Elaborato Matassini Cosimo

5°Bif

Sezione Informatica

ITT. Galileo Ferraris ISIS Valdarno

Prima Parte : Informatica

Il seguente modello cerca di adattarsi a quelle che sono le queries, in modo da risultare innanzitutto funzionale all'esecuzione delle interrogazioni stesse, ma anche di facile comprensione e implementazione. Ogni medico gestisce più patologie (e deve gestirne almeno una, per essere definito medico); allo stesso tempo ogni patologia viene gestista da un solo medico (non necessariamente, perché magari una patologia non è stata ancora assegnata ad un medico), in modo da semplificare tutto il processo. Ogni paziente deve presentare almeno una patologia per esserlo, e può averne più di una, inoltre ogni patologia può essere presente in più pazienti, ma non necessariamente (possono esserci delle patologie che non sono presenti in nessun paziente). Di conseguenza l'associazione che lega queste due entità è di tipo N a N. I farmaci, nelle queries richieste, non hanno bisogno di effettuare nessuna congiuzione con altre entità. Per convenzione è stato scelto comunque che questi si colleghino in maniera N a 1 con le patologie, in modo da risultare più aderenti alla situazione reale. Nel modello concettuale è stato stabilito che un farmaco per esistere nel database debba essere per forza associato ad una patologia, ma una patologia può non avere associato un farmaco perché ad esempio probabilmente non è stato ancora scoperto o sperimentato. Oltre alle informazioni ordinarie delle tabelle (ad esempio nome, cognome e data di nascita delle persone) sono state inserite ulteriori informazioni al fine della risoluzione ottimale delle queries richieste:

- ➔ Numero delle richieste dei farmaci (indicato come n_farmaci nel database)
- ➔ Terza tabella che collega Pazienti a Patologie, la quale contiene l'informazione "esenzione". Questo perché innanzitutto, come regola standard, il "mondo" dei database relazionali prevede un'operazione di questo tipo; secondo: l'attributo

esenzione, se fosse stata un'informazione su un paziente, avrebbe indicato che che ogni paziente ha l'esenzione per tutte le patologie che presenta (cosa poco probabile nella realtà), e se fosse stata un'informazione sulle patologie, avrebbe indicato che ogni patologia di quel tipo godrebbe dell'esenzione (neanche questa situazione avrebbe rispecchiato la realtà).

Le tabelle sono state progettate in modo da rispettare tutte le regole di normalizzazione ed evitare ridondanze:

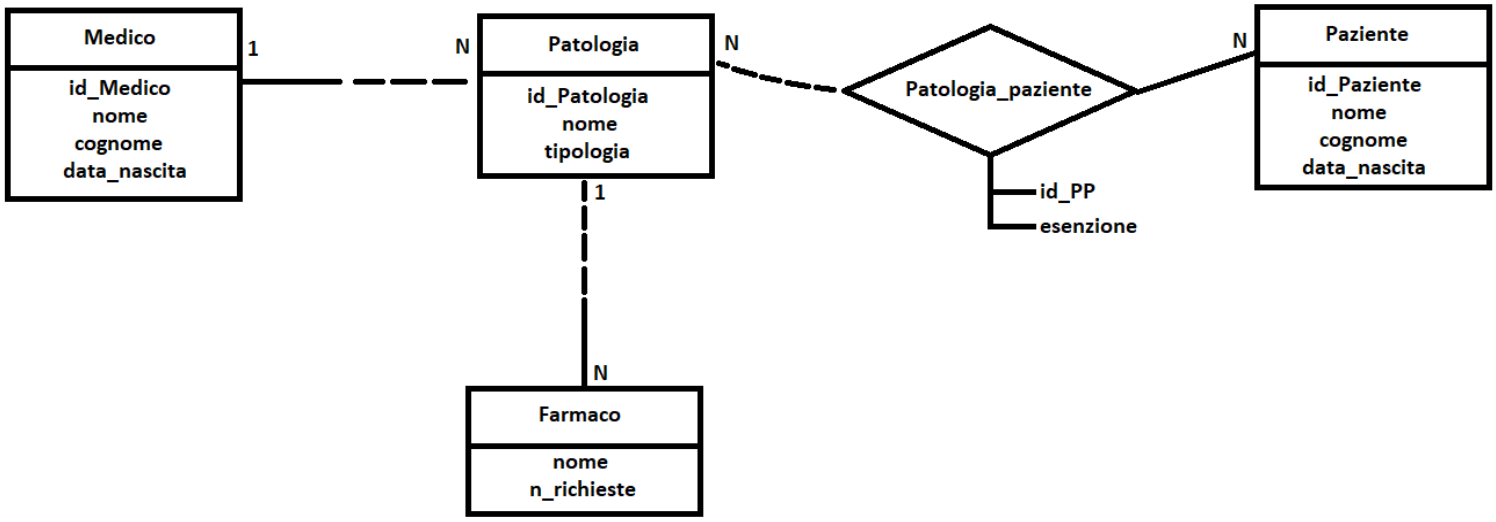
→ tutte le righe sono diverse tra loro e contengono lo stesso numero di colonne (Prima Forma Normale 1FN)

→ ogni attributo rappresenta informazioni non ulteriormente scomponibili

→ tutti gli attributi non-chiave di tutte le entità dipendono da una sola chiave (eliminazione della dipendenza parziale) (Seconda Forma Normale 2FN)

→ tutti gli attributi non-chiave di tutte le entità dipendono per forza da una chiave e non da un attributo non-chiave (eliminazione della dipendenza transitiva) (Terza Forma Normale 3FN)

MODELLO E/R (Entity/Relationship)



MODELLO LOGICO

Naturalmente, nel passaggio da modello concettuale a logico, viene trasformato il nome delle entità al plurale e vengono inserite le chiavi esterne. Gli attributi sottolineati indicano ogni chiave primaria delle relazioni, ovvero il dato che identifica ogni record.

Medici(id_Medico, nome, cognome, data_nascita);

Patologie(id_Patologia, nome, tipologia, id_Medico(FK));

Farmaci(id_Farmaco, nome, n_richieste, id_Patologia(FK));

Pazienti(id_Paziente, nome, cognome, data_nascita);

Patologie/Pazienti(id_PP, esenzione, id_Patologia(FK), id_Paziente(FK));

INTERROGAZIONI

1) SELECT Medici.cognome AS 'Cognome Medico', Medici.nome AS 'Nome Medico', Patologie.nome AS 'Nome patologia', tipologia

FROM Medici INNER JOIN Patologie USING (id_Medico)

ORDER BY Medici.cognome, Medici.nome, Patologie.nome;

2) SELECT nome AS 'Farmaco', n_richieste AS 'Numero richieste'

FROM farmaci

WHERE n_richieste = (SELECT MAX(n_richieste) FROM farmaci);

3) SELECT COUNT(id_paziente) AS 'Numero pazienti'

FROM Patologie INNER JOIN patologie_pazienti USING (id_patologia)

WHERE esenzione = True AND nome != 'Asma';

GUIDA ALL'IMPLEMENTAZIONE DEL DATABASE IN MySQL

```
CREATE DATABASE SistemaSanitario;
```

COMANDI DATA DEFINITION LANGUAGE

```
CREATE TABLE Medici(
id_Medico INT PRIMARY KEY auto_increment,
nome CHAR(20) NOT NULL,
cognome CHAR(20) NOT NULL,
data_nascita DATE
);
```

NOTA: L'ATTRIBUTO DATE NON È STATO IMPOSTATO "NOT NULL" PERCHÉ NON È FONDAMENTALE AI FINI DEL DATABASE E DELLE INTERROGAZIONI. NOME E COGNOME INVECE, DEVONO ESSERE NECESSARIAMENTE PRESENTI, POICHÉ NELLA MAGGIOR PARTE DEI CASI RAPPRESENTANO IL MODO PER IDENTIFICARE OGNI RECORD ALL'INTERNO DELLE TABELLE NEL CASO SI DOVESSERO ELENCARE IN UNA INTERROGAZIONE (LA LORO CHIAVE PRIMARIA LI INDENTIFICA UNIVOCAMENTE, MA NON È COMPRENSIBILE).

```
CREATE TABLE Pazienti(
id_Paziente INT PRIMARY KEY auto_increment,
nome CHAR(20) NOT NULL,
cognome CHAR(20) NOT NULL,
data_nascita DATE
);
```

```
CREATE TABLE Patologie(
id_Patologia INT PRIMARY KEY auto_increment,
nome CHAR(20) NOT NULL,
tipologia CHAR(20),
id_Medico INT,
CONSTRAINT fk_medicopatologia FOREIGN KEY (id_Medico) REFERENCES Medici(id_Medico)
ON UPDATE CASCADE
ON DELETE SET NULL
);

CREATE TABLE Farmaci(
id_Farmaco INT PRIMARY KEY auto_increment,
nome CHAR(20) NOT NULL,
n_richieste INT,
id_Patologia INT,
CONSTRAINT fk_patologiafarmaco FOREIGN KEY (id_Patologia) REFERENCES Patologie(id_Patologia)
ON UPDATE CASCADE
ON DELETE SET NULL
);
```

```
CREATE TABLE Patologie_Pazienti(
id_PP INT PRIMARY KEY auto_increment,
id_Patologia INT,
id_Paziente INT,
esenzione BOOLEAN DEFAULT False,
CONSTRAINT fk_patologiapaziente FOREIGN KEY (id_Patologia) REFERENCES Patologie(id_Patologia)
ON UPDATE CASCADE
ON DELETE SET NULL,
CONSTRAINT fk_pazientepatologia FOREIGN KEY (id_Paziente) REFERENCES Pazienti(id_Paziente)
ON UPDATE CASCADE
ON DELETE SET NULL
);
```

COMANDI DATA MANIPULATION LANGUAGE

3 medici, 5 tipi di patologie, 5 pazienti, 6 farmaci, 8 patologie presenti

```
INSERT INTO Medici(nome, cognome, data_nascita) VALUES ("Massimo", "Cuccia", "1975-04-06");
INSERT INTO Medici(nome, cognome, data_nascita) VALUES ("Guido", "Bergamini", "1968-11-20");
INSERT INTO Medici(nome, cognome, data_nascita) VALUES ("Anna", "Barozzi", "1979-05-07");
```

```
INSERT INTO Patologie(nome, tipologia, id_Medico) VALUES ("Asma", "Pneumologia", 3);

INSERT INTO Patologie(nome, tipologia, id_Medico) VALUES ("Dermatite", "Dermatologia", 1);

INSERT INTO Patologie(nome, tipologia, id_Medico) VALUES ("Scoliosi", "Ortopedia", 2);

INSERT INTO Patologie(nome, tipologia, id_Medico) VALUES ("Miopia", "Oculistica", 2);

INSERT INTO Patologie(nome, tipologia, id_Medico) VALUES ("Diabete", "Diabetologia", 3);
```

```
INSERT INTO Pazienti(nome, cognome, data_nascita) VALUES ("Cosimo", "Nuti", "2002-04-03");

INSERT INTO Pazienti(nome, cognome, data_nascita) VALUES ("Mirko", "Tognaccini", "1999-10-07");

INSERT INTO Pazienti(nome, cognome, data_nascita) VALUES ("Asia", "Verdi", "1968-02-10");

INSERT INTO Pazienti(nome, cognome, data_nascita) VALUES ("Giulia", "Nannini", "2001-12-12");

INSERT INTO Pazienti(nome, cognome, data_nascita) VALUES ("Ernesto", "Mici", "1965-12-01");
```

```
INSERT INTO Farmaci(nome, n_richieste, id_Patologia) VALUES ("Beclometasone", 155, 1);

INSERT INTO Farmaci(nome, n_richieste, id_Patologia) VALUES ("Clobetasolo", 36, 2);

INSERT INTO Farmaci(nome, n_richieste, id_Patologia) VALUES ("Doxiciclina", 27, 2);

INSERT INTO Farmaci(nome, n_richieste, id_Patologia) VALUES ("Empagliflozin", 99, 5);

INSERT INTO Farmaci(nome, n_richieste, id_Patologia) VALUES ("Atropina", 478, 4);
```

INSERT INTO Farmaci(nome, n_richieste, id_Patologia) VALUES ("Tazarotene", 370, 3);

INSERT INTO Patologie_Pazienti(id_Patologia, id_Paziente, esenzione) VALUES (1, 2, False);

INSERT INTO Patologie_Pazienti(id_Patologia, id_Paziente, esenzione) VALUES (2, 4, True);

INSERT INTO Patologie_Pazienti(id_Patologia, id_Paziente, esenzione) VALUES (3, 5, True);

INSERT INTO Patologie_Pazienti(id_Patologia, id_Paziente, esenzione) VALUES (2, 2, False);

INSERT INTO Patologie_Pazienti(id_Patologia, id_Paziente, esenzione) VALUES (3, 1, False);

INSERT INTO Patologie_Pazienti(id_Patologia, id_Paziente, esenzione) VALUES (3, 3, True);

INSERT INTO Patologie_Pazienti(id_Patologia, id_Paziente, esenzione) VALUES (4, 4, False);

INSERT INTO Patologie_Pazienti(id_Patologia, id_Paziente, esenzione) VALUES (1, 4, True);

CENNI SUL SITO WEB

Per quanto riguarda il sito web, ho utilizzato un file .php per la pagina principale, uno per la pagina dell'esecuzione della query scelta tra le proposte; il file "connessione.php" include il codice per connettersi al database. L'utilizzo di un file unico esterno che viene incluso negli altri porta almeno 2 vantaggi:

- il codice nelle altre pagine risultà più pulito (ci sono meno righe di codice)
- riutilizzabilità per cambiamenti futuri (le modifiche ad un unico file si applicano a tutti quelli che lo includono)

Infine è stato utilizzato un file .css esterno per gestire il design della pagina. Non è stata implementata la possibilità di gestire il layout su dispositivi mobili perché la pagina contiene talmente pochi elementi da non averne bisogno.

Seconda Parte: T.E.P.S.I.T.

Analisi programma

Il programma Client/Server è stato realizzato con il linguaggio di programmazione Java, oggetto di studio di tutto l'anno.

Le classi utilizzate sono 6 (5 per il server ed 1 per il client):

- Server
- ConnessioneClient
- Protocollo
- Visita
- Paziente

- Client

Naturalmente, date le circostanze, in un programma del genere non può mancare la connessione di più utenti nello stesso momento al server. Di conseguenza sono stati utilizzati i thread tramite l'interfaccia Runnable, implementata nella classe "ConnessioneClient". Le classi che gestiscono la memorizzazione dei pazienti e le visite sono Paziente e Visita. Paziente è composto da due stringhe (nome e cognome) e un id (attributo di tipo intero int) assegnato automaticamente nel momento dell'inserimento del paziente stesso: id che dovrà essere poi utilizzato durante la richiesta delle visite e la visualizzazione delle stesse correlate all'utente. Visita è composta anch'essa da un id che indentifica il paziente, una stringa che descrive il tipo/motivo della visita e la data comprensiva dell'orario grazie alla classe LocalDateTime. La conversione da stringa a LocalDateTime viene gestita da DateTimeFormatter con il seguente pattern: dd-MM-yyyy HH:mm dove dd sta per il giorno, MM per mese, yyyy l'anno completo a 4 cifre, HH l'ora e mm i minuti. Per gestire la comunicazione tra il client e il server stesso, è stato implementato un protocollo di comunicazione personalizzato in base alle esigenze (Protocollo).

Di seguito i 3 comandi riconosciuti dal server:

→ INSERT *nome* *cognome*

Gestisce l'inserimento di un paziente e riceve un messaggio di conferma con il codice assegnato.

→ REQUEST *idpaziente* *scopo* *data*

Richiede una visita per il paziente in questione e riceve un messaggio di conferma.

Mi rendo conto che per risultare il più realistico possibile dovrebbe esserci un elenco di visite prenotabili, cosa comunque non richiesta dalla traccia. Per semplicità e questioni di tempo ho deciso di impedire l'inserimento di più richieste nella stessa ora dello stesso giorno per la stessa tipologia (ognuna viene gestita solamente da un medico, come esplicato nella parte di informatica).

→ VISITS *idpaziente*

Mostra tutte le visite correlate al paziente con il codice inserito, se presenti.

Per quanto riguarda la connessione, ho deciso di sviluppare un programma client che funzioni da riga di comando. Su quella base, ho adattato il codice anche a delle pagine JSP per gestire il programma da pagine web, le quali, tramite css, possono essere fruibili in modo ottimale anche da dispositivi mobili come da richiesta. Nel file "stile.css" viene gestito il design delle pagine e riadattato tramite le media queries nel caso in cui ce ne fosse bisogno (per i dispositivi mobili che hanno lunghezza massima di 1000px). Inoltre, le pagine JSP, invece di connettersi al server ogni volta che si passa da una all'altra, lo fanno un'unica volta poiché viene gestito tutto tramite una sessione con il browser.

ANALISI INVIO NOTIFICHE

Inviare una mail di notifica ad ogni paziente qualche giorno prima della visita sarebbe un ottimo modo per avvisarli. Se il sistema sanitario dovesse creare e inviare manualmente una mail a tutti i pazienti registrati, dovrebbe assumere personale che occuperebbe tempo e risorse; a mio parere, la scelta migliore sarebbe quella dell'utilizzo della "e-mail automation", ovvero un indirizzo di posta elettronica che invia e-mail in automatico (in base a determinate condizioni preimpostate dagli amministratori) ai contatti presenti nel database. Ovviamente quest'ultimo ha bisogno di conoscere l'indirizzo di posta elettronica dei pazienti, che possono fornirlo al momento della registrazione. In questo caso, il miglior metodo, siccome gli avvisi per le visite sono solamente dei promemoria, sarebbe quello di utilizzare una indirizzo mail normale, in quanto gli unici costi da sostenere sarebbero quelli del servizio di email automatico, che sia gestito internamente o esternamente da terzi a cui si è fatto affidamento. Nonostante ciò, dato che nell'esercizio è chiaramente richiesto di fornire una comunicazione ufficiale (che quindi abbia valore legale), la scelta ricade sull'utilizzo di un sistema PEC, in quanto tutte le comunicazioni sono protette grazie alla crittografia e firmate digitalmente. Questo, a differenza del metodo standard, comporta dei costi nettamente più alti e a mio avviso non rappresenta una necessità per un servizio del genere, a meno che nei messaggi non siano compresi dati sensibili (non specificati nella traccia) che richiedono necessariamente l'utilizzo di una Posta Elettronica Certificata. In entrambi i casi i costi rispetto a l'utilizzo di avvisi cartacei, che devono essere spediti fisicamente ai destinatari, sono estremamente più bassi. Infatti i tempi risulterebbero estremamente più lunghi: sarebbero sull'ordine delle ore se non giorni senza contare eventuali ritardi, contro i secondi impiegati per l'invio elettronico; inoltre verrebbe occupato meno spazio fisico e materiale (toccasana per l'ambiente, verso il quale negli ultimi tempi viene rivolta molta attenzione).

Terza Parte: G.P.O.I.

ANALISI

La prima attività (Analisi strategia migliore) si concentra nel capire quali potrebbero essere le modalità migliori per agire durante il progetto. Gli amministratori saranno coloro che dovranno programmare il database e successivamente gestire ed effettuare eventuali modifiche. I designer e i programmatori saranno coloro che svilupperanno il sito web. Gli amministratori di sistema dovranno gestire la manutenzione dei dispositivi, mentre quelli di rete gestiranno tutto ciò che riguarda le telecomunicazioni, come dice la parola stessa. Nel programma Planner sono state inserite le attività con le rispettive durate, vincoli e risorse. I costi delle risorse di tipo materiale si intendono per unità, mentre quelli relativi alla manodopera per unità di tempo (orari). Una volta scelti, i dispostivi hardware dell'infrastruttura (server, database, gruppi di continuità ed altri secondari), verranno installati. Da notare che l'installazione dei gruppi di continuità comporta il vincolo di dover essere terminata un giorno dopo l'installazione degli altri due dispositivi, in modo da permettere l'ottimizzazione nella gestione degli spazi e agevolare l'avvio delle attività successive. In seguito viene creato il database e testato, sviluppato il programma server e infine creato, testato e, una volta scelto il fornitore del servizio di email automation per le notifiche agli utenti, pubblicato.

ELENCO ATTIVITÀ (lettera, nome, durata (giorni), precedenze, vincoli)

→ A Analisi strategia migliore, 7, NO PRECEDENZE, NO VINCOLI

→ B Scelta amministratori database, 4, NO PRECEDENZE, NO VINCOLI

→ C Scelta designer sito web, 4, A, FS

→ D Scelta programmatori, 3, A, FS

→ E Scelta amministratori di sistema 6, A, FS

→ F Scelta amministratori di rete 5, A, FS

→ G Scelta dispositivi hardware infrastruttura, 5, (E, F), FS

→ H Installazione database, 8, G, FS

→ I Installazione server, 12, G, FS

→ J Installazione Gruppi di continuità, 5, I, FF+1

→ K Creazione database, 16, (H, B), FS

→ L Testaggio funzionamento database, 7, K, FS

→ M creazione programma server, 10, (D, J), FS

→ N Creazione sito web, 19, (C, L, M), FS

→ O Testaggio funzionamento sito web, 7, N, FS

→ P scelta email automation provider,6, F, FS

→ Q pubblicazione sito web, 3, (O, P), FS

NOTE

Si specifica che planner (di default) gestisce automaticamente le durate delle attività non tenendo conto dei giorni sabato e domenica, quindi le attività potrebbero essere traslate per questo motivo e quindi differire da quelle del CPM. Planner riduce la durata delle attività in proporzione alle risorse umane ad esse assegnate. Per quanto riguarda la stima delle risorse umane tramite rappresentazione grafica, il programma prevede una sezione dedicata che genera automaticamente. Nonostante questo, ho deciso comunque di sviluppare la RBS.

Il grafico del calcolo del percorso critico è un metodo in cui le attività vengono rappresentate nei cosiddetti “box” collegati da frecce (che rappresentano i vincoli di precedenza) e mostra i collegamenti delle attività a cui sono vincolate o che vincolano. A valle dei calcoli delle durate, un percorso risulta critico (quello evidenziato in rosso), ovvero l’insieme delle attività che non devono ritardare per non causare ritardi all’intero progetto. Le altre attività possono iniziare e finire entro i limiti stabiliti da LS e LF (Data d’inizio e di fine al più tardi).

Il diagramma di Gantt permette di visualizzare graficamente le attività in relazione al tempo, in quanto sull'asse delle y vengono rappresentate le attività e sull'asse x queste vengono scandite in base al tempo suddiviso in giorni e settimane.

La Work Breakdown Structure (o WBS) è la rappresentazione grafica delle attività in maniera gerarchica, dettagliandole il più possibile e rendere chiara l'assegnazione di personale, costi e risorse. Naturalmente è stata adattata a quelle che sono le richieste della traccia.

La Resource Breakdown Structure (o RBS) è anch'essa la rappresentazione grafica delle risorse. Ho voluto riportare le risorse materiali, oltre a quelle umane, anche se non richieste, per avere una RBS completa.

Quarta Parte: Sistemi e Reti

Il servizio sanitario ha bisogno di memorizzare i dati in un database, che deve essere collegato a tutti gli edifici sparsi per l'Italia. Naturalmente, il database contiene informazioni sensibili e personali e, di conseguenza, ha bisogno di essere protetto. Dato che è possibile che ogni tanto un nuovo dispostivo venga inserito in questa rete, l'assegnazione manuale di indirizzi IP costituirebbe una perdita di tempo, risorse e costi al sistema. Pertanto sarebbe ottimale la gestione automatica dell'assegnazione di indirizzi ip tramite il protocollo DHCP. L'invio di dati attraverso la rete deve essere necessariamente gestito da sistemi di sicurezza avanzati: la crittografia.

Per quanto riguarda la postazione medica, serve una rete privata LAN/WLAN (dipende dal collegamento tramite cavi o wireless) di classe C (254 host dovrebbero essere più che sufficienti), domestica come quelle che abbiamo tutti nelle nostre abitazioni; basta semplicemente un modem-router che interfaccia la rete pubblica con quella privata all'interno della postazione medica. Nel caso in cui l'edificio fosse abbastanza grande da non permettere a questo dispostivo di coprire tutta l'area, non credo che utilizzare i cavi sia il metodo più conveniente sia in termini di costi che di comodità e lavoro; sarebbe possibile quindi utilizzare i cosiddetti "repeaters" o "range extenders", che amplificherebbero il segnale ripetendolo e quindi rendendolo disponibile in aree non accessibili dal router principale (il tutto in modalità wireless, a differenza degli "Access Points"). Anche in questo caso basterebbe assegnare gli indirizzi in modalità automatica, poiché non sono presenti server che hanno la necessità di avere un indirizzo statico. I software necessari sono un sistema operativo e un browser che permettono l'accesso alle pagine web del sito hostato dal server.

Nel sistema centrale, invece, tutte le richieste che vengono ricevute sono filtrate attraverso un router firewall, dopodiché vengono passate ad un "load balancer", che permette la distribuzione del carico di elaborazione tra più server, permettendo in questo modo scalabilità e affidabilità dell'architettura nel suo complesso. In base al numero di richeste vengono collocati un numero indefinito di server, che devono fare riferimento ad un grande database che contiene tutti i dati. Stavolta , ad ognuno di essi viene assegnato un indirizzo ip statico, in modo da essere riconosciuti e quindi raggiungibili. Il server necessita di un sistema operativo preferibilmente ottimizzato per i server e un web server, che permette di gestire le richieste di trasferimento di pagine web dei client.

Le informazioni che ciascun medico deve inviare al sistema centrale (e viceversa) viaggiano nella rete pubblica: per questo hanno bisogno di essere trasportate in modo sicuro. HTTPS è un protocollo che, a differenza di HTTP, il quale trasporta sulla rete il testo in chiaro, consiste nella comunicazione all'interno di una connessione criptata, che utilizza un sistema di crittografia asimmetrica (con una chiave privata ed una pubblica), dal Transport Layer Security (TLS) o dal suo predecessore, Secure Sockets Layer (SSL), permettendo la riservatezza. Una volta arrivate al server, dopo essere state filtrate dal firewall e distribuite dal load balancer, le informazioni vengono memorizzate nel database e in quelli di backup per evitare perdite di dati.

Ogni utente avrebbe la sua area personale, a cui potrebbe accedere utilizzando un sistema di autenticazione con nome utente o mail e password. Per ottenere un altro livello di sicurezza, potrebbe essere utile aggiungere l'opzione di utilizzare l'autenticazione a due fattori, che consiste nell'utilizzo di un doppio sistema di accesso. Il sistema sanitario potrebbe aggiungere un'altra modalità di accesso attraverso lo SPID.

Il controllo di flusso verrebbe gestito dal protocollo TCP (Transmission Control Protocol), che manterrebbe l'integrità dei dati ed eviterebbe che il mittente inviasse una quantità eccessiva di dati che potrebbero genererare una perdita di pacchetti tramite il meccanismo delle sliding windows. Il protocollo TCP è orientato alla connessione, è affidabile perché assicura che tutti i dati vengano trasmessi completamente e possano essere ricomposti dal destinatario nell'ordine corretto proprio grazie al controllo di flusso menzionato sopra. Le informazioni inviate via email verrebbero gestite dal protocollo SMTP, che permette l'invio di email. Volendo aumentare ulteriormente il livello di sicurezza, è possibile rendere sicura la connessione con TSL per assicurare a mittente e destinatario, a livello applicativo, la garanzia della comunicazione e l'integrità delle informazioni inviate.

Un progetto così complesso, esteso ad una nazione, deve garantire la sicurezza, l'affidabilità e la continuità del servizio. Tutti i dati che vengono inseriti, modificati o eliminati dal database devono essere registrati in uno o più dispositivi di backup (dislocati anche in posti diversi per evitare che eventi imprevisti in un luogo li distruggano senza possibilità di recupero). Inoltre in caso di manutenzione del database (sia a livello hardware che software) sarebbe possibile assicurare il servizio 24 ore su 24 e 7 giorni su 7. Per evitare imprevisti, è necessario installare diversi gruppi di continuità, per garantire elettricità in caso di cali di tensione, blackout o

altre anomalie. Dato che generano calore, i database devono essere inseriti in stanze adeguate provviste di sistema di raffreddamento, impianto antincendio, antiallagamento e altri sistemi che garantiscano la sicurezza.