grafiche internazionali, tra cui:

- quadrato per il maschio, cerchio per la femmina;
- soggetti affetti da una patologia riempiti o tratteggiati;
- linea continua per matrimonio, tratteggiata per convivenza;
- doppia linea per relazioni forti o co-dipendenze;
- barre trasversali per indicare decesso.

Differenze principali

Tabella 2.1. Differenze tra albero genealogico e genogramma clinico

Caratteristica	**Albero genealogico**	**Genogramma clinico**
Scopo principale	Tracciare la discendenza familiare	Analizzare pattern clinici ed emotivi familiari
Tipo di dati	Biografici (nascita, parentela)	Clinici, emotivi, relazionali
Struttura	Lineare, discendente	Strutturata con simboli codificati e legende
Ambito di utilizzo	Storico, genealogico, legale	Medico, psicologico, genetico
Livello di dettaglio	Limitato	Elevato, multi-livello

Il genogramma rappresenta oggi uno strumento imprescindibile in medicina personalizzata: consente al clinico di avere una visione d'insieme della storia familiare e sociale del paziente, di valutare predisposizioni genetiche e fattori ambientali condivisi, e di orientare diagnosi e strategie di prevenzione.

2.4 Simbologia, standard e convenzioni nei genogrammi

Nomenclatura Standard per Pedigree

La rappresentazione grafica dei pedigree segue convenzioni internazionali consolidate, sviluppate per garantire uniformità e comprensibilità universale. Questi standard, codificati da organizzazioni come l'American Society of Human Genetics (ASHG) e la European Society of Human Genetics (ESHG), definiscono simboli, forme e connessioni specifiche per rappresentare individui, relazioni familiari e informazioni mediche.

Elementi grafici di base

Rappresentazione del sesso:

- **Maschi:** quadrati
- **Femmine:** cerchi
- **Sesso sconosciuto/indeterminato o genere non binario:** rombi o triangoli

Stati di salute:

- **Individui non affetti:** simboli vuoti/bianchi
- **Individui affetti da una condizione:** simboli completamente riempiti/neri o tratteggiati
- **Portatori sani:** simboli riempiti a metà, con punto centrale o parzialmente riempiti
- **Stato sconosciuto:** simboli con punto interrogativo

Stato vitale:

- **Individui viventi:** simboli senza modifiche
- **Deceduti:** barra diagonale attraverso il simbolo
- **Aborti spontanei:** piccoli triangoli neri
- **Morti perinatali:** simboli più piccoli con linea diagonale

Relazioni familiari e matrimoniali

Matrimoni e partnership:

- **Matrimonio:** linea orizzontale continua tra due simboli
- **Convivenza:** linea orizzontale tratteggiata
- **Relazione temporanea:** linea punteggiata
- **Matrimoni multipli:** linee multiple numerate cronologicamente

Relazioni speciali:

- **Divorzio o separazione:** linea interrotta, con due barre diagonali o linea diagonale che taglia la linea matrimoniale
- **Separazioni:** due linee diagonali parallele

Discendenza:

- **Figli:** linea verticale discendente dalla linea matrimoniale
- **Adozione:** linea tratteggiata con parentesi
- **Ordine di nascita:** da sinistra a destra (più vecchio a più giovane)

Gemelli e nascite multiple

Tipi di gemelli:

- **Gemelli monozigoti (identici):** linea che si biforca con connessione tra i simboli
- **Gemelli dizigoti (fraterni):** linea che si biforca senza connessione diretta tra i simboli

Relazioni emotive (clinica psicologica o familiare)

- **Relazione positiva:** linea doppia
- **Relazione conflittuale:** linea frastagliata o con zig-zag
- **Relazione assente:** nessuna connessione
- **Dipendenza o co-dipendenza:** linea doppia molto ravvicinata

Identificazione di individui chiave

Proband:

- Individuo attraverso cui la famiglia viene identificata
- Indicato con freccia che punta al simbolo

Informazioni mediche e genetiche

Codifica delle malattie:

- **Colori specifici:** ogni malattia ha un colore standard
- **Pattern di riempimento:** diversi pattern per malattie multiple

Età e date:

- **Età attuale:** numero sotto il simbolo
- **Età al decesso:** numero seguito da "d" (es. "45d") con linea diagonale che taglia il nodo

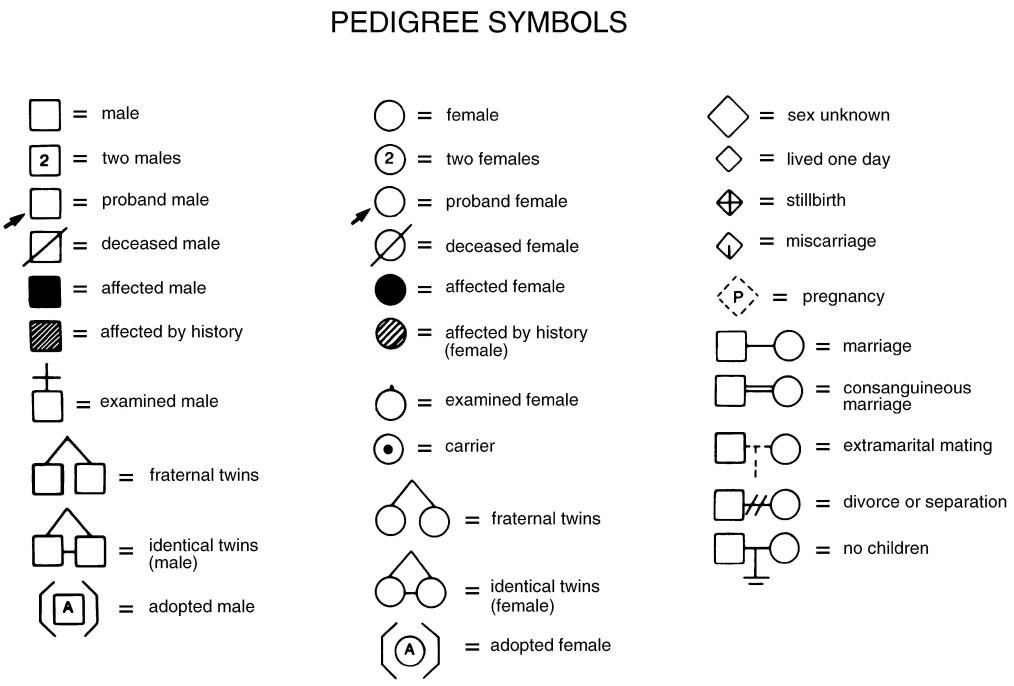

Figura 2.5. Legenda standardizzata dei simboli utilizzati nei genogrammi clinici.

Importanza della standardizzazione

L'uso coerente della simbologia consente:

- la comprensione immediata da parte di medici, genetisti, infermieri, psicologi;
- la trasmissione uniforme dell'informazione clinica tra specialisti;
- la conservazione digitale strutturata nei fascicoli sanitari elettronici;
- il supporto ai modelli predittivi o alle applicazioni di AI in ambito genetico.

2.5 Utilizzo dei genogrammi in ambito clinico

Nel contesto clinico, il genogramma rappresenta uno strumento fondamentale per la documentazione e l'analisi della storia familiare di un paziente, con l'obiettivo di supportare diagnosi, predizione e prevenzione di malattie ereditarie. A differenza dell'albero genealogico classico, il genogramma arricchisce la rappresentazione con informazioni cliniche dettagliate, facilitando l'identificazione di pattern genetici e predisposizioni patologiche.

2.5.1 Medicina genetica e consulenza

L'impiego più consolidato del genogramma si riscontra nella genetica medica. La sua costruzione consente di:

- individuare la trasmissione ereditaria di patologie (es. tumori ereditari, malattie rare);
- stimare il rischio di trasmissione ai discendenti;
- identificare portatori sani o soggetti ad alto rischio;
- supportare la pianificazione di test genetici mirati nei familiari.

Un esempio emblematico è rappresentato dalla mutazione nei geni *BRCA1* e *BRCA2*, associata al carcinoma mammario e ovarico: tramite il genogramma si possono visualizzare i casi precedenti in famiglia, favorendo una diagnosi precoce e percorsi personalizzati di prevenzione.

2.5.2 Applicazione nei percorsi clinici

I genogrammi trovano applicazione in diverse fasi del percorso sanitario:

- durante la raccolta dell'anamnesi familiare in ambulatorio;
- nel processo decisionale condiviso tra medico e paziente;
- nella gestione dei programmi di screening e prevenzione primaria;
- nell'inquadramento delle malattie croniche complesse con componente familiare.

Ad esempio, nella gestione del diabete di tipo 2 o delle dislipidemie familiari, la storia clinica familiare supporta l'identificazione precoce dei soggetti a rischio e l'adozione di strategie di intervento preventivo.

2.5.3 Integrazione nei sistemi informativi sanitari

L'integrazione dei genogrammi nei sistemi di Electronic Health Record (EHR) rappresenta un passo fondamentale verso una medicina personalizzata e basata sulla prevenzione. La rappresentazione strutturata della storia familiare permette di collegare in modo dinamico i dati genealogici con informazioni cliniche, genomiche e ambientali, rendendo il genogramma un vero e proprio modulo funzionale nei sistemi sanitari digitali.

Questa integrazione offre diverse funzionalità utili:

- permette di consultare e aggiornare costantemente la storia familiare del paziente;
- collega i genogrammi con i dati clinici e i referti medici;
- attivazione automatica degli avvisi nei sistemi di supporto quando ci sono malattie ereditarie;
- crea database organizzati per la ricerca clinica e gli studi epidemiologici.

I genogrammi integrati sono particolarmente importanti quando c'è un alto rischio genetico. In questi casi, riconoscere per tempo i pattern familiari aiuta i medici a impostare subito programmi di prevenzione e controlli mirati.

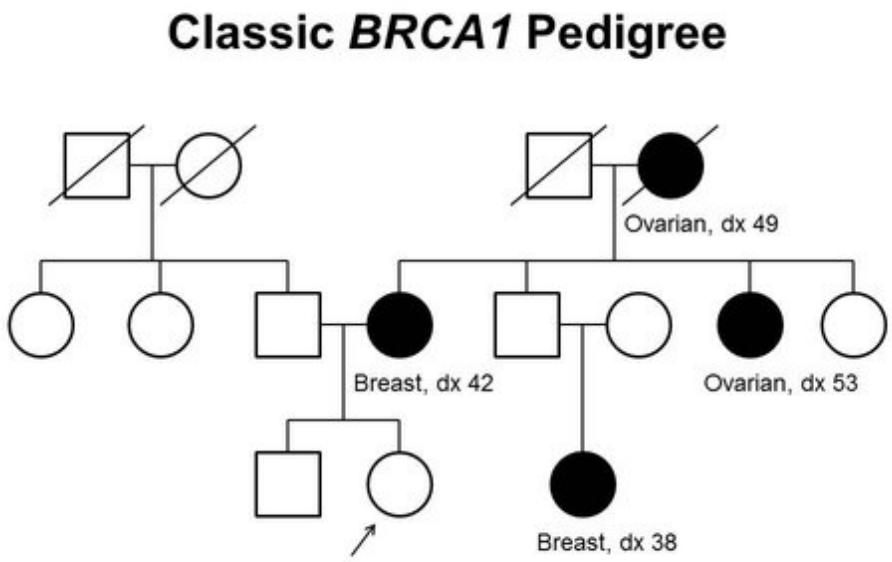

Figura 2.6. Pedigree familiare con mutazione BRCA1 su tre generazioni. I soggetti affetti da tumore al seno e alle ovaie sono evidenziati con simboli pieni. La trasmissione autosomica dominante suggerisce un alto rischio genetico per i discendenti diretti. Diagrammi di questo tipo sono frequentemente impiegati nella consulenza genetica per supportare la decisione di effettuare test predittivi e attuare strategie di sorveglianza precoce. Fonte: National Cancer Institute.

L'esempio sopra riportato illustra un pedigree relativo alla mutazione *BRCA1*. La presenza ricorrente di tumori mammari e ovarici nella linea materna costituisce un indicatore significativo per proporre test genetici ai familiari non ancora affetti e per pianificare programmi di sorveglianza intensiva.

Strumenti come **PedigreeJS**, **Progeny** o moduli basati su standard **FHIR** permettono oggi di integrare i genogrammi nei fascicoli sanitari elettronici, supportando una medicina predittiva basata sui dati e orientata al rischio familiare.

2.6 Considerazioni finali

La rappresentazione grafica della storia clinica familiare ha subito un'evoluzione significativa: da semplici alberi genealogici utilizzati per scopi storici o legali, si è

giunti a strumenti complessi e codificati come i genogrammi clinici. Questi ultimi non solo mappano le relazioni familiari, ma forniscono anche una visione dettagliata delle condizioni patologiche ricorrenti, dei pattern di ereditarietà e delle interazioni tra fattori genetici e ambientali.

L'adozione di simbologie standardizzate, l'utilizzo di strumenti digitali interattivi e l'integrazione nei sistemi informativi sanitari hanno reso il genogramma un dispositivo fondamentale per la prevenzione, la diagnosi precoce e la medicina personalizzata. In ambito clinico, esso consente di identificare i soggetti a rischio, supportare decisioni terapeutiche informate e migliorare la comunicazione tra operatori sanitari e pazienti.

Nel complesso, il genogramma si configura oggi come una risorsa insostituibile nella pratica medica moderna, in grado di tradurre la storia familiare in conoscenza clinica strutturata e immediatamente fruibile.

Capitolo 3

Standard e strumenti per la gestione di pedigree genetici

Introduzione

La rappresentazione e gestione dei pedigree genetici in ambito clinico richiede l'adozione di standard consolidati che garantiscano interoperabilità, accuratezza e riproducibilità delle analisi. Nel corso degli anni, la comunità scientifica internazionale ha sviluppato formati standardizzati, algoritmi di calcolo del rischio e strumenti software specifici per supportare i professionisti sanitari nella valutazione del rischio genetico.

Questo capitolo presenta una panoramica completa degli standard attualmente utilizzati nel settore, con particolare focus sui formati *BOADICEA* e *CanRisk*, sulla nomenclatura standardizzata per la rappresentazione grafica dei pedigree, e sugli strumenti software disponibili per la loro gestione. L'analisi si estende dalle soluzioni storiche e tradizionali fino alle moderne piattaforme *web-based*, evidenziando vantaggi, limitazioni e aree di miglioramento.

La comprensione di questi standard è fondamentale per inquadrare il contributo innovativo del sistema sviluppato in questa tesi, che si propone di integrare la gestione dei dati clinici dei pazienti con un editor di pedigree conforme agli standard internazionali, fornendo un *workflow* completo dalla raccolta dati all'esportazione in formati compatibili con i principali algoritmi di calcolo del rischio genetico.

3.1 BOADICEA: un modello predittivo per il rischio oncologico ereditario

BOADICEA è un modello statistico sviluppato nei primi anni 2000 dal **Centre for Cancer Genetic Epidemiology** dell'Università di Cambridge. Il suo obiettivo principale è fornire stime affidabili del rischio che un individuo possa sviluppare un tumore al seno o alle ovaie, tenendo conto sia della **storia familiare** che di eventuali **risultati di test genetici**.

Nel tempo, l'algoritmo è diventato uno strumento di riferimento nella **consulenza genetica** e nella **medicina preventiva**, grazie alla capacità di analizzare pedigree familiari anche molto complessi. BOADICEA consente, ad esempio, di stimare la probabilità che un individuo sia portatore di mutazioni in geni noti per l'alta penetranza tumorale, come *BRCA1* e *BRCA2*, e calcolare il rischio cumulativo di sviluppare determinate neoplasie nel corso della vita.

La robustezza del modello si basa su dati derivati da ampie coorti epidemiologiche e su analisi di segregazione genetica, integrando anche fattori come l'età alla diagnosi dei parenti affetti e l'eventuale presenza di test genetici. Nelle sue versioni più recenti, BOADICEA include anche geni come *PALB2*, *ATM* e *CHEK2*, oltre a marcatori istopatologici come *ER*, *PR* e *HER2*.

I dati familiari devono essere inseriti nel sistema usando un formato strutturato specifico, come lo standard **BOADICEA v4**. Questo standard codifica ogni membro della famiglia con una serie di informazioni: sesso, età, diagnosi oncologiche, esiti dei test genetici e altri parametri clinici. Ogni persona occupa una riga del file, consentendo al software di ricostruire il pedigree familiare e di eseguire calcoli accurati dei rischi.

Nella pratica clinica, BOADICEA viene oggi impiegato da consulenti genetici, oncologi e ricercatori per:

- identificare soggetti ad alto rischio genetico;
- supportare decisioni su test genetici da proporre;
- personalizzare percorsi di screening e sorveglianza;
- contribuire a studi epidemiologici e predittivi.

Tuttavia, non mancano alcune limitazioni. Il modello è stato costruito principalmente su popolazioni di origine europea e si concentra su un numero ristretto di geni ad alta penetranza. Inoltre, tiene in considerazione in misura limitata i fattori ambientali o di stile di vita, e richiede una certa esperienza per l'interpretazione clinica dei risultati.

In ogni caso, BOADICEA rappresenta uno dei più avanzati strumenti oggi disponibili per la valutazione del rischio oncologico su base familiare, ed è frequentemente integrato all'interno di portali come **CanRisk**, che ne facilitano l'utilizzo in ambito clinico attraverso interfacce web intuitive.

3.2 CanRisk: piattaforma avanzata per la valutazione del rischio oncologico

CanRisk è una piattaforma di nuova generazione sviluppata a partire dall'algoritmo BOADICEA, dal quale eredita le fondamenta scientifiche, ma che estende notevolmente il campo di applicazione. Realizzata dal **Centre for Cancer Genetic Epidemiology** dell'Università di Cambridge, CanRisk rappresenta oggi uno degli strumenti più completi per stimare il rischio individuale di sviluppare forme tumorali ereditarie, in particolare il tumore al seno, alle ovaie e alla prostata.

A differenza del modello originario, centrato principalmente sulle mutazioni nei geni *BRCA1* e *BRCA2*, CanRisk adotta un approccio **multifattoriale e multigenico**, integrando l'analisi di oltre dieci geni ad alta o moderata penetranza, tra cui *PALB2*, *ATM*, *CHEK2*, *RAD51C* e *HOXB13*. Inoltre, la piattaforma considera **fattori di rischio clinici, riproduttivi, ambientali e legati allo stile di vita**, come età del menarca, numero di figli, BMI e assunzione di terapie ormonali.

Una delle innovazioni più rilevanti introdotte da CanRisk è l'integrazione del **Polygenic Risk Score (PRS)**, ovvero un indice complessivo di rischio basato sull'accumulo di varianti genetiche comuni, ciascuna con piccolo effetto, che insieme possono influenzare significativamente la predisposizione individuale.

La piattaforma è accessibile tramite il sito web https://canrisk.org/, che consente l'inserimento guidato dei dati familiari e clinici attraverso un'interfaccia intuitiva. CanRisk è inoltre disponibile via API per l'integrazione in sistemi clinici

esistenti, permettendo l'automatizzazione delle valutazioni di rischio nei flussi di lavoro ospedalieri.

Limiti e considerazioni pratiche

Nonostante le sue potenzialità, CanRisk presenta alcuni problemi. Dal lato tecnico, raccogliere tutti i dati necessari è complicato e richiede molto tempo, oltre a dover essere molto precisi per ottenere stime affidabili. I risultati dipendono molto da quanto sono complete le informazioni che si forniscono, e siccome non esistono procedure standard uguali per tutti gli ospedali, ogni struttura sanitaria può raccogliere i dati in modo diverso.

Dal punto di vista scientifico, il modello è stato testato principalmente su persone di origine europea, quindi potrebbe essere meno preciso se usato su altri gruppi etnici. Inoltre, considera solo in parte fattori come l'ambiente e le condizioni socioeconomiche, e non riesce ancora a modellare bene le complesse interazioni tra geni e ambiente. Infine, l'utilizzo clinico di CanRisk richiede una certa **formazione specialistica** per interpretare correttamente i risultati, e l'integrazione all'interno dei sistemi informativi ospedalieri comporta costi organizzativi e tecnici. Sebbene la piattaforma sia disponibile gratuitamente, l'adozione su larga scala implica investimenti in termini di interoperabilità, aggiornamenti, gestione della privacy e sicurezza dei dati genetici sensibili.

Nonostante questi limiti, CanRisk rappresenta uno strumento estremamente avanzato per la **medicina personalizzata**, consentendo valutazioni predittive precise e supportando decisioni cliniche basate sul rischio individuale.

3.3 Altri Strumenti Web Esistenti per la Gestione di Pedigree

Strumenti Professionali Attuali

Progeny Genetics Pedigree Tool Il *Progeny Genetics Pedigree Tool* (https://pedigree.progenygenetics.com/) è uno degli strumenti più utilizzati in ambito clinico-professionale per la gestione dei pedigree genetici. Pensato principalmente per genetisti clinici e consulenti genetici, offre un editor avanzato conforme agli

standard BOADICEA, con la possibilità di integrarsi direttamente con i laboratori diagnostici. Tra i principali punti di forza si segnalano l'interfaccia altamente professionale, la possibilità di esportare i dati in più formati e la validazione clinica del sistema. Tuttavia, si tratta di uno strumento a pagamento, con costi spesso elevati, e caratterizzato da una curva di apprendimento piuttosto ripida, ovvero lo strumento, pur offrendo funzionalità avanzate, richiede un periodo iniziale significativo di apprendimento da parte dell'utente, dovuto alla complessità dell'interfaccia e alla ricchezza delle opzioni disponibili. questo può rappresentare una barriera iniziale per alcuni utenti.

GenoPro *GenoPro* (https://genopro.com/it/) è un software che funziona sia su computer che via web, nato principalmente per i genealogisti ma utilizzato anche da professionisti medici per rappresentare le strutture familiari. Offre funzioni genealogiche classiche ma anche strumenti per inserire dati medici, generare report su misura e gestire informazioni anagrafiche dettagliate. La sua flessibilità lo rende utilizzabile in contesti diversi. Tuttavia, l'interfaccia può risultare complessa per chi non lo conosce, e l'impostazione generale privilegia ancora la genealogia tradizionale rispetto all'uso clinico specifico, il che limita l'integrazione con strumenti di analisi genetica specializzati.

CEGAT Pedigree Chart Designer Il *CEGAT Pedigree Chart Designer* (https://cegat.com/diagnostics/general-information/pedigree-chart-designer/) è uno strumento sviluppato per supportare la rappresentazione dei pedigree all'interno dei servizi diagnostici offerti da CEGAT, un laboratorio specializzato in analisi genetiche. È pensato specificamente per l'utilizzo da parte di laboratori e cliniche, con un'interfaccia focalizzata sull'integrazione con i test genetici offerti dall'azienda. Tra i suoi punti di forza vi è la conformità agli standard internazionali e la possibilità di operare all'interno di un workflow integrato tra laboratorio e clinica, facilitando la gestione dei dati genetici nel contesto diagnostico. Tuttavia, lo strumento risulta fortemente legato all'ecosistema CEGAT, e le sue funzionalità sono limitate rispetto a soluzioni più generiche o personalizzabili, rendendolo meno adatto in contesti che richiedono flessibilità o indipendenza da un fornitore specifico.

Strumenti Obsoleti e Tradizionali

Software Desktop Legacy e Metodi Tradizionali Accanto alle soluzioni moderne, sono ancora presenti software e approcci tradizionali per la gestione dei pedigree, ormai considerati largamente obsoleti. Questi vecchi programmi desktop erano molto diffusi per costruire alberi genealogici in contesto medico, ma oggi non ricevono più aggiornamenti, hanno problemi di compatibilità con i sistemi operativi moderni e offrono opzioni di esportazione molto limitate. Tuttavia, alcune strutture sanitarie continuano a utilizzarli per consultare archivi storici non ancora migrati verso piattaforme più aggiornate.

Oltre ai software, permangono anche metodi tradizionali come l'uso di carta e matita, ancora adottati per la realizzazione di bozze iniziali o per raccogliere informazioni in fase di colloquio clinico. In alternativa, alcuni professionisti ricorrono a software generici come PowerPoint o Visio per costruire rappresentazioni semplici dei pedigree. Questi metodi, tuttavia, non rispettano gli standard grafici internazionali, risultano poco adatti alla condivisione e soggetti a errori manuali, rendendoli inadatti a un uso clinico strutturato e ripetibile.

3.4 PedigreeJS: Libreria di Visualizzazione

Origini e Sviluppo

PedigreeJS è una libreria JavaScript open source sviluppata dal Centre for Cancer Genetic Epidemiology (CCGE-BOADICEA) dell'Università di Cambridge, già noto per aver creato i modelli BOADICEA e CanRisk. La libreria nasce con l'obiettivo di offrire uno strumento di visualizzazione standardizzato e interoperabile, pensato per integrarsi con i formati di dati utilizzati nel calcolo del rischio genetico.

Caratteristiche Tecniche

L'architettura della libreria si basa su JavaScript (versione ES6+) e sfrutta la potenza di D3.js per la manipolazione del DOM e il rendering grafico in formato SVG. Questo approccio garantisce compatibilità con tutti i browser moderni e supporto per un design responsive, adattabile a diversi dispositivi.

Tra le funzionalità principali si annoverano un editor interattivo per la creazione e modifica di pedigree tramite interfaccia grafica, il rispetto rigoroso degli standard internazionali di rappresentazione genealogica, la possibilità di navigare tra strutture familiari complesse tramite zoom e pan, e un sistema interno di validazione della coerenza dei dati inseriti.

Integrazione con Standard Medici

PedigreeJS supporta nativamente l'importazione e l'esportazione nei formati BOADICEA v4 e CanRisk v4, inclusi i metadati estesi necessari per l'integrazione con algoritmi di rischio genetico. Utilizza un formato interno basato su JSON per la persistenza dei dati e offre anche compatibilità con formati impiegati nella ricerca genetica (es. linkage format). È inoltre predisposta per l'interazione con servizi web e API esterne, tra cui quelle offerte dal modello BOADICEA, risultando così facilmente integrabile in sistemi clinici più ampi.

Caratteristiche Distintive

Tra i principali punti di forza della libreria vi sono la gratuità, essendo distribuita sotto licenza open source, e l'affidabilità garantita dall'istituzione accademica che l'ha sviluppata. La manutenzione attiva e la disponibilità di una documentazione tecnica completa ne facilitano l'adozione da parte degli sviluppatori. Va sottolineato che PedigreeJS ha un focus esclusivo sulla visualizzazione: non esegue direttamente calcoli di rischio, ma si limita a preparare e rappresentare i dati da esportare verso strumenti di analisi.

Utilizzo nel Progetto di Tesi

Nel contesto del progetto di tesi, la scelta di PedigreeJS è stata motivata da diversi fattori. Innanzitutto, la piena aderenza agli standard medici internazionali ha garantito una rappresentazione clinicamente valida delle strutture familiari. Inoltre, la sua affidabilità, derivante dalla provenienza accademica, e la flessibilità nella personalizzazione hanno permesso un'integrazione efficace all'interno dell'applicazione sviluppata.

PedigreeJS è stato adottato come componente principale per la visualizzazione dei pedigree. Attorno ad esso è stato costruito un wrapper personalizzato per semplificare l'interfaccia destinata agli utenti clinici, ed è stata aggiunta una gestione dei pazienti collegata a un sistema backend. Infine, è stata implementata la funzionalità di esportazione automatica dei dati in formato BOADICEA e CanRisk, in modo da permettere successive analisi esterne.

Durante una serie di incontri con i medici del Centro per le Malattie Neurodegenerative (CMND) dell'Università di Bari "Aldo Moro", presso l'Ospedale Cardinale Panico di Tricase, sono emerse alcune necessità cliniche non coperte dalla versione originale della libreria. In particolare, è stato evidenziato che non era possibile aggiungere figli a un singolo individuo senza associare un partner, né rappresentare coppie di partner non associate a figli. Per soddisfare tali esigenze, sono state introdotte due modifiche strutturali alla logica della libreria: la possibilità di associare figli a un nodo singolo e la creazione di relazioni di coppia indipendenti dalla presenza di figli.

Ho inoltre apportato modifiche generali alla libreria per migliorarne la flessibilità e l'adattamento al contesto clinico, oltre a garantire una migliore integrazione con l'architettura dell'applicazione. Questi aggiornamenti hanno reso PedigreeJS più efficace nella rappresentazione di situazioni familiari complesse e più user-friendly per gli operatori sanitari.

Questa scelta ha portato un significativo valore aggiunto al sistema, garantendo coerenza con gli standard di settore, manutenibilità nel tempo grazie agli aggiornamenti della libreria, e credibilità attraverso l'utilizzo di una tecnologia validata dalla comunità scientifica.

Conclusioni

In questo capitolo sono stati analizzati gli standard internazionali per la rappresentazione dei pedigree genetici, gli strumenti software disponibili per la loro gestione e la libreria PedigreeJS, scelta come base per la visualizzazione interattiva all'interno del sistema sviluppato. È emersa l'importanza di adottare soluzioni conformi agli standard medici, interoperabili e adattabili al contesto clinico, in modo da garantire coerenza, affidabilità e facilità di integrazione.

Nel capitolo successivo verranno descritte nel dettaglio la progettazione e l'implementazione dell'applicazione realizzata, evidenziando le scelte architetturali, le funzionalità implementate e le modalità con cui gli strumenti e le tecnologie illustrati in questa sezione sono stati integrati in un sistema completo e funzionale.

Capitolo 4

Strumenti e tecnologie utilizzate

Negli ultimi anni, la crescente complessità delle applicazioni software ha reso indispensabile l'impiego di strumenti e tecnologie capaci di supportare un ciclo di sviluppo strutturato, efficiente e manutenibile. In questo capitolo verranno presentati i principali ambienti, framework, librerie e architetture impiegati durante lo sviluppo dell'applicazione.

L'obiettivo è fornire una panoramica dell'infrastruttura tecnologica adottata, delle scelte progettuali effettuate e delle modalità operative che hanno guidato l'implementazione. Il progetto, sviluppato interamente in locale, ha richiesto l'integrazione di componenti eterogenei — frontend, backend e database — coordinati attraverso una comunicazione RESTful.

Il risultato è una piattaforma web moderna e modulare che consente la gestione dei pazienti e la visualizzazione interattiva dei genogrammi clinici. Il progetto rappresenta un caso d'uso concreto di applicazione full-stack, sviluppata secondo pratiche consolidate della moderna ingegneria del software.

4.1 Ambiente di sviluppo

Lo sviluppo dell'applicazione è avvenuto interamente in ambiente locale, su sistema operativo **Windows 11**, versione `10.0`. Sono stati utilizzati strumenti differenti per ciascun livello dell'architettura applicativa.

Per il **backend**, basato su Spring Boot, è stato impiegato **Eclipse IDE**, che ha permesso una gestione strutturata del progetto Maven e l'integrazione delle

dipendenze tramite il file `pom.xml`.

Il **frontend**, realizzato con Angular, è stato sviluppato utilizzando **Visual Studio Code**, editor leggero e altamente personalizzabile, con estensioni per TypeScript, HTML, CSS e Bootstrap.

Per la gestione del **database PostgreSQL**, è stato utilizzato **pgAdmin**, interfaccia grafica che ha facilitato la creazione delle tabelle e l'esecuzione delle query durante le fasi di test.

Durante lo sviluppo, è stato fatto ampio uso di **Postman** per testare le API REST lato backend, fase essenziale per assicurarsi che i servizi offrissero risposte corrette prima dell'integrazione con il frontend.

Principali versioni degli strumenti utilizzati:

- **Node.js**: v22.14.0
- **npm**: v10.9.2
- **Java**: 17.0.6 LTS
- **Angular**: 19.0.5
- **Angular CLI**: 19.0.6
- **TypeScript**: 5.6.3

Il codice sorgente è stato gestito tramite **Git** e versionato su **GitHub**.

4.2 Architettura generale del sistema

L'applicazione è stata progettata secondo un'architettura **client-server a tre livelli**, con una netta separazione tra frontend, backend e database. Tale struttura consente maggiore scalabilità, manutenibilità del codice e riusabilità dei componenti.

- Il **frontend**, sviluppato con **Angular**, si occupa dell'interazione con l'utente e dell'invio di richieste HTTP al backend.
- Il **backend**, implementato in **Spring Boot**, espone API **RESTful** responsabili dell'elaborazione delle richieste e della comunicazione con il database.

- Il **database PostgreSQL** conserva i dati relativi a pazienti, caregiver, pedigree e associazioni.

La comunicazione tra frontend e backend avviene tramite protocollo `HTTP` con scambio di dati in formato `JSON`. Per motivi di sviluppo locale, sono state configurate porte diverse:

- Angular: `http://localhost:4200`
- Spring Boot: `http://localhost:8085`
- PostgreSQL: `localhost:5432`

L'architettura è di tipo **stateless**, e ogni richiesta HTTP è trattata indipendentemente, secondo i principi REST.

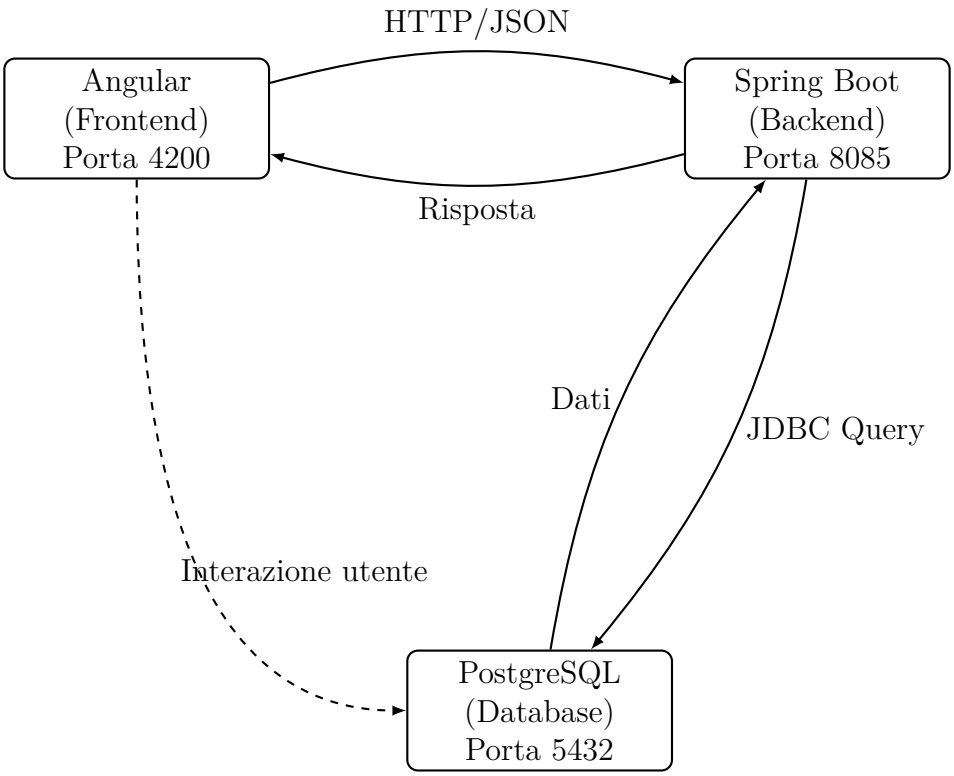

Figura 4.1. Architettura circolare dell'applicazione: flusso bidirezionale tra frontend, backend e database.

4.3 Frontend: tecnologie e librerie utilizzate

Il frontend dell'applicazione è stato sviluppato utilizzando il framework **Angular**, scelto per la sua struttura modulare e la capacità di gestire applicazioni complesse e reattive. Angular consente una chiara separazione tra logica di presentazione e logica di business lato client.

Angular

Per lo sviluppo dell'interfaccia utente è stato impiegato Angular, nella versione 19.0.5. Questo framework ha permesso di creare un'applicazione modulare, basata su componenti riutilizzabili e servizi centralizzati, facilitando la gestione della logica lato client. La navigazione tra le diverse sezioni dell'app è stata implementata tramite il sistema di routing integrato, mentre la comunicazione con il backend avviene grazie al modulo HttpClient, che consente di inviare e ricevere richieste HTTP in formato JSON, secondo il paradigma REST.

Bootstrap e ng-bootstrap

Per lo stile dell'interfaccia è stato impiegato **Bootstrap 5.3.3**, combinato con `ng-bootstrap`, che integra i componenti Bootstrap in modo nativo con Angular. Queste librerie hanno consentito la realizzazione di un'interfaccia moderna, responsive e compatibile con i principali browser.

jQuery

Alcune funzionalità della libreria **PedigreeJS** si basano su **jQuery** (versione 3.7.1). Sebbene l'utilizzo diretto di jQuery in Angular sia generalmente sconsigliato, è stato necessario includerla per garantire la compatibilità dei grafi genealogici interattivi.

Altre librerie

Le icone utilizzate sono fornite da **FontAwesome** (versione 6.7.2). Il sistema di notifiche è implementato con **ngx-toastr** (versione 19.0.0), mentre per la generazione di documenti PDF è utilizzata **jsPDF** (versione 3.0.1).

Esempio di chiamata API

Di seguito è riportato un esempio del servizio `caregiver.service.ts`, che effettua una richiesta POST per creare un nuovo caregiver:

```
createCaregiver(caregiver: CaregiverRequestDTO): Observable<ApiResponse<any>> {
  return this.http.post<ApiResponse<any>>(this.apiUrl, caregiver);
}
```

Listing 4.1. Chiamata POST al backend per la creazione di un caregiver

Modello dati: `CaregiverResponse`

L'interfaccia `CaregiverResponse` rappresenta la forma dei dati di un caregiver restituiti dal backend:

```
export interface CaregiverResponse {
    id: number;
    firstName: string;
```

```
    lastName: string;
    email: string;
    phone: string;
    address: string;
    dateOfBirth: string;
    fiscalCode: string;
    gender: string;
}
```

Listing 4.2. Interfaccia TypeScript per il modello `CaregiverResponse`

4.4 Backend: architettura e tecnologie utilizzate

Il backend dell'applicazione è stato sviluppato utilizzando **Spring Boot**, una piattaforma moderna per la creazione di applicazioni Java basate su architettura RESTful.

Architettura a livelli

Il backend è organizzato secondo il paradigma **MVC (Model-View-Controller)**:

- **Controller**: riceve le richieste HTTP e le instrada verso i servizi appropriati.
- **Service**: contiene la logica di business.
- **Repository**: interagisce con il database tramite Spring Data JPA.
- **Entity e DTO**: mappano le tabelle del database e i dati scambiati tramite API.

Testing delle API con Postman

Durante lo sviluppo, **Postman** è stato fondamentale per testare le API esposte dal backend:

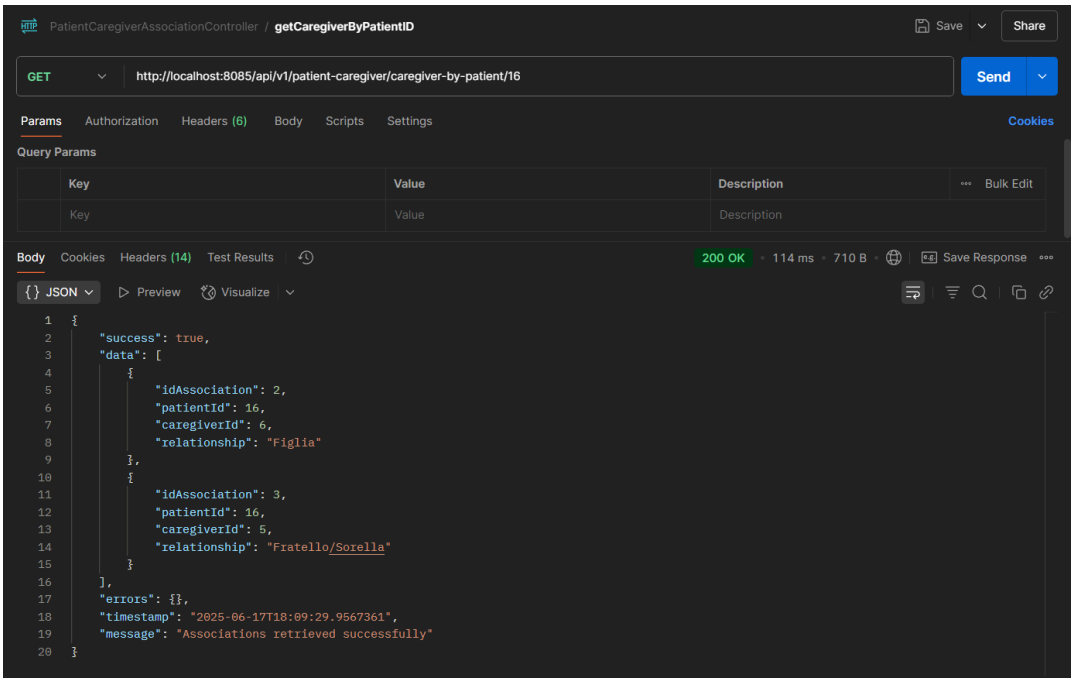

Figura 4.2. Chiamata API tramite Postman all'endpoint `/caregiver-by-patient/16`.

Il test restituisce un oggetto JSON con: il campo **success** (esito operazione), il campo **data** (lista di oggetti PatientCaregiverAssociationDTO), il campo **message** (debugging) ed eventuali **errors**.

Struttura del progetto

Il progetto backend segue l'architettura MVC: i **controller** gestiscono gli endpoint REST, i **service** implementano la logica di business, i **repository** forniscono le interfacce JPA, i **dto** definiscono gli oggetti di trasferimento dati e le **entity** mappano le tabelle del database.

Best Practice – Perché usare il Model-View-Controller (MVC)

L'architettura **Model-View-Controller (MVC)** è una delle più utilizzate nello sviluppo software per applicazioni web e desktop perché consente di separare nettamente le responsabilità tra componenti:

- **Model** – Gestisce la logica di accesso ai dati e rappresenta lo stato dell'applicazione.
- **View** – Si occupa della presentazione dei dati all'utente.
- **Controller** – Riceve gli input dell'utente e li traduce in operazioni da eseguire sul model o sulla view.

Questa separazione porta a numerosi benefici:

- Maggiore **manutenibilità** del codice;
- Facilitazione dei **test unitari** e dell'isolamento delle componenti;
- Supporto alla **collaborazione** tra più team (es. frontend/backend);
- Possibilità di riutilizzo del modello dati in contesti diversi (es. Web, API REST, Mobile).

Questa suddivisione ha reso il progetto leggibile, scalabile e conforme alle best practices di ingegneria del software in ambito Java Enterprise.

4.5 Database e gestione dei dati

La componente dati dell'applicazione è stata gestita tramite **PostgreSQL**, un sistema di database relazionale open source, noto per l'affidabilità, la robustezza

e la compatibilità con il linguaggio SQL standard. L'istanza del database è stata configurata in locale sulla porta `5432`, con accesso e gestione tramite interfaccia grafica **pgAdmin**.

Le tabelle rappresentano le entità principali: `pazienti`, `caregiver`, `pedigree`, e le associazioni tra questi soggetti. Il campo `data` della tabella `pedigree` è definito come `jsonb` per memorizzare il contenuto grafico generato tramite **PedigreeJS**.

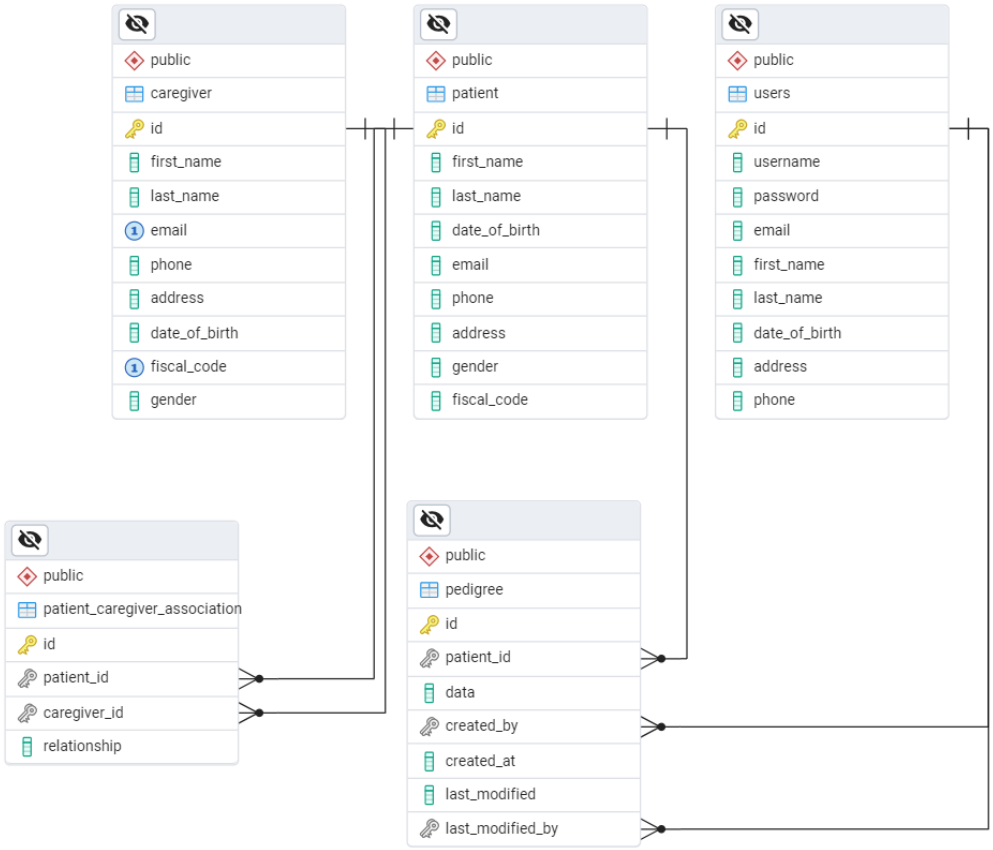

Figura 4.3. Diagramma ER del database: entità, attributi e relazioni dell'applicazione.

4.6 Versionamento del codice: Git e GitHub

È stato adottato il sistema **Git** in combinazione con **GitHub** per gestire l'evoluzione del codice sorgente.

Repository GitHub: https://github.com/D4n1l0S4/tesi-app

Il flusso di lavoro segue il **GitHub Flow**:

- **Creazione branch** - ogni funzionalità parte da un nuovo branch
- **Sviluppo** - codice scritto localmente con commit descrittivi
- **Push remoto** - branch caricati su GitHub
- **Pull Request** - revisione cambiamenti
- **Merge** - unione nel branch principale

Figura 4.4. Diagramma del flusso di lavoro GitHub Flow: creazione, modifica e integrazione del codice.

Grazie a GitHub è stato possibile avere backup online, tornare indietro in caso di bug, documentare lo sviluppo e tenere separati gli ambienti di lavoro.

Considerazioni finali

L'integrazione delle tecnologie descritte ha permesso la realizzazione di un'applicazione robusta, modulare e facilmente estendibile. Ogni componente è stato scelto per soddisfare requisiti specifici in termini di usabilità, scalabilità e mantenibilità.

Capitolo 5

Analisi funzionale e comportamento dell'applicazione

5.1 Panoramica generale dell'applicazione

L'applicazione sviluppata fornisce uno strumento informatico moderno per la **gestione dei pazienti** e la **rappresentazione grafica dei genogrammi clinici**, con particolare attenzione alle relazioni familiari e alle condizioni ereditarie.

Il sistema si configura come una *piattaforma web full-stack* che consente:

- l'inserimento e consultazione dei dati anagrafici di pazienti e caregiver;
- la creazione e modifica interattiva dei genogrammi mediante editor visuale integrato;
- la gestione delle associazioni paziente-caregiver con specifica del tipo di relazione;
- l'**esportazione dei pedigree** in diversi formati per documentazione e analisi cliniche.

L'applicazione è rivolta a figure professionali del settore sanitario:

- **Genetisti clinici**, per la creazione di pedigree durante la consulenza genetica, con esportazione verso strumenti come BOADICEA o CanRisk;

- **Oncologi e specialisti**, per consultare la storia familiare e valutare predisposizioni ereditarie;
- **Assistenti clinici**, per gestire l'anagrafica e predisporre la documentazione per le visite.

Dal punto di vista funzionale, l'applicazione offre una **dashboard** iniziale, un'interfaccia **intuitiva e responsive**, funzionalità di **ricerca globale** e un editor interattivo per la costruzione grafica del pedigree con pannello laterale per la modifica dei dettagli clinici.

L'adozione dell'approccio digitale ha permesso la **dematerializzazione del processo** precedentemente gestito su carta, migliorando leggibilità, condivisione tra specialisti ed efficienza nelle consulenze genetiche. L'archiviazione digitale sicura consente accesso rapido ai dati e migliore tracciabilità.

L'architettura, basata su *Angular* per il frontend e *Spring Boot* per il backend, garantisce separazione netta tra interfaccia e logica applicativa. I dati sono memorizzati in **PostgreSQL**, con tipo `jsonb` per la gestione strutturata delle informazioni genealogiche. Il sistema è progettato per essere **scalabile, manutenibile e sicuro**, con validazioni multiple e rispetto dei principi di integrità referenziale.

L'applicazione fornisce uno strumento concreto per supportare la *medicina personalizzata e preventiva*, agevolando l'analisi dei pattern familiari e la documentazione clinica standardizzata.

5.2 Requisiti dell'applicazione

5.2.1 Requisiti funzionali

I requisiti funzionali descrivono le principali operazioni che il sistema deve svolgere per rispondere alle necessità degli utenti target. Le funzionalità sono progettate considerando il contesto clinico e la necessità di documentare la storia familiare dei pazienti.

RF1 – Gestione anagrafica pazienti

- Creazione, visualizzazione, modifica ed eliminazione dei pazienti

- Validazione del codice fiscale come campo univoco e obbligatorio
- Ricerca e filtraggio tramite nome, codice fiscale o altri attributi
- Paginazione e ordinamento per gestione efficiente di grandi volumi di dati

RF2 – Gestione caregiver e associazioni

- Inserimento e aggiornamento dei dati anagrafici del caregiver
- Ricerca caregiver tramite codice fiscale
- Associazione caregiver-paziente con specifica del tipo di relazione
- Gestione dinamica: associazione di più caregiver a un singolo paziente

RF3 – Creazione e gestione dei pedigree

- Editor interattivo per la costruzione dell'albero genealogico familiare
- Pannello laterale per la modifica dei dati clinici e anagrafici dei membri, inclusi:
  - diagnosi oncologiche con età di insorgenza
  - storia riproduttiva (aborti, nati morti, interruzioni volontarie)
  - stato di adozione e gestione dei gemelli
- Controllo navigazione: prevenzione perdita dati tramite guard che bloccano il cambio pagina con modifiche non salvate
- Monitoraggio in tempo reale delle modifiche con indicatori visivi
- Salvataggio persistente in formato `jsonb` con validazione coerenza dati genealogici

RF4 – Esportazione dei dati

- Esportazione pedigree in formato PNG/SVG per documentazione
- Esportazione in formato JSON per backup e interoperabilità
- Esportazione in formato BOADICEA v4 e CanRisk v4.0 per valutazione rischio genetico

5.2.2 Requisiti non funzionali

I requisiti non funzionali descrivono le caratteristiche qualitative del sistema relative a prestazioni, usabilità, compatibilità, sicurezza e manutenibilità, fondamentali per garantire efficienza e affidabilità in contesto reale.

RNF1 – Performance

- Le API devono rispondere entro 2 secondi nella maggior parte dei casi d'uso
- Rendering fluido anche con strutture genealogiche complesse (>50 persone) tramite ottimizzazione PedigreeJS e D3.js

RNF2 – Usabilità

- Interfaccia responsiva e navigabile su dispositivi desktop e mobile
- Layout adattivo alle dimensioni dello schermo per fruizione ottimale
- Validazione in tempo reale nei form (codice fiscale, campi obbligatori)
- Indicatori di stato chiari, messaggi di errore descrittivi e workflow intuitivi

RNF3 – Compatibilità

- Compatibilità con principali browser moderni (Chrome, Firefox, Edge)
- Interfaccia fruibile con diverse risoluzioni e sistemi operativi
- Esportazione dati conforme agli standard medici internazionali (CanRisk, BOADICEA)

RNF4 – Sicurezza

- Validazione dell'input per prevenire attacchi comuni tramite Spring Data JPA con query methods e prepared statements automatici
- Gestione dati rispettosa dei principi di integrità e consistenza del database

RNF5 – Manutenibilità e scalabilità

- Codice organizzato in moduli separati e riutilizzabili (componenti Angular, pacchetti backend)
- Architettura supporta aggiunta nuove funzionalità senza modifiche drastiche alla struttura esistente

RNF6 – Affidabilità e recuperabilità

- Persistenza dati garantita anche in caso di ricaricamento o interruzione sessione
- Dati parzialmente modificati segnalati tramite indicatori visivi per prevenire perdite accidentali

5.3 Modello dati e progettazione del database

La progettazione del database garantisce la persistenza strutturata delle informazioni cliniche e genealogiche. È stato adottato un modello **relazionale** bassu su *PostgreSQL*, con campi in formato `JSONB` per la gestione flessibile dei dati complessi dei pedigree.

5.3.1 Struttura delle entità principali

Entità `patient` L'entità patient rappresenta il nucleo anagrafico del sistema, raccogliendo le principali informazioni personali dei pazienti. Tra i campi obbligatori troviamo il nome, il cognome, la data di nascita, il genere e il codice fiscale, mentre email, numero di telefono e indirizzo sono considerati opzionali. Sono stati inoltre applicati vincoli di unicità sul codice fiscale, insieme a meccanismi di validazione per garantire la correttezza del genere inserito.

Entità `caregiver` e associazioni Gestisce i caregiver con struttura simile ai pazienti ma con campi di contatto obbligatori per garantire reperibilità. La tabella di associazione `patient_caregiver_association` implementa una relazione molti-a-molti con specifica del tipo di legame (es. "Genitore", "Familiare", "Amico/a").

Entità `pedigree` Contiene le informazioni genealogiche in formato `JSONB`, supportando l'integrazione diretta con l'editor PedigreeJS. Include metadati di creazione e modifica collegati agli utenti del sistema.

Entità `users` Gestisce le credenziali di accesso e i dati degli operatori autorizzati, con vincoli di unicità su username ed email e password hashate con BCrypt.

5.3.2 Struttura dei dati genealogici

Il campo `data` della tabella `pedigree` contiene oggetti JSON che descrivono le caratteristiche di ciascun membro della famiglia:

```
[
  {
    "sex": "M",
    "name": "LionelMessi18",
    "age": 37,
    "proband": true,
    "display_name": "Lionel",
    "father": "FjVS",
    "mother": "JWOP",
    "diabetes_diagnosis_age": 1,
    "breast_cancer_diagnosis_age": null
  }
]
```

Listing 5.1. Esempio di struttura JSONB per un pedigree

5.3.3 Considerazioni progettuali

Il modello garantisce:

- **Flessibilità**: l'impiego di `jsonb` consente di modellare strutture genealogiche complesse (adozioni, diagnosi multiple, gemelli, storia riproduttiva) difficilmente gestibili con modelli relazionali classici.
- **Performance**: è previsto l'uso di indici GIN sui campi `jsonb` per ottimizzare le query di ricerca sui pedigree.

- **Integrità**: vincoli di chiave esterna garantiscono coerenza tra pazienti, pedigree, caregiver e utenti.
- **Scalabilità**: la struttura è pensata per essere estendibile con ulteriori attributi clinici, meccanismi di log, e interfacce con sistemi esterni.
- **Sicurezza**: separazione tra dati clinici (pazienti e pedigree) e dati di sistema (utenti), con validazioni su tutti i campi critici.

5.4 Interfaccia utente e flusso funzionale

Il frontend dell'applicazione, sviluppato con *Angular*, rappresenta il punto di contatto diretto con gli utenti finali e deve garantire un'esperienza fluida nella gestione dei pazienti, nella creazione dei pedigree e nell'esportazione dei dati clinici.

L'architettura del frontend è stata progettata seguendo i principi di modularità, riusabilità e separazione delle responsabilità, caratteristici del framework Angular. Ogni funzionalità è implementata attraverso componenti specializzati che comunicano tramite servizi centralizzati.

5.4.1 Architettura del frontend

L'applicazione Angular è strutturata secondo un'architettura *component-based*, dove ogni elemento dell'interfaccia utente è incapsulato in componenti riutilizzabili. La comunicazione con il backend avviene attraverso servizi dedicati che implementano il pattern *dependency injection*.

Struttura modulare e routing L'applicazione adotta l'approccio *standalone components* (Angular 19), eliminando la necessità di moduli `NgModule`. Il sistema di routing gestisce le diverse sezioni dell'applicazione:

```
const routes: Routes = [
  {
    path: '',
    component: AdminComponent,
    children: [
      { path: 'analytics', loadComponent: () =>
    import('./demo/dashboard/dash-analytics.component') },
```

```
      { path: 'patients', loadComponent: () =>
    import('./pages/patients/patient-list/patient-list.component') },
      { path: 'pedigree-viewer', loadComponent: () =>
    import('./components/pedigree-viewer/pedigree-viewer.component'),
        canDeactivate: [UnsavedChangesGuard] }
    ]
  }
];
```

Listing 5.2. Configurazione delle rotte principali

Il routing implementa *lazy loading* per ottimizzare le performance e include *guard* per prevenire la perdita accidentale di dati nell'editor di pedigree.

Servizi Angular e comunicazione HTTP I servizi Angular rappresentano il layer di comunicazione tra frontend e backend, implementando il pattern *Repository*. Ogni entità ha un servizio dedicato che gestisce le operazioni CRUD tramite chiamate HTTP asincrone:

```
@Injectable({
  providedIn: 'root'
})
export class CaregiverService {
  private apiUrl = `${environment.apiUrl}/caregivers`;

  constructor(private http: HttpClient) {}

  createCaregiver(caregiverDTO: CaregiverDTO): Observable<CaregiverResponse> {
    return this.http.post<CaregiverResponse>(this.apiUrl, caregiverDTO).pipe(
      tap(response => console.log('Caregiver creato:', response)),
      catchError(this.handleError)
    );
  }

  findByFiscalCode(fiscalCode: string): Observable<ApiResponse> {
    return this.http.get<ApiResponse>(`${this.apiUrl}/search-by-fiscal-code`, {
      params: { fiscalCode }
    });
  }
}
```

Listing 5.3. Esempio di servizio Angular per i caregiver

Interfacce TypeScript (DTO) L'applicazione utilizza interfacce TypeScript per la definizione dei *Data Transfer Object*, garantendo coerenza tra frontend e backend:

```
export interface CaregiverResponse {
    id: number;
    firstName: string;
    lastName: string;
    email: string;
    phone: string;
    address: string;
    dateOfBirth: string;
    fiscalCode: string;
    gender: string;
}

export interface ApiResponse<T = any> {
    success: boolean;
    message: string;
    data?: T;
    errors?: string[];
}
```

Listing 5.4. Interfacce TypeScript per la tipizzazione

Integrazione con PedigreeJS L'integrazione con la libreria `PedigreeJS`, basata su jQuery e D3.js, ha richiesto una gestione personalizzata del ciclo di vita dei componenti Angular. L'inizializzazione della libreria avviene con controllo asincrono del DOM e callback customizzate per la sincronizzazione dei dati:

```
export class PedigreeViewerComponent implements OnInit, AfterViewInit {

  initializePedigreeJS(dataset: any): void {
    const checkDOMElements = (retryCount = 0): boolean => {
      const pedigreeContainer = document.getElementById('pedigree-container');
```

```
6
7       if (!pedigreeContainer) {
8         if (retryCount < 10) {
9           setTimeout(() => checkDOMElements(retryCount + 1), 100);
10          return false;
11        }
12        return false;
13      }
14
15      this.pedigreeOptions = {
16        targetDiv: 'pedigree-container',
17        dataset: dataset,
18        width: 800,
19        height: 600,
20        node_click: this.onSelezioneNodo
21      };
22
23      pedigreejs.build(this.pedigreeOptions);
24      return true;
25    };
26
27    checkDOMElements();
28  }
29
30  onSelezioneNodo = (nodo: any): void => {
31    this.personaSelezionata = { ...nodo };
32    this.mostraDettagliPersona = true;
33  };
34}
```

Listing 5.5. Integrazione PedigreeJS in Angular

Gestione dello stato e reattività L'applicazione implementa un sistema di gestione dello stato basato su componenti stateful e `Observables`. È stata implementata la protezione della navigazione in presenza di modifiche non salvate, particolarmente importante nell'editor di pedigree dove le modifiche potrebbero andare perse. L'interfaccia rimane reattiva grazie a property binding, event binding e two-way binding, garantendo robustezza e modularità.

5.4.2 Comportamento dell'interfaccia utente (UI)

L'interfaccia utente dell'applicazione è stata progettata per offrire un'esperienza intuitiva e professionale, adatta al contesto clinico e sanitario. Le funzionalità principali si concentrano sulla gestione dei pazienti e sull'editor di pedigree genetici, accessibili dalla sidebar di navigazione e ottimizzate per il workflow clinico.

Dashboard La dashboard fornisce il punto di ingresso principale dell'applicazione con una panoramica del sistema e navigazione rapida verso le sezioni funzionali principali: *Pazienti* e *Genetica*.

Lista pazienti La sezione di gestione pazienti implementa funzionalità avanzate per la navigazione e il filtraggio di dataset clinici attraverso diversi strumenti integrati.

Ricerca globale e filtraggio Il sistema di ricerca permette la localizzazione rapida dei pazienti attraverso un campo di input che supporta ricerche in tempo reale, case-insensitive e parziali su tutti i campi configurabili (nome, cognome, codice fiscale). L'interfaccia fornisce feedback immediato durante la digitazione, aggiornando dinamicamente la lista dei risultati.

Sistema di paginazione avanzato Per gestire efficacemente grandi volumi di dati, è stato implementato un sistema di paginazione che consente la selezione della dimensione della pagina (10, 25, 50, 100 elementi), la navigazione tra le pagine e l'ordinamento dinamico delle colonne. Gli utenti possono cliccare sulle intestazioni delle colonne per ordinare i dati in modo crescente o decrescente.

Form pazienti e caregiver I moduli di inserimento e modifica implementano un sistema di validazione robusto e user-friendly.

Validazione in tempo reale Tutti i campi critici sono sottoposti a validazione immediata durante l'inserimento. Il codice fiscale, in particolare, viene verificato sia per il formato che per l'unicità nel sistema. Gli errori vengono visualizzati tramite messaggi descrittivi sotto i campi interessati, mentre il pulsante di salvataggio rimane disabilitato fino alla risoluzione di tutti i problemi di validazione.

Feedback visivo L'interfaccia utilizza codici colore standard (rosso per errori, verde per successo) e icone intuitive per comunicare lo stato dei campi. I tooltip forniscono informazioni aggiuntive sui requisiti di compilazione.

Editor pedigree L'editor rappresenta il cuore funzionale dell'applicazione, offrendo strumenti avanzati per la creazione e modifica dei genogrammi.

Interfaccia principale L'editor presenta una toolbar superiore con controlli per il salvataggio, il caricamento e la gestione del pedigree. Gli indicatori di stato informano l'utente sulle operazioni in corso (salvataggio, caricamento) e sulla presenza di modifiche non salvate attraverso alert visivi prominenti.

Funzionalità di esportazione Una toolbar dedicata offre opzioni di esportazione multiple: formato PNG/SVG per la documentazione clinica, formato JSON per il backup, e formati specializzati BOADICEA v4 e CanRisk v4.0 per l'integrazione con strumenti di valutazione del rischio genetico.

Protezione delle modifiche Il sistema implementa una protezione automatica contro la perdita accidentale di dati attraverso guard di navigazione che intercettano i tentativi di abbandono della pagina in presenza di modifiche non salvate, richiedendo conferma esplicita dall'utente.

Pannello laterale per editing dettagliato Un pannello laterale specializzato consente la modifica granulare dei dettagli di ogni membro del pedigree, includendo informazioni anagrafiche (nome, sesso, età), cliniche (diagnosi oncologiche con età di insorgenza), riproduttive (aborti, nati morti) e familiari (stato di adozione, gemelli monozigoti/dizigoti).

Responsività e accessibilità L'applicazione è progettata per adattarsi in modo automatico a qualsiasi dispositivo, garantendo un'esperienza d'uso ottimale su schermi di diverse dimensioni. Su desktop, l'interfaccia sfrutta un layout a più colonne che consente una visione ampia e ordinata delle informazioni, mentre su tablet e smartphone viene adottata una disposizione verticale semplificata, pensata per facilitare la navigazione da dispositivi mobili.

Va però sottolineato che, per utilizzare al meglio l'editor dei pedigree — soprattutto nel caso di strutture familiari particolarmente complesse — è preferibile disporre di uno schermo abbastanza ampio, così da non compromettere la leggibilità e l'interazione con i dati genealogici.

5.4.3 Comportamento del sistema e API REST

Il backend dell'applicazione rappresenta il cuore logico del sistema, responsabile della gestione dei dati, dell'implementazione delle regole di business e dell'esposizione delle API REST per la comunicazione con il frontend. L'architettura adottata segue i principi consolidati del framework Spring Boot, garantendo scalabilità, manutenibilità e conformità agli standard enterprise.

Architettura del backend

Il backend dell'applicazione è stato realizzato seguendo l'architettura Model-View-Controller (MVC), sfruttando le funzionalità messe a disposizione da Spring Boot. Questo approccio, basato sulla separazione delle responsabilità tra i vari livelli dell'applicazione, rende il codice più leggibile e organizzato, facilitando sia le attività di test che gli eventuali interventi di manutenzione o ampliamento futuri.

Struttura Spring Boot Il framework Spring Boot fornisce la base per l'implementazione dell'architettura REST, con configurazione automatica e convenzioni che accelerano lo sviluppo. La struttura del progetto segue l'organizzazione standard:

```
it.uniba.crud/
|-- controller/      # Endpoint REST
|-- service/         # Logica di business
|-- repository/      # Accesso ai dati
|-- entity/          # Mappatura database
|-- dto/             # Data Transfer Objects
|-- config/          # Configurazioni
\-- exception/       # Gestione eccezioni
```

Controller Layer I controller rappresentano il punto di ingresso delle richieste HTTP e gestiscono la comunicazione REST. Un esempio concreto è il `PatientController`:

```
@RestController
@RequestMapping("/api/v1/patients")
@CrossOrigin(maxAge = 3360)
public class PatientController {
  @Autowired
  private PatientService patientService;

  @GetMapping("")
  public ResponseEntity<List<Patient>> fetchAllPatient() {
    return ResponseEntity.ok(patientService.fetchAllPatient());
  }

  @PostMapping("")
  public ResponseEntity<Patient> createPatient(@RequestBody Patient patient) {
    patient.setId(null);
    return ResponseEntity.ok(patientService.createPatient(patient));
  }

  @PutMapping("/{id}")
  public ResponseEntity<Patient> updatePatient(@PathVariable("id") Long id,
    @RequestBody Patient patient) {
    Patient patObj = patientService.fetchById(id);
    if(patObj!=null) {
      patObj.setFirstName(patient.getFirstName());
      patObj.setLastName(patient.getLastName());
      // ... altri campi
    }
    return ResponseEntity.ok(patientService.updatePatient(patObj));
  }
}
```

Service Layer I servizi implementano la logica di business. L'interfaccia `PatientService` definisce il contratto:

```
public interface PatientService {
  List<Patient> fetchAllPatient();
```

```
    Patient fetchById(Long id);
    Optional<Patient> fetchByIdSafe(Long id);
    Patient createPatient(Patient patient);
    Patient updatePatient(Patient patient);
    String deletePatient(Patient patient);
    boolean existsById(Long patientId);
}
```

Repository Layer I repository gestiscono l'accesso ai dati utilizzando Spring Data JPA.

Entity + DTO Le entità rappresentano la mappatura diretta delle tabelle del database. In questo esempio la classe `Patient` utilizza le annotazioni JPA e Lombok:

```
@Data
@AllArgsConstructor
@NoArgsConstructor
@Builder
@Entity
@Table(name = "patient", schema = "public")
public class Patient {
    @Id
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    private Long id;

    @Column(name = "first_name", length = 250, nullable = false)
    private String firstName;

    @Column(name = "last_name", length = 250, nullable = false)
    private String lastName;

    @JsonFormat(pattern = "yyyy-MM-dd")
    @Column(name = "date_of_birth", nullable = false)
    private LocalDate dateOfBirth;

    @Column(name = "fiscal_code", length = 16, unique = true, nullable = false)
    private String fiscalCode;
```

```
24
25    @Column(name = "gender", length = 1, nullable = false)
26    private String gender;
27
28    @JsonIgnore
29    @OneToMany(mappedBy = "patient", cascade = CascadeType.ALL, orphanRemoval =
        true)
30    private Set<PatientCaregiverAssociation> caregiverAssociations;
31  }
```

Il sistema utilizza una classe `ApiResponse<T>` generica per standardizzare tutte le risposte:

```
1  public class ApiResponse<T> {
2    private boolean isSuccessful;
3    private String responseMessage;
4    private T responseData;
5    private Map<String, String> validationErrors;
6
7    public ApiResponse(boolean isSuccessful, String responseMessage, T
       responseData) {
8      this(isSuccessful, responseMessage, responseData, null);
9    }
10
11   public ApiResponse(boolean isSuccessful, String responseMessage) {
12     this(isSuccessful, responseMessage, null, null);
13   }
14
15   public ApiResponse(boolean isSuccessful, String responseMessage, Map<String,
       String> validationErrors) {
16     this(isSuccessful, responseMessage, null, validationErrors);
17   }
18 }
```

Lista di alcuni endpoint principali con descrizione

- `GET /api/v1/patients` – Recupera la lista di tutti i pazienti
- `POST /api/v1/patients` – Crea un nuovo paziente
- `GET /api/v1/patients/{id}` – Recupera un paziente specifico
- `PUT /api/v1/patients/{id}` – Aggiorna un paziente
- `DELETE /api/v1/patients/{id}` – Elimina un paziente
- `GET /api/v1/caregivers/search-by-fiscal-code` – Cerca un caregiver per codice fiscale
- `GET /api/v1/pedigrees/by-patient/{patientId}` – Recupera il pedigree di un paziente
- `GET /api/v1/pedigrees/exists/{patientId}` – Verifica esistenza pedigree
- `POST /api/v1/pedigrees` – Crea o aggiorna un pedigree

Esempi di chiamate API

Ricerca caregiver per codice fiscale Richiesta:

```
GET /api/v1/caregivers/search-by-fiscal-code?fiscalCode=RSSMRA85M01H501Z
```

Risposta:


```
{
  "success": true,
  "message": "Trovato Caregiver associato a quel Codice Fiscale.",
  "data": {
    "id": 8,
    "firstName": "Anna",
    "lastName": "Verdi",
    "fiscalCode": "VRDNNA80M01H501Z",
    "email": "anna.verdi@email.com",
    "phone": "+39 987 654 3210"
  },
```

```
  "validationErrors": null
}
```

Struttura della risposta JSON Tutte le API seguono una struttura di risposta standardizzata attraverso la classe `ApiResponse<T>`, che garantisce:

- `success`: booleano che indica l'esito dell'operazione
- `message`: messaggio descrittivo
- `data`: i dati restituiti (in caso di successo)
- `validationErrors`: eventuali errori (in caso di fallimento)

Questa struttura assicura coerenza tra le risposte e una gestione uniforme degli errori da parte del frontend.

Esportazione dati e interoperabilità

L'applicazione include funzionalità di esportazione dei pedigree in diversi formati, garantendo l'interoperabilità con altri sistemi sanitari e strumenti di analisi genetica, elemento importante nell'ecosistema della medicina personalizzata.

Tipologie di export implementate **Export grafico (PNG/SVG)** Finalizzato alla documentazione clinica e inserimento in referti medici. Il formato PNG offre immagini raster ad alta risoluzione per stampa e archiviazione, mentre SVG fornisce grafici vettoriali scalabili per web e documenti digitali.

Export JSON (backup e interscambio) Permette il backup completo dei dati e la migrazione tra sistemi informativi. Il formato include metadati del paziente, struttura del pedigree, data di esportazione e versione del formato per garantire la tracciabilità.

Export formati clinici (BOADICEA / CanRisk) Consente l'integrazione con software di analisi del rischio genetico già diffusi in ambito clinico:

- **BOADICEA v4**: formato standard utilizzato per l'analisi del rischio di cancro al seno e ovaio, compatibile con i principali strumenti di genetica clinica

- **CanRisk v4.0**: formato più recente che supporta analisi multi-tumore e applicazioni di medicina preventiva

Valore clinico degli export Gli export grafici consentono l'integrazione immediata in cartelle cliniche e referti, mentre i formati JSON garantiscono backup sicuri e migrazione tra sistemi informativi ospedalieri. I formati BOADICEA e CanRisk permettono l'utilizzo dei dati con software specializzati per il calcolo del rischio genetico, facilitando l'integrazione nel workflow clinico esistente.

La funzionalità di export rappresenta un elemento di connessione tra il sistema sviluppato e l'ecosistema clinico, consentendo l'utilizzo efficace dei dati genetici in contesti di ricerca, diagnosi e terapia personalizzata.

5.5 Criticità e soluzioni adottate

Nel corso dello sviluppo dell'applicazione sono emerse alcune criticità tecniche, tra cui la più rilevante ha riguardato l'adattamento della libreria PedigreeJS alle esigenze pratiche del contesto clinico.

In questa sezione vengono approfondite le principali problematiche incontrate lungo il percorso, con particolare attenzione alle soluzioni adottate per superarle. Molte di queste soluzioni sono state frutto di un lavoro di analisi e sperimentazione, necessario per garantire un funzionamento affidabile e adatto all'utilizzo in ambito sanitario.

5.5.1 Identificazione delle limitazioni della libreria originale

La libreria PedigreeJS, sviluppata dal Centre for Cancer Genetic Epidemiology dell'Università di Cambridge, implementa le regole standard della genetica formale seguendo una logica rigorosamente scientifica. Tuttavia, durante l'analisi dei requisiti con il personale medico che avrebbe utilizzato il sistema, sono emerse due limitazioni critiche che rendevano la libreria inadatta alle esigenze cliniche specifiche. Su richiesta esplicita dei medici che dovevano utilizzare l'applicazione, è emersa la necessità di superare i vincoli imposti dalla libreria originale per adattarla alle situazioni cliniche reali che si presentano quotidianamente nella pratica ospedaliera.

La prima limitazione riguarda l'impossibilità di rappresentare genitori single con figli. I medici hanno evidenziato come nel contesto clinico sia frequente incontrare pazienti che hanno figli ma per i quali il partner non è noto, non è disponibile per l'anamnesi, o semplicemente non è rilevante dal punto di vista genetico. La libreria originale, tuttavia, richiede obbligatoriamente la presenza di entrambi i genitori per ogni figlio, impedendo la rappresentazione di queste situazioni comuni.

Un'altra limitazione emersa riguarda l'impossibilità, nella versione originale della libreria, di rappresentare coppie senza figli. Dal confronto con alcuni medici è emersa l'esigenza, frequente in ambito genetico, di documentare relazioni di coppia anche in assenza di una progenie. Questo si verifica, ad esempio, nei casi in cui entrambi i partner siano portatori di mutazioni genetiche rilevanti, informazioni fondamentali per la consulenza genetica.

Tuttavia, la versione standard di PedigreeJS richiedeva almeno un figlio in comune per poter visualizzare una coppia, limitando così la possibilità di rappresentare correttamente molte situazioni cliniche reali.

Sulla base di queste esigenze specifiche espresse dai medici, si è reso necessario sviluppare una soluzione che mantenesse l'integrità tecnica della libreria originale adattandola alle necessità operative del contesto clinico.

5.5.2 Sviluppo della soluzione basata sui nodi nascosti

Per risolvere queste limitazioni mantenendo l'integrità strutturale della libreria originale, è stato sviluppato un sistema innovativo basato sul concetto di nodi nascosti. Questa soluzione introduce flessibilità nella rappresentazione visiva preservando la coerenza logica della struttura dati sottostante. Un nodo nascosto è un elemento del pedigree che esiste fisicamente nella struttura dati del sistema ma non viene visualizzato nell'interfaccia grafica. Questo approccio permette di mantenere le relazioni logiche richieste dalla libreria originale mentre si offre all'utente una rappresentazione visiva semplificata e clinicamente appropriata. Il sistema dei nodi nascosti introduce inoltre la possibilità di riattivazione, permettendo di rendere visibili elementi precedentemente nascosti quando diventano clinicamente rilevanti, garantendo così flessibilità operativa senza perdita di informazioni.

5.5.3 Implementazione della gestione dei partner nascosti

Una delle prime modifiche significative ha riguardato la funzione `addpartner`, responsabile dell'aggiunta di nuovi partner all'interno del pedigree. Nella versione originale della libreria, non era possibile creare una coppia senza almeno un figlio in comune, limitando fortemente la rappresentazione di scenari clinici reali. Per risolvere questo problema, è stato introdotto un meccanismo di controllo più intelligente, che prima di tutto verifica se esistono già partner "nascosti" associati al nodo selezionato. In questo modo è possibile riattivare relazioni precedentemente esistenti anziché crearne di nuove, evitando duplicazioni e garantendo la coerenza del pedigree.

```
//Controllare se esistono gia partner (nascosti)
let existingPartners = utils.get_partners(dataset, tree_node.data);

if (existingPartners.length > 0) {
    // Partner nascosto esiste, renderlo visibile
    let partnerName = existingPartners[0];
    let partner = utils.getNodeByName(dataset, partnerName);
    if (partner && partner.hidden) {
        delete partner.hidden;
        return; // Non creare nuovo partner
    }
}
```

Quando non esistono partner nascosti da riattivare, il sistema procede con la creazione di una nuova coppia implementando una strategia innovativa: viene creato il partner richiesto insieme a un figlio nascosto che funge da collegamento logico tra i due. Questo figlio nascosto è invisibile all'utente ma mantiene la coerenza strutturale richiesta dalla libreria originale.

```
let partner = addsibling(dataset, tree_node.data, tree_node.data.sex === 'F' ?
    'M' : 'F', tree_node.data.sex === 'F');
partner.noparents = true;

let child = {"name": utils.makeid(4), "sex": "U", "hidden": true};
```

```
child.mother = (tree_node.data.sex === 'F' ? tree_node.data.name :
    partner.name);
child.father = (tree_node.data.sex === 'F' ? partner.name :
    tree_node.data.name);
```

Questo approccio permette di visualizzare coppie senza figli apparenti mantenendo la struttura logica necessaria per il corretto funzionamento della libreria.

5.5.4 Implementazione della gestione dei figli per coppie precedentemente single

La seconda modifica significativa riguarda la funzione `addchild`, che gestisce l'aggiunta di figli nel pedigree. La modifica implementata introduce un meccanismo di riattivazione intelligente che trasforma l'esperienza utente da rigida a fluida e naturale. Quando l'utente richiede l'aggiunta di un figlio a una coppia che precedentemente era composta da due individui single, il sistema verifica innanzitutto se esistono figli nascosti da riattivare. Questa verifica è cruciale poiché quando si forma una coppia da due single, viene automaticamente creato un figlio nascosto per mantenere il collegamento logico.

```
let hiddenChildren = findHiddenChildren(dataset, node);

if (hiddenChildren.length > 0 && nchild === 1) {
    let hiddenChild = hiddenChildren[0];
    delete hiddenChild.hidden;
    hiddenChild.sex = sex;  // Allow user to choose the sex
    return [hiddenChild];
}
```

Il meccanismo di riattivazione permette di trasformare il figlio nascosto in un figlio reale, consentendo all'utente di specificarne il sesso e le caratteristiche specifiche di questo nodo/individuo. Questo approccio evita la creazione di nodi duplicati e mantiene la coerenza temporale delle operazioni effettuate sul pedigree. Nel caso in cui non esistano figli nascosti da riattivare, il sistema procede con la logica standard di aggiunta di figli.

5.5.5 Impatto clinico e valore innovativo

Le modifiche implementate hanno trasformato radicalmente l'usabilità clinica del sistema, risolvendo completamente le limitazioni identificate dal personale medico. Il sistema ora permette la rappresentazione fluida e naturale di qualsiasi configurazione familiare reale, mantenendo la flessibilità operativa necessaria per l'uso quotidiano in ambito sanitario.

L'idea di utilizzare nodi nascosti si è rivelata una soluzione efficace e piuttosto originale. Ha permesso di mantenere piena compatibilità con standard clinici come BOADICEA e CanRisk, senza dover stravolgere la struttura dati esistente.

Le modifiche introdotte non hanno intaccato le funzionalità già presenti nella libreria, ma hanno aggiunto quella flessibilità che mancava per adattarsi meglio alle esigenze reali di chi lavora in ambito medico. È un esempio concreto di come strumenti pensati in contesti accademici possano essere adattati con successo alla pratica clinica, senza perdere coerenza o precisione.

Il sistema risultante offre un'esperienza utente naturale e intuitiva che si adatta al modo di pensare dei clinici, eliminando i vincoli artificiali che limitavano l'adozione della libreria originale in contesti clinici reali.

Capitolo 6

Conclusioni e sviluppi futuri

Il lavoro svolto ha portato alla realizzazione di un'applicazione web completa per la gestione di pazienti e la visualizzazione interattiva dei relativi genogrammi clinici. L'obiettivo era quello di creare uno strumento semplice da usare, ma al tempo stesso conforme agli standard medici e realmente utile in un contesto sanitario. In questo senso, il sistema sviluppato risponde bene alle necessità pratiche riscontrate durante il confronto con professionisti del settore.

Dal punto di vista tecnologico, l'applicazione si basa su un'architettura full-stack moderna, con l'uso di Angular per il frontend, Spring Boot per il backend e PostgreSQL per la gestione dei dati. Questa combinazione ha permesso di mantenere il progetto chiaro, modulare e facilmente estendibile. La libreria PedigreeJS è stata integrata e modificata per adattarsi meglio al contesto clinico, introducendo funzionalità che in origine non erano supportate, come la gestione di genitori single o di coppie senza figli.

Dal punto di vista tecnico, il progetto ha mostrato come sia possibile costruire un sistema solido partendo da tecnologie open-source, adattandole con interventi mirati alle esigenze di un contesto clinico reale. Senza dover reinventare tutto da zero, è stato possibile combinare strumenti esistenti in modo efficace, integrando frontend, backend e motore grafico in un'unica piattaforma coerente. Questo approccio ha permesso di concentrarsi sugli aspetti che davvero servivano, ottenendo un risultato funzionale, stabile e pronto per essere ampliato in futuro.

L'applicazione offre attualmente funzionalità complete per la creazione, modifica

e visualizzazione dei pedigree, oltre alla possibilità di esportarli in formati compatibili con BOADICEA e CanRisk. Questi elementi rendono il sistema adatto non solo all'uso interno, ma anche all'integrazione con strumenti predittivi già utilizzati nella pratica clinica. Gli sviluppi futuri di questo progetto possono seguire diverse direzioni. Una prima evoluzione naturale riguarda l'estensione delle funzionalità cliniche, come l'integrazione con dati fenotipici o referti genetici, al fine di arricchire ulteriormente le informazioni visualizzabili nel pedigree. In secondo luogo, potrebbe essere introdotto un modulo di autenticazione e gestione dei ruoli utente, per consentire un accesso differenziato tra medici, ricercatori e personale amministrativo. Un'altra prospettiva interessante consiste nell'integrazione di modelli di intelligenza artificiale, in grado di analizzare automaticamente i pedigree per suggerire pattern familiari ricorrenti o segnalare anomalie cliniche. Inoltre, l'evoluzione verso un'architettura cloud-based — eventualmente containerizzata e distribuita — permetterebbe di migliorare l'accessibilità, la scalabilità e l'adozione del sistema in ambienti multi-centro.

In conclusione, il lavoro svolto rappresenta un contributo concreto e applicabile alla digitalizzazione dei dati familiari in ambito medico. L'approccio adottato, incentrato su esigenze reali e supportato da tecnologie affidabili, ha dimostrato come sia possibile progettare strumenti clinicamente rilevanti partendo da un'analisi accurata del contesto e da un solido impianto tecnico. Le basi gettate da questo progetto aprono a numerose possibilità di espansione, nella direzione di una sanità sempre più personalizzata, predittiva e supportata dalla tecnologia.

Bibliografia

[1] Ambula Healthcare. Structured data vs unstructured data in ehrs, 2022. [online].

[2] Brian Eastwood. How to navigate structured and unstructured data as a healthcare organization. *HealthTech Magazine*, 2023.

[3] Amen Faridoon and M. Tahar Kechadi. Healthcare data governance, privacy, and security – a conceptual framework. *arXiv*, 2024.

[4] HL7 International. Consolidated clinical document architecture, 2023. [online].

[5] HL7 International. Fast healthcare interoperability resources, 2023.

[6] Sheng Lee et al. Digital health data quality issues: Systematic review. *Journal of Medical Internet Research*, 25:e42615, 2023.

[7] Abigail E. Lewis, Nicole Weiskopf, Zachary B. Abrams, Randi Foraker, Albert M. Lai, Philip R.O. Payne, and Aditi Gupta. Electronic health record data quality assessment and tools: a systematic review. *Journal of the American Medical Informatics Association*, 30(10):1730–1740, 2023.

[8] National Library of Medicine (US), NCBI Bookshelf. *Obtaining Data From Electronic Health Records*. National Center for Biotechnology Information, 2023.

[9] PubMed Central. Patient perspective on predictive models in healthcare. *PubMed Central*, 2025.

[10] Michael J. Soucie. Public health surveillance and data collection: General principles. *PubMed Central*, 17:–, 2015.

[11] StatPearls. *Healthcare Analytics*. NCBI Bookshelf, 2025.

[12] Reed T. Sutton, David Pincock, Daniel C. Baumgart, Daniel C. Sadowski, Richard N. Fedorak, and Karen I. Kroeker. An overview of clinical decision support systems: benefits, risks, and strategies for success. *npj Digital Medicine*, 3:17, 2020.