2022

PAMAC: realizzazione di un’interfaccia per il consulente della salute e relativa documentazione

VALENTINA VILLA

UNIBG

Sommario

Progetto PAMAC 2
Di cosa si tratta? 2
Team e organizzazione 4
Standard, linee guida e procedure 5
Sviluppi 6
Interfaccia consulente della salute 7
Introduzione 7
Organizzazione del lavoro 8
Requisiti 10
Architettura 11
Design 13
Testing 16
Manutenzione 17

Progetto PAMAC

Di cosa si tratta?

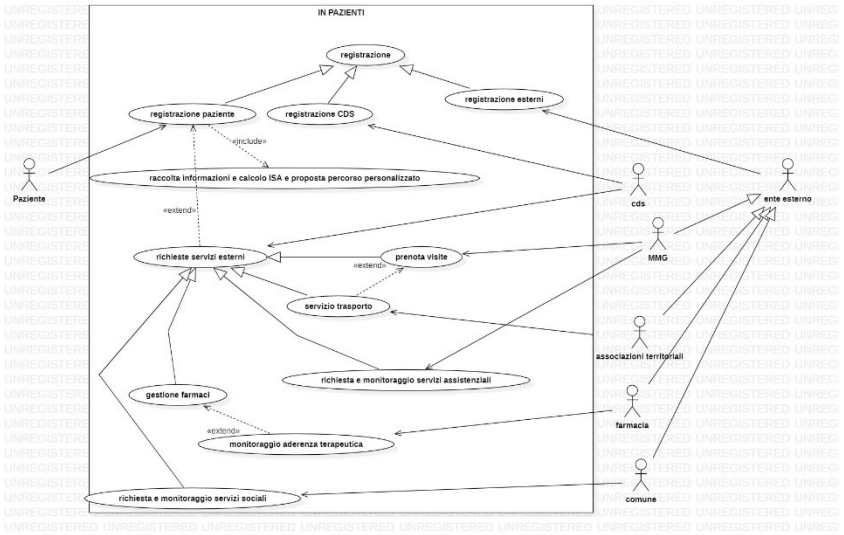
Il progetto PAMAC nasce per creare un software ideato per facilitare l'accesso ai servizi sanitari ai cittadini over 65. L'utilizzo sempre crescente della tecnologia e la sua enorme accelerazione negli ultimi anni, anche a causa dell'emergenza sanitaria, hanno portato ad un miglioramento a supporto dei pazienti, lasciando però meno spazio a coloro che già presentavano un disagio con la tecnologia stessa. Si viene conseguentemente a creare un contesto in cui il "paziente over65 solo" si ritrova "sempre più solo". Il progetto PAMAC si pone l'obiettivo di aiutare il paziente over 65 ad abbattere le "barriere digitali", nel contesto sia sociale che assistenziale, agevolando e rafforzando l'alfabetizzazione digitale, in quanto spesso non riesce a "tenere il passo" con l'evoluzione della nostra sanità digitalizzata.

Il servizio offerto vuole essere di supporto alle necessità dei sistemi esistenti, spesso in difficoltà nella loro interazione ed alleggerire i processi burocratici che spesso rallentano le attività, contribuendo ad aumentare il senso di solitudine e disagio in cui si vengono a trovare i cittadini. Inoltre, il progetto PAMAC si occuperà di collaborare con alcuni enti quali la protezione civile, i volontari, le farmacie territoriali, i Comuni e i servizi sociali, per quanto riguarda la fornitura di servizi, quali il trasporto durante l'esecuzione di visite cliniche dei cittadini over 65. Ciò viene attuato mediante la creazione di una piattaforma di interoperabilità dei dati sanitari per monitorare il follow-up iniziale e successivo alla presa in carico dei cittadini over 65: questa piattaforma permette di effettuare una classificazione della fragilità delle persone assistite sulla base dello stato funzionale (indipendenza-disabilità) e sociale; stratificazione prognostica e monitoraggio tramite allerte.

Sarà necessaria la presenza di un consulente della salute (CDS) che si interfaccerà tra le richieste del paziente e la piattaforma. Poiché la piattaforma tratterà dati sensibili è fondamentale gestirli secondo le normative vigenti.

1. Deve essere implementata un'area ad accesso pubblico con un'area per il login, una per la registrazione e aree informative per pazienti, medici, volontari e per enti esterni (farmacie, comuni,..).
2. Possibilità di registrarsi come paziente, consulente della salute, medico di medicina generale, volontario o ente esterno;
3. Chi è registrato come paziente deve:
    - poter compilare il questionario per il calcolo dell'indice di fragilità,
    - poter visualizzare i suoi dati nell'anagrafica,
    - poter visualizzare nella propria dashboard le informazioni di contatto relative al proprio cds, le ultime notifiche registrate nella piattaforma relative alle attività richieste e pianificate e il riepilogo dei servizi e appuntamenti, nonché la possibilità di richiederli direttamente dalla dashboard. Tra i servizi che l'utente deve poter richiedere ci sono:
        a) richiesta servizio di trasporto;
        b) richiesta di prenotazione visite (che verranno poi monitorate dal CDS);
        c) richiesta di assistenza fiscale;
        d) richiesta di presidi/ausili/device;
        e) richiesta di consegna farmaci e pasti;
        f) eventuale richiesta servizi socio-assistenziali e sociali;

4. Chi accede come volontario deve:
    - poter indicare che tipo di servizio vuole offrire (trasporto pazienti, consegna/ritiro farmaci)
    - poter indicare la sua disponibilità, per esempio attraverso un calendario;
    - poter accedere solo alle informazioni necessarie dell'utente al quale viene assegnato per il determinato servizio
5. I servizi necessari a chi accede come Consulente della Salute sono elencati in seguito.



Nello Use Case Diagram a sinistra sono illustrate tutte le funzionalità che si pensa di implementare in futuro; oltre a quelle già nominate ci sono infatti le richieste di monitoraggio per servizi assistenziali e servizi sociali gestiti rispettivamente dal medico di medicina generale (MMG) e dal comune. Inoltre si pensa di implementare un monitoraggio dell'aderenza terapeutica del paziente gestita con l'aiuto delle farmacie.

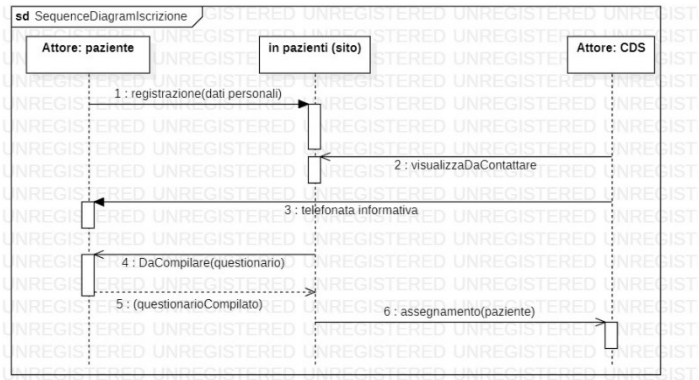
Tutte queste funzionalità verranno implementate tramite micro-servizi in modo da ottimizzare scalabilità e interoperabilità del sistema. Per implementarlo saranno inoltre necessari:

- Un'intelligenza artificiale che calcoli l'indice di fragilità del paziente a partire dai dati forniti dal questionario;
- Un algoritmo che assegni a ogni paziente un CDS;
- Un algoritmo che assegni a ogni volontario un paziente che ha richiesto il servizio da lui offerto.

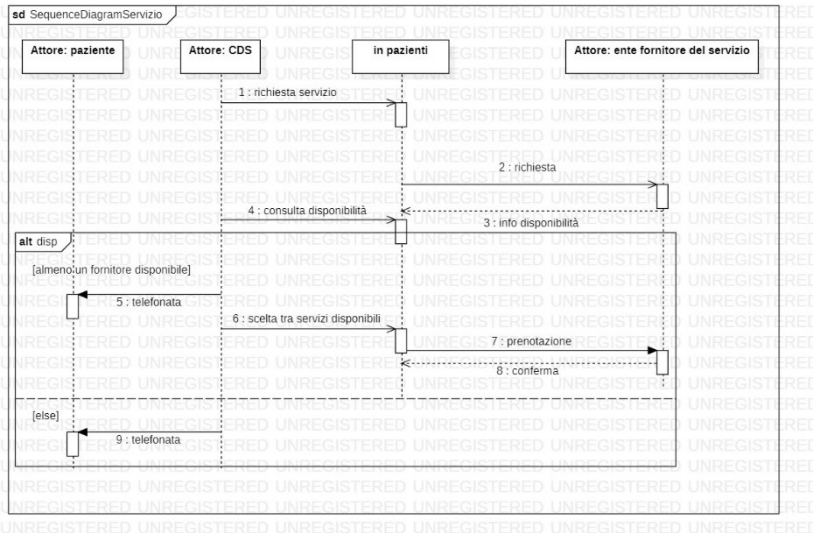
In futuro si vorrebbe implementare anche un'app mobile tramite cui erogare tali servizi ai pazienti.

Nei sequence diagram (un tipo di diagramma che modella le interazioni tra gli oggetti nel sistema) riportati in seguito si possono osservare due esempi di interazione data la richiesta di un servizio. Nel primo diagramma è riportato il caso in cui un nuovo paziente si voglia registrare al sito web: una volta effettuata la registrazione, il paziente viene assegnato a un consulente della salute il quale, dopo aver visionato i suoi dati, sarà tenuto a fare una telefonata per illustrare i passi successivi al cliente. In particolare quest'ultimo

dovrà compilare un questionario che darà una visione completa del quadro clinico, psicologico e sociale del paziente dopodichè verrà ufficialmente assegnato al consulente della salute.



Nel secondo sequence diagram è riportata la gestione di una richiesta che può provenire dal consulente della salute, dal paziente o dal sistema stesso in caso di visite periodiche; come si può osservare nel diagramma, una volta che la richiesta giunge al sistema, esso mostra le disponibilità fornite dagli enti esterni, in particolare il frammento alt evidenzia nel primo caso come avanza la procedura se gli enti esterni forniscono almeno una disponibilità: viene contattato il paziente per confermare la prenotazione che, se approvata, viene immediatamente effettuata. Nel secondo caso, quello in cui gli enti esterni non abbiano alcuna disponibilità, viene semplicemente fatta una telefonata al paziente per informarlo, la richiesta rimarrà nel sistema e si ripeterà la procedura illustrata dal diagramma dopo un breve periodo di tempo.



Team e organizzazione

Il team è composto da tre membri:

-Aldo Farese, che si occupa dello sviluppo web e front-end del progetto;

-Marco Spreafico, che si occupa dello sviluppo ed integrazione della parte d'intelligenza artificiale;

-Mariangela Vanalli, che si occupa dell'organizzazione del progetto, del coordinamento del team e delle interazioni con enti esterni.

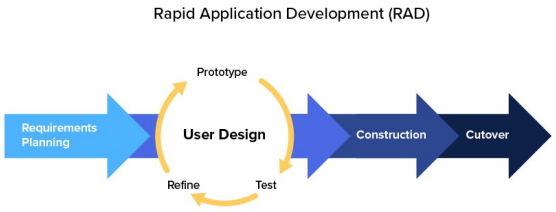
Viene applicata una gestione delle persone di tipo SWAT (Skilled Workers with Advanced Tools) dove le comunicazioni tra membri sono brevi ed informali, a lunghe riunioni formali si preferiscono sessioni di brainstorming e workshops ed essendo i membri del team fortemente specializzati in ambiti differenti ognuno di essi si concentrerà maggiormente sull'ambito di sua competenza, utilizzando i tool ritenuti più idonei.

È comunque necessario avere una costante visione d'insieme del progetto e per farlo vengono tenute riunioni settimanali in cui viene discusso lo stato del progetto, i successivi passi da eseguire e la suddivisione del lavoro. Tali riunioni vengono fatte attraverso Google Meet.

Per la comunicazione e la cooperazione tra i membri del team verranno utilizzati:

- Google meet per le riunioni del team;
- Gmail e Whatsapp per brevi comuniucazioni;
- Google Calendar per la gestione delle attività;
- Bitbucket e Github per condividere il codice.

È stato scelto di seguire la modalità di sviluppo RAD (Rapid Application Development) perché adatto a progetti seguiti da poche persone qualificate. Infatti la modalità RAD consente una buona velocità di sviluppo poiché il rapid prototyping permette di avere un feedback immediato e da' la possibilità agli sviluppatori di implementare più aggiornamenti velocemente ripartendo dai risultati dell'ultimo prototipo. Questa modalità aiuta ad assicurare che il software sia in linea coi requisiti dell'utente e che chi lo implementa sia focalizzato sulla qualità del software stesso.



I requisiti sono stati stabiliti durante l'ideazione del progetto; essi non sono stati forniti da un cliente ma essendo il progetto di natura market-driven sono stati ricavati tramite una scenario based analysis. Attraverso triage viene data una priorità ai requisiti, dopodiché vengono assegnati degli slot temporali entro i quali gli sviluppatori cercheranno di implementare quanto più possibile del requisito loro assegnato. Al termine dello slot temporale verrà tenuta una riunione col team durante la quale si discuterà e approverà quanto fatto o si proporranno eventuali modifiche. Una volta completata ed approvata l'implementazione di un requisito, si potrà passare al successivo requisito o, se necessario, si migliorerà un requisito già precedentemente implementato.

Standard, linee guida e procedure

Per assicurare la qualità del software verranno eseguiti test sul programma durante tutti gli stadi di sviluppo, in particolare per quanto riguarda i dati prodotti dall'intelligenza artificiale si divideranno i dati in due parti: training e testing, in modo tale da poter fare validazione dei risultati ottenuti.

Tutto il codice del progetto dovrà essere scritto rispettando le seguenti “best practices”:

I. Commentare: l’uso dei commenti consente di revisionare il programma più facilmente e renderà il codice immediatamente più comprensibile
II. Assegnare nomi chiari a variabili e metodi
III. Indentazione: un codice ordinato è più comprensibile
IV. Seguire il principio di “Separation of concerns”: si vuole suddividere la logica del progetto in unità indipendenti ognuna con un compito ben preciso, così facendo si riduce la complessità del progetto spezzandola in componenti più semplici, si semplificano le operazioni di testing e debugging, si facilita il riutilizzo di codice.
I. Scalabilità: il codice viene progettato per supportare la scalabilità in quanto per questo progetto è prevista l’aggiunta incrementale di nuove funzionalità.

Verranno utilizzati i seguenti linguaggi:
Per la creazione e gestione del sito web: vbScript, .NET, html e css;
Per la web app: Java;
Verranno trattati file nei formati: XML e JSON;
Per la parte di intelligenza artificiale e analisi dei dati: Python;

Tutte le procedure sopra citate servono a mitigare i rischi derivanti dallo sviluppo del software.
I principali individuati per questo progetto sono i seguenti:

- Cambiamenti nei requisiti
- Deviazione dagli standard
- La soluzione non soddisfa le aspettative
- Resistenza nell’ adottare questa soluzione da parte dell’utente finale
- Codice sorgente poco commentato
- Scarsa riutilizzabilità del codice
- Il software tratta dati sensibili, è dunque necessario rimanere sempre aggiornati sulle politiche per il trattamento dei dati

Come ulteriore accorgimento nelle riunioni il team controllerà periodicamente che il prodotto rispetti i requisiti e gli standard, rimarrà aggiornato sulle politiche per il trattamento dei dati sensibili, si assicurerà inoltre che il software non devi dalle specifiche e dai requisiti originariamente previsti, in caso si noti una tale deviazione si provvederà a correggerla.

Sviluppi

Il progetto ha recentemente raccolto attraverso l’ATS di Bergamo fondi per 50.000 euro; con il progredire si vorrebbero raccogliere ulteriori finanziamenti, eventualmente anche provenienti dalla Regione.

Per uniformarsi a standard europei è stato scelto di utilizzare, al posto del questionario appositamente creato, quello del Tilburg Fragilty Indicator. Questa scelta presenta un ulteriore vantaggio: la disponibilità di dati reali su cui poter addestrare gli algoritmi per il calcolo dell’ISA. Questi dati sono stati gentilmente forniti dalla dottoressa Anna Mulasso, la quale ha contribuito allo sviluppo dell’indicatore stesso e alla raccolta dati per la conferma della sua validità.
L’indice effettivamente usato dalla piattaforma, come già accennato, non sarà semplicemente il TFI (il quale è una sommatoria di punteggi assegnati a ogni domanda), ma esso verrà calcolato con l’ausilio di un’intelligenza artificiale la quale oltre ai dati presi dal questionario del TFI unirà eventuali note provenienti dal personale sanitario, dai medici e dai volontari e otterrà attraverso una rete neurale tale indice. In futuro si potrà integrare questo primo indice con altri riconosciuti dalla comunità medico-scientifica per poter avere una visione più completa delle condizioni dei pazienti, si vorrebbe inoltre mappare tale indice sul territorio e creare uno storico sullo stato dei pazienti in modo tale da poter fare previsioni basandosi su tali dati.

Dopo il primo rilascio, che si stima sarà a ottobre 2022, le nuove versioni sviluppate saranno messe a disposizione ogni qualvolta verrà inserito un nuovo servizio o migliorato un servizio esistente.

Questa prima versione avrà le seguenti funzionalità:

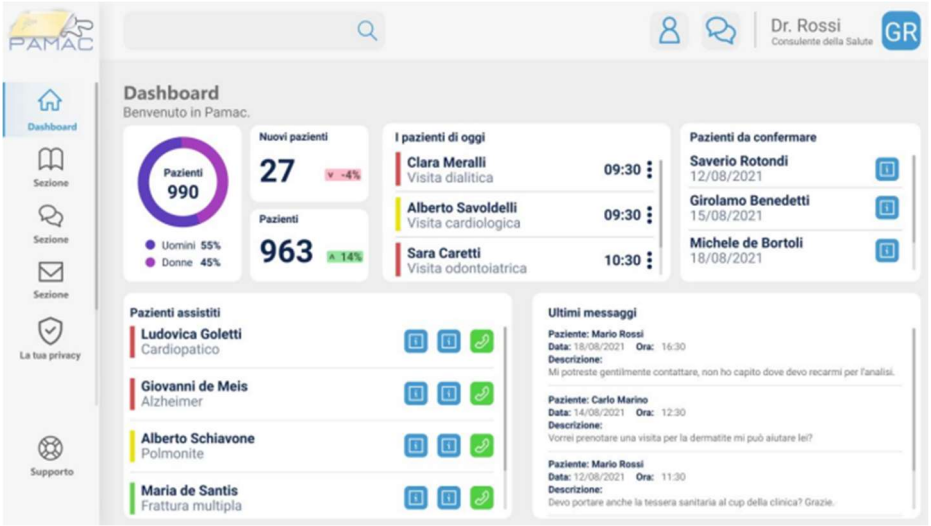
- Dal sito web sarà possibile registrarsi come pazienti e compilare il questionario del TFI; sarà dunque presente una pagina di anagrafica dove il paziente potrà consultare i propri dati;
- Calcolo dell'indicatore di fragilità: verrà calcolato a partire dai dati raccolti tramite il questionario citato attraverso un'intelligenza artificiale;

Interfaccia consulente della salute

Introduzione

Il consulente della salute (CDS) sarà colui che si interfaccerà tra le richieste del paziente e la piattaforma, è quindi fondamentale che l'interfaccia dedicatagli implementi le seguenti funzioni:

a) Poter visualizzare nella propria home un accesso diretto alle principali e fondamentali informazioni che l'operatore CDS deve avere sempre sotto controllo, ad esempio deve poter visualizzare statistiche sui pazienti a lui assegnati, le attività dei suoi pazienti schedulate per la data corrente, i pazienti da valutare tramite questionario, parte dell'elenco dei suoi assistiti, poter visualizzare gli ultimi messaggi inviati da essi.



i) Entrando nell'area dei pazienti assistiti deve poter: cercare i pazienti, per ognuno di loro dovrà inoltre poter accedere alla visualizzazione del sinottico del paziente e al suo diario socio-assistenziale, dovrà inoltre poter aggiungere nuovi pazienti ai suoi assistiti;

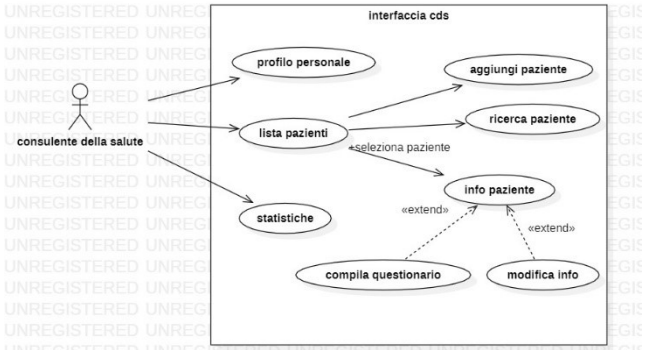
ii) Nel sinottico di ogni paziente viene rappresentato il dettaglio delle sue ultime attività, questa rappresentazione permette all'operatore di avere il massimo controllo nel processo di assistenza ritrovando in un riassunto tutte le pianificazioni effettuate o da effettuare, evitando di perdere di vista informazioni importanti;

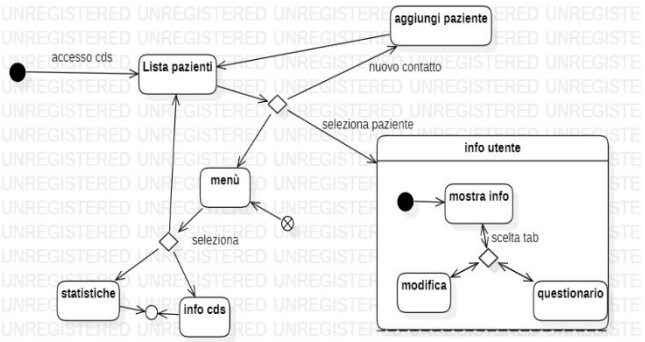
iii) Nel diario socio-assistenziale i l'operatore descrive le attività svolte e pianifica l'assistenza del paziente. Come in un vero diario clinico, le informazioni sono sempre aggiunte in ordine cronologico riportando i riferimenti sia temporali che dell'operatore che ha aggiornato il diario.

Ne risulta una vera e propria storia del paziente che racchiude tutti gli eventi di assistenza e cura scaturiti dal processo di assistenza.

b) Nell'area servizi il consulente della salute deve poter attivare una procedura guidata per la selezione e pianificazione del servizio richiesto dal paziente, in caso di richiesta prenotazione visita, la piattaforma integrerà i servizi offerti dalla regione stessa.

Queste sono le funzioni principali che l'interfaccia per il consulente della salute dovrà nel tempo implementare (visibili anche nello use case diagram e nello state machine diagram). Tutte queste funzionalità verranno implementate tramite micro-servizi in modo da ottimizzare la scalabilità e l'interoperabilità. Il linguaggio di programmazione scelto è Java e viene utilizzata la piattaforma per lo sviluppo di web app Vaadin.





Il primo rilascio, stimato per metà settembre 2022 dovrà avere le funzionalità del gruppo a). In seguito sarà possibile aggiungere nuove funzionalità e migliorare quelle esistenti e di conseguenza potranno essere rilasciate nuove versioni.

Organizzazione del lavoro

Il team è composto da un unico sviluppatore senza precedenti esperienze nell'uso di vaadin ma con buone conoscenze di Java. Inoltre il membro del team PAMAC (che per quest'applicativo rappresenta il cliente) Marco Spreafico parteciperà a riunioni settimanali dove oltre a dare un feedback immediato sul lavoro svolto potrà, avendo ottime competenze nella programmazione a oggetti, dare consiglio e se necessario fornire aiuto allo sviluppatore.

I software utilizzati sono quelli già elencati per il progetto nella sua interezza cosi come gli standard, le linee guida e procedure.

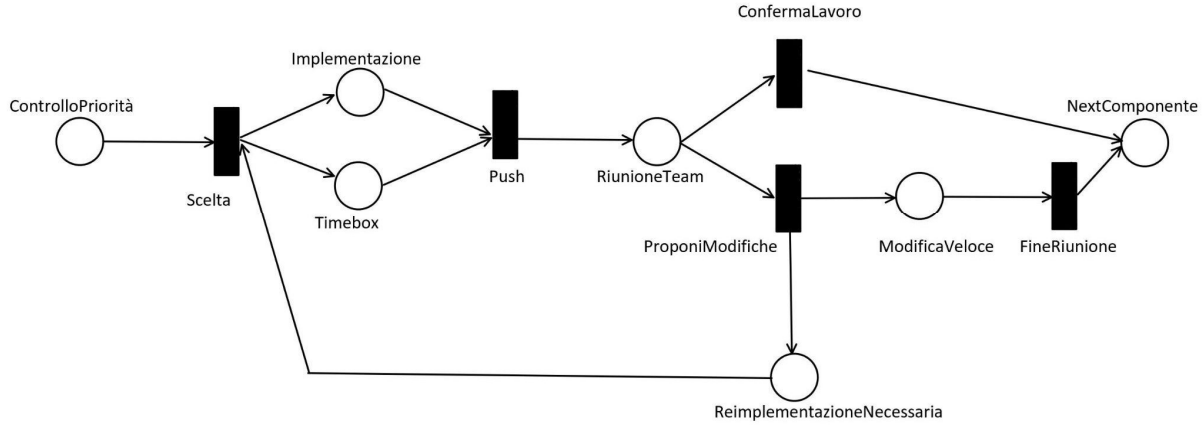
Oltre ai rischi associati al progetto PAMAC nella sua interezza, per questa interfaccia ne sono stati individuati ulteriori:

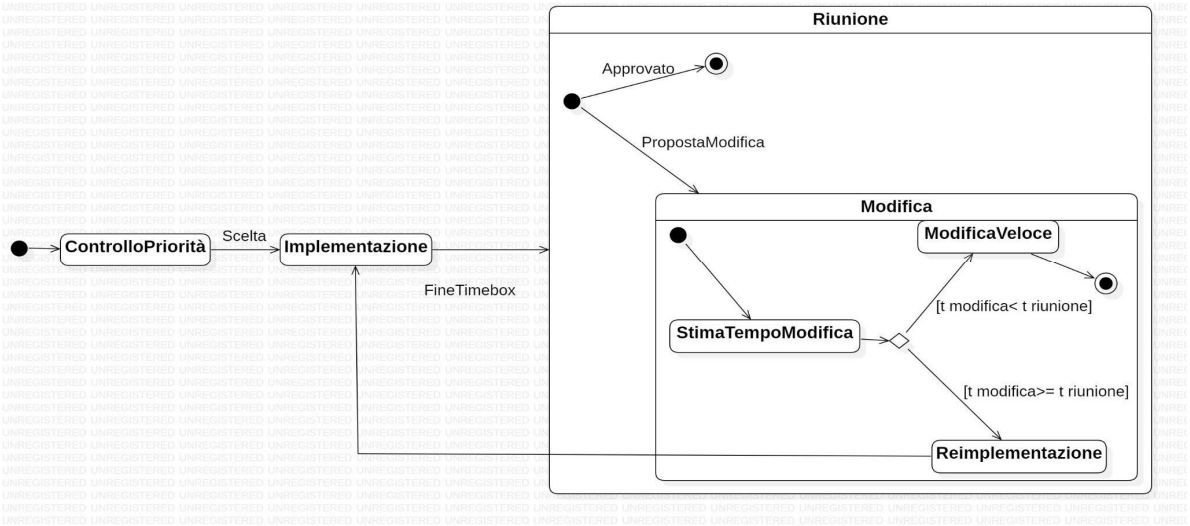
1. Difficoltà nell'Utilizzo di Vaadin, in quanto si tratta della prima esperienza con questa piattaforma;
2. Difficoltà nel risolvere problemi: lavorare in team è un eccellente modo per individuarli e risolverli velocemente, qui lo sviluppatore lavora da solo e potrebbe quindi avere qualche difficoltà a individuare eventuali errori.

Anche per questa parte del progetto è stato scelto di seguire una modalità di sviluppo del software di tipo RAD (Rapid Application Development) descritta nella prima parte del documento . Tramite la tecnica triage si assegnano priorità ai requisiti e sono stati scelti degli slot temporali (time box) di una settimana in cui lavorare su determinati componenti. Al termine della time box si tiene una riunione col cliente in cui viene

discusso quanto ottenuto. È opportuno che il cliente possa sempre avere a disposizione il codice aggiornato, motivo per cui è fondamentale l'utilizzo di Github.

Nella rete di Petri e nel diagramma degli stati UML di seguito, è' descritto il procedimento per confermare l'implementazione di un nuovo componente. Inizialmente viene scelto un componente controllando la lista delle priorità, dopodichè inizia la fase di implementazione la cui durata è di una time box. Al termine della time box il lavoro svolto viene condiviso col cliente che durante la riunione sceglie se confermarlo, apportare delle modifiche veloci (da intendersi come risolvibili durante la riunione stessa) o se è necessario re-implementare in un secondo momento il componente in questione.





Procedendo così per prototipi non si rischia di deviare dalle specifiche date in quanto si ha un continuo confronto con il cliente che entra a far parte del team.

La qualità del software è una proprietà intrinseca del progetto, pensata sin dalle prime fasi. Procediamo in modo che il software abbia:

- Qualità del prodotto:
    - Durante il funzionamento del software vogliamo garantire correttezza, prevenire un uso improprio del software stesso, affidabilità e un utilizzo intuitivo.

- Diamo importanza alla manutenibilità del nostro sistema: è mantenibile, flessibile e testabile.
- Il software è portabile su tutti i computer che hanno una JVM.
- Interfaccia: deve essere di facile utilizzo per i consulenti della salute, ma al tempo stesso permettere tutte le operazioni per la gestione del sistema.
- Privacy: dato che il software immagazzina i dati sensibili dei clienti, bisogna permettere solo a chi è autorizzato di accedervi.
- Velocità: il software deve essere reattivo in modo da fornire un'esperienza piacevole ai clienti.

- Qualità del processo:
  - Documentazione: le fasi di sviluppo del progetto sono documentate prima e durante lo svolgimento dello stesso seguendo lo standard IEEE 9001

Requisiti

La specifica dei requisiti si divide in quattro fasi:

1. *Elicitation*, l'estrazione dei requisiti avviene usando la tecnica basata sullo scenario (scenario based analysis), mettendosi quindi nei panni dell'utente finale che userà il software e pensando a ciò che dovrà fare con esso. Una volta estrapolati i requisiti saranno organizzati tramite il MoSCoW.

Must have	- visualizzare elenco pazienti, - aggiungere nuovi pazienti, - possibilità di modifica dei dati dei pazienti, -possibilità di visualizzare tutti i dati del paziente selezionato, - protezione dati pazienti.
Should have	- permettere ricerca pazienti, -buona user experience, - pagina personale con i dati utente.
Could have	- statistiche sui pazienti.
Won't have	- database con storico pazienti.

2. *Specification:* Secondo lo standard IEEE 830:
   - Introduzione: In questo documento si specificano i requisiti e le funzionalità richieste per un software per la gestione di pazienti. Il software vuole sviluppare le

funzioni descritte nell'introduzione che costituisce un punto di partenza per la fase di design.

- Descrizione generale: non sono presente un database non attualmente richiesto, potrà tuttavia essere implementato in seguito. Le funzionalità che dovranno essere integrate sono descritte nei requisiti
- Specifica dei requisiti*:*
    a) Requisiti funzionali: inizialmente l'utente dovrà vedere l'elenco dei suoi assistiti, con la possibilità di cercarli attraverso una barra di ricerca e di aggiungerne di nuovi attraverso una form. Si dovranno poter visualizzare statistiche riguardanti i pazienti e i dati dell'user. Cliccando sul paziente dovranno essere visibili le informazioni che lo riguardano e dovrà essere possibile modificarle.
    b) Requisiti non funzionali: il sistema deve gestire contemporaneamente un numero potenzialmente infinito di pazienti (almeno 100). I dati sensibili dei pazienti vengono protetti tramite l'utilizzo della visibilità dei campi. Il software deve inoltre poter rimanere attivo anche per lunghi periodi di tempo. Si vuole inoltre garantire una buona user experience: per farlo controlliamo la velocità di caricamento delle view, l'ordine dei loro contenuti e cerchiamo di creare un software che sia quanto più intuitivo possibile.

3. *Validation:* il processo di verifica e validazione sono svolti in ogni fase dello sviluppo del sistema. Vengono controllati più volte i requisiti assicurandosi che rispettino le proprietà di correttezza, completezza, consistenza, accuratezza, leggibilità e quindi siano adatti allo sviluppo del progetto.

4. *Negotiation:* i requisiti sono stati approvati direttamente dal cliente il quale è a tutti gli effetti membro del team, avendo un confronto costante attraverso meeting settimanali si può dire che la fase di negoziazione sia in continuo sviluppo.

Architettura

Lo standard IEEE 1471 fornisce una struttura generale per la rappresentazione dell'architettura del software. Gli elementi principali di questo standard sono le persone interessate dal sistema, view e viewpoint. Questi ultimi a loro volta sono suddivisi in tre classi: Module viewpoint, Component-and-connector viewpoint, Allocation viewpoint.

Come Module viewpoint, per rappresentare una vista statica del nostro sistema, illustreremo una vista a strati (Layered), dove rappresentando il sistema su una serie di livelli, nella quale gli elementi del livello N possono usare gli elementi appartenenti ai livelli inferiori ad N. Il software è pensato in modo che i consulenti utilizzino questa web app per accedere alle varie funzionalità presenti nelle diverse viste.

Il software è organizzato secondo l'architettura model view controller per cui:

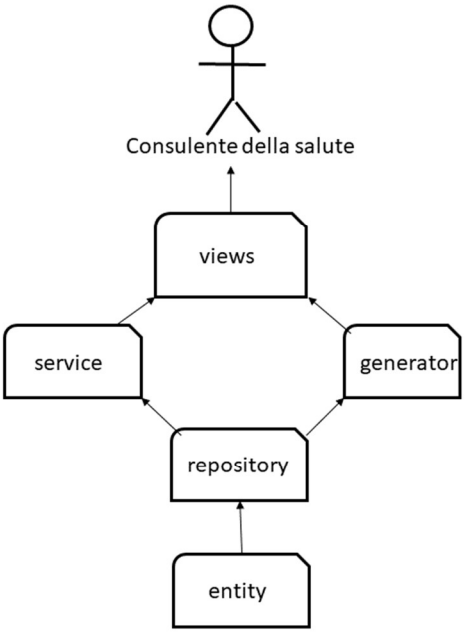
I componenti a livello più alto sono le viste (view), contenute nel package views; in questo package sono presenti tutte le viste a cui il consulente della salute può accedere suddivise in ulteriori package propri della

singola vista. Inoltre in views è presente il package utili che contiene tutte le classi utilizzate per dare una presentazione ordinata a tutte le viste del sistema.

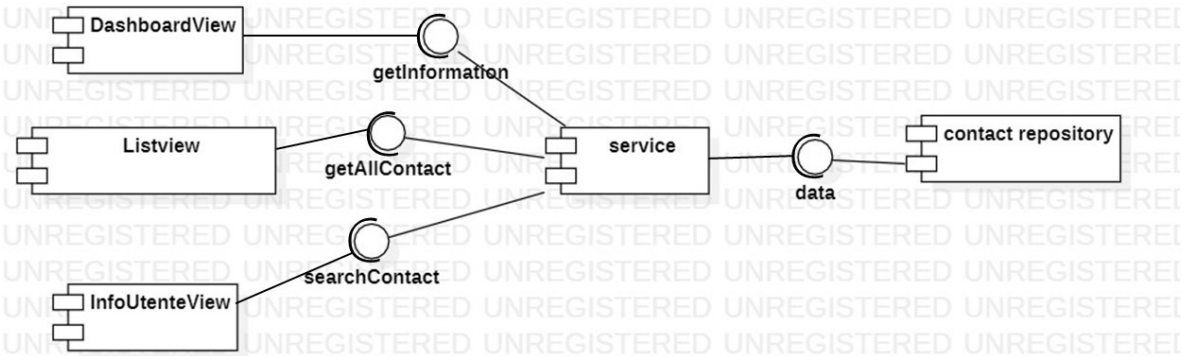
Nel secondo livello si trovano i package service e generator. Il primo fornisce servizi e metodi alle diverse viste mentre il secondo genera i dati che verranno poi visualizzati.

Scendendo troviamo il package repository che contiene interfacce per eseguire query e effettuare il salvataggio e la modifica dei dati generati.

Nell'ultimo livello troviamo: il package entity contenente le entità come contatto, visita, richiesta, ... Esse forniscono metodi per la creazione, modifica e consulto dei valori dei campi di queste classi.



Oltre al Module viewpoint costruiamo anche il Component-and-connector viewpoint nel quale si rappresenta la vista dinamica del sistema e, in particolare, come interagiscono tra loro i vari packaging e classi e quali interfacce vengono utilizzate. Tra i package si prendono in considerazione service, views e repository. Essendo il modello del tipo Model-View-Controller le viste (nel diagramma sono riportate le tre viste principali del sistema) interagiscono col controller (nel nostro caso rappresentato dalla classe service) il quale fornisce alle view metodi per accedere al modello, ossia ai dati contenuti nelle varie repository e rappresentate dalle classi del package entity (nel diagramma rappresentata contactRepository la quale contiene i contatti generati precedentemente) . La classe service ha un ruolo fondamentale perché fa da tramite tra le classi repository e tutte le viste che vogliono utilizzarle (simula quindi le funzioni di una classe che accede a un database simulato appunto dalle repository stesse).



Design

Poiché per il progetto è stato scelto di utilizzare UML e Java, entrambi linguaggi object-oriented, nella fase di design utilizziamo un metodo OOAD. In particolare scegliamo di usare il metodo Booch, seguendo in modo iterativo i seguenti passaggi:

- Identificazione delle classi, package e oggetti;
- Identificazione della loro semantica e comportamento;
- Identificazione delle relazioni tra le classi e package;
- Identificazione delle interfacce e implementazioni di classi e oggetti.

Seguendo lo standard IEEE 1016 viene documentato il processo di design come di seguito:

Identification	Service
Type	Classe
Purpose & Function	Deve fornire tutti i metodi per: modificare, cancellare e salvare i contatti fornire i dati per fare statistiche cercare i contatti
Dependencies	Usa le classi del package repository viene usato solo dalle classi del package views, infatti ricopre il ruolo del Controller nel modello Model-View-Controller
Interfaces	Sistema utilizza i metodi delle diverse repository. In particolare utilizza le query per cercare i contatti, ottenere i possibili valori del campo stato e del campo genere (campi di Contatto).
Data	Vengono salvate in campi appositi, all'interno della classe le repository che essa utilizza.

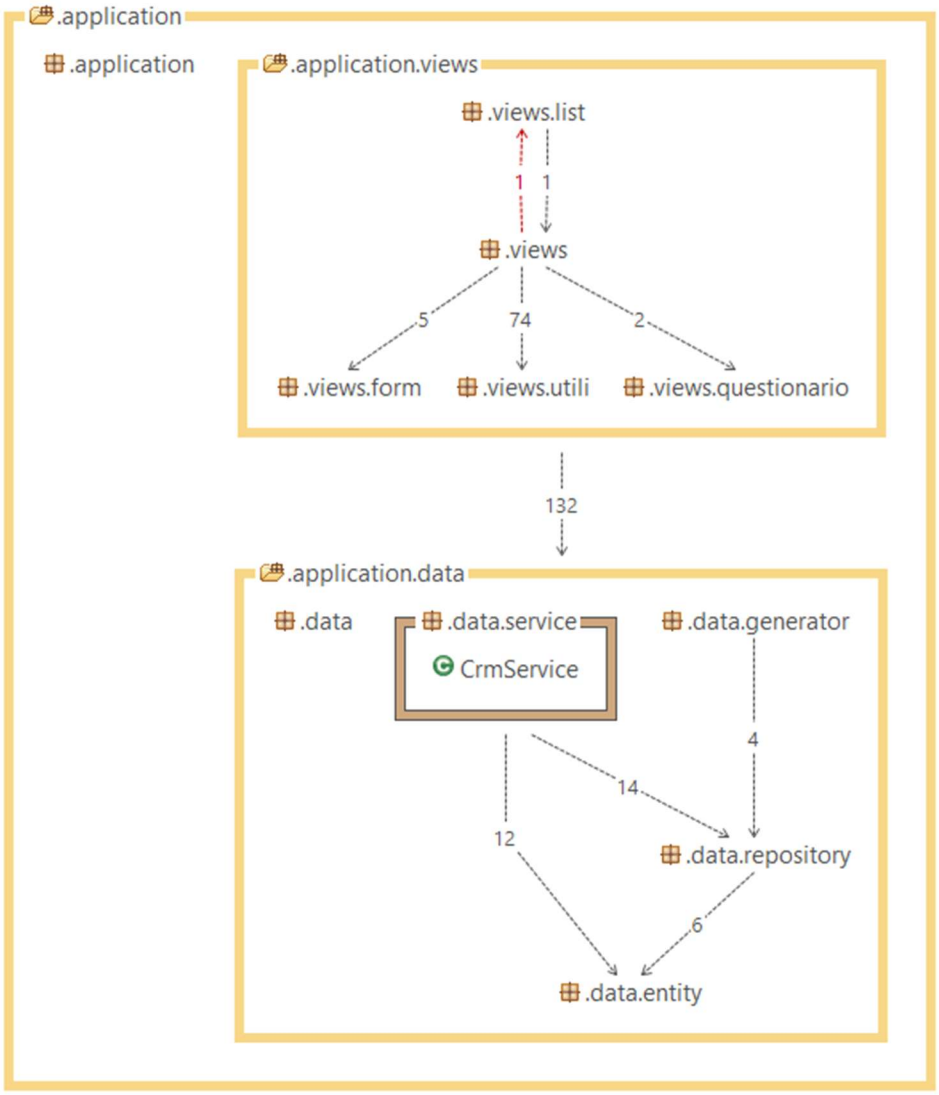
Identification	ListView	DashboardView
Type	Classe	Classe
Purpose & Functions	Consente la visualizzazione dell'intero elenco dei pazienti attualmente a carico del CDS, la ricerca di un paziente tramite una	Mostra le statistiche riguardanti i pazienti attualmente in carico al condìsulente della salute.

	barra di ricerca e l'inserimento di un nuovo paziente attraverso una form	
Dependencies	Utilizza la classe Service, non viene utilizzata da nessun altra classe, essendo appunto una vista per il modello Model-View-Controller.	Utilizza la classe Service, non viene utilizzata da nessun altra classe, essendo appunto una vista per il modello Model-View-Controller.
Interfaces	Utilizza il metodo di ricerca tramite nome del Servizio, oltre ai suoi eventi per il salvataggio di un nuovo paziente.	Utilizza i metodi che forniscono i dati per creare grafici della classe Servizi come ad esempio countContact(), findAllStatuses()…
Data	Contiene una form per il salvataggio di nuovi pazienti e mostra l'elenco completo dei pazienti	Mostra alcune statistiche ricavate dai dati ottenuti attraverso a classe Service

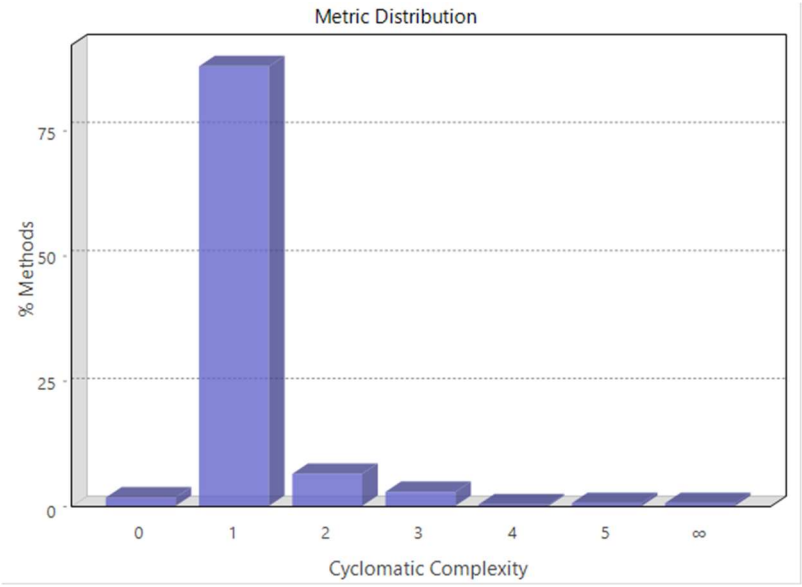
Identification	Entity
Type	Package
Purpose & Function	In questo package c'è il "modello" nel modello Model-View-Controller. Ossia contiene tutte le classi le cui istanze contengono i dati visualizzati nelle view. Ad esempio contiene la classe Contact che rappresenta i pazienti, la classe Richieste e Visite che rappresentano rispettivamente le richieste di un servizio da parte di un paziente e le visite fatte e in programma del paziente stesso.
Dependencies	Viene usata dalle repository e dalle view, non usa alcuna classe
Interfaces	Mette a disposizione della classe Sistema il costruttore e il metodo inserisciCertificazione()
Data	Gli attributi contengono le informazioni sensibili del cliente, vengono infatti gestiti come privati. Solo attraverso la classe Service si riesce a risalire ai suddetti dati, per visualizzarli, modificarli e cancellarli.

Il design del progetto è descritto dal seguente diagramma generato tramite il tool stan4j, in cui si nota che la Application è a un livello gerarchico superiore a quello delle View che a loro volta sono ad un livello superiore rispetto alla classe service, a livello più basso rimane il package entity che ha soltanto connessioni afferenti.

Il grado di accoppiamento è, come si può osservare nell'immagine sottostante, adeguato e segue il modello a strati descritto precedentemente. Il grafico è stato creato sulla base dei package e non delle singole classi, in quanto dato l'elevato numero di queste ultime il diagramma sarebbe risultato poco chiaro e comprensibile.



Inoltre con il tool stan4j calcoliamo la complessità ciclomatica di McCabe e osserviamo che la maggior parte dei metodi hanno una bassa complessità.



Si ricorre all'utilizzo di design pattern per la risoluzione di alcuni problemi tipici riscontrati durante lo sviluppo del progetto.

Pattern di design:

- Singleton: è fondamentale che la classe service venga istanziata una sola volta. Nessun'altra classe (nemmeno Application) può creare sue istanze che tuttavia devono essere disponibili alle classi utilizzatrici, la sua istanza viene infatti passata come parametro in ogni vista e salvata in un'apposita variabile per poterne utilizzare i metodi. Il costruttore di questa classe è privato.
- Delegation: è necessario che la classe Service utilizzi metodi di altre classi (ad esempio tutti i metodi di ricerca di un contatto). Per renderlo possibile colleghiamo la classe "delegator" Service alle classi "delegate" Repository tramite un'associazione diretta. All'interno della classe Service esistono quindi metodi che chiamano al loro interno metodi delle classi "delegate".

Testing

Per testare manualmente il codice si possono seguire le seguenti istruzioni che utilizzano tutte le funzionalità implementate dal codice.

1) Avviando il codice viene mostrata la prima vista con l'elenco di tutti i dipendenti, la barra di ricerca, un bottone per l'inserimento di un nuovo contatto e un menù laterale per poter navigare tra le views, inoltre runnando l'applicazione si testa automaticamente tutta la parte di generazione dei dati (DataGenerator) e la creazione della prima view.
2) Digitando nella barra di ricerca ne si può osservare il funzionamento, provando per esempio a digitare un nome o un cognome inesistente non comparirà alcun contatto mentre cercandone uno tra quelli presenti esso sarà l'unico a comparire.
3) Cancellando quanto scritto nella barra di ricerca tornerà a comparire l'elenco completo dei pazienti.
4) Dopodichè si può testare l'inserimento di un nuovo contatto, premendo sul bottone "Nuovo Contatto" comparirà sulla destra una form che richiederà di inserire i campi obbligatori, se essi non vengono compilati totalmente o la mail non è nel formato esatto il sistema non darà la possibilità di salvare questo nuovo contatto disabilitando il tasto "Save" della form.
5) Cliccando invece su un paziente compare una nuova vista contenente le sue informazioni personali tra cui foto, dati anagrafici, elenco delle richieste del paziente, elenco delle visite e risultati della compilazione del questionario. Cliccando sulla tab questionario viene avviata la compilazione di un nuovo questionario. Mentre cliccando sulla tab Modifica comparirà una form con all'interno i dati del paziente, modificandoli e premendo il bottone "Salva" si può osservare come i dati cambino nella parte delle info personali del paziente. Nella form è presente anche un tasto Delete il quale tuttavia non cancella i dati del paziente, infatti cliente non ha ancora scelto se implementare questa funzionalità direttamente nell'interfaccia del Consulente della salute, in caso potrà essere successivamente implementata.
6) Dal menù a sinistra è sempre possibile passare, da qualsiasi view sia aperta in quel momento, a una delle viste elencate nel menu.
7) Selezionando la vista statistiche dal menu a sinistra si aprirà una vista contenente diversi grafici, spostando il mouse su essi questi diverranno a rilievo.
8) Selezionando la vista con le informazioni personali del consulente della salute dal menù a sx comparirà una view con una foto e alcuni dati dell'utilizzatore di quest'interfaccia.

9) In qualsiasi vista ci si trovi, riducendo le dimensioni della finestra, la vista si adatterà a tali dimensioni.
10) Infine come ultimo test è possibile far generare al generator un grandissimo numero di contatti, verrà rallentato di molto l'avvio dell'applicazione ma essa sarà comunque in grado di gestire la grande quantità di contatti (ben oltre i 100 previsti nei requisiti non funzionali).

Manutenzione

Alla fine dell'implementazione del codice e dopo averne testato il funzionamento, svolgiamo delle operazioni di manutenzione riguardanti in particolare la facilità di utilizzo del software e la maggior leggibilità del codice.

La revisione principale è stata un'operazione di "pulizia" del codice, in modo che sia più leggibile da nuovi programmatori e per eventuali nuove operazioni di manutenzione. Essendo frutto del lavoro di un'unica persona, il codice risultava ben organizzato ma poco commentato e di conseguenza poco leggibile. È stato necessario quindi rielaborarlo in modo da avere un risultato migliore.

Sono state inoltre aggiunte delle notifiche sui bottoni le cui funzionalità non sono ancora state implementate per evitare di confondere l'utente con bottoni non funzionanti.