*Slide 1: Titolo*

Buongiorno,

Presento l’applicazione “Cartella Cardiologica Virtuale” realizzata durante il tirocinio.

*Slide 2: Introduzione*

Ho svolto lo stage presso l'azienda informatica “Sync Lab Srl” e, ho avuto la possibilità di lavorare allo sviluppo full-stack di un'applicazione web.

Tale applicazione consiste nella virtualizzazione della cartella cardiologica ponendo particolare importanza al calcolo del rischio cardiologico.

La Cartella Clinica è uno strumento utilizzato in ambiente sanitario per documentare il decorso clinico dei degenti; ovvero contiene tutti i referti medici e tutti i report delle visite che vengono effettuate ai pazienti durante la degenza.

Il Rischio Cardiologico invece è un indicatore utilizzato in ambiente sanitario che permette di valutare la probabilità che una persona vada incontro ad un rischio cardiovascolare maggiore come, ad esempio, un infarto del miocardio o un ictus.

Tale rischio viene calcolato in base alle Carte Del Rischio, ovvero delle classi di rischio definite in base ai fattori di: sesso, Età, pressione arteriosa, Colesterolemia, abitudine al fumo e diabete.

(Tale rischio viene definito in base ad alcuni parametri come età, sesso, ecc...)

*Slide 3: Funzionalità Principali*

Gli utenti che possono accedere all'applicazione vengono suddivisi in tre tipologie, ovvero l'amministratore, il medico e il paziente

L'amministratore ha la funzione di gestire gli utenti dell'applicazione, ovvero può inserire nuovi medici, nuovi pazienti e visionare i dati di tali soggetti.

Gli utenti di tipo medico e paziente hanno invece la possibilità di accedere alla cartella clinica e di visualizzare il rischio cardiologico.

In particolare i pazienti effettuano l'accesso alla cartella clinica e hanno la possibilità di visualizzare i propri documenti e il rischio cardiologico ad essi associato.

Mentre il medico può accedere alle cartelle dei diversi pazienti, inserire le visite e calcolarne il rischio cardiologico

Nelle slides successive vengono riportate alcune immagini che mostrano l'applicazione

*Slide 4: Interfaccia utente - Medico*

Questa immagine mostra l'area personale del medico dalla quale egli può visualizzare la lista dei pazienti presenti all'interno del sistema e per ognuno di loro può scegliere se:

-effettuare una visita, e in tal caso si aprirà un form che il medico dovrà completare inserendo tutti i dati relativi alla visita;

- scegliere se eliminare tale paziente;

-oppure può accedere alla cartella clinica del paziente dalla quale egli può visualizzare tutti i referti presenti, inserire o eliminare i referti e le visite oppure calcolare il rischio cardiologico

E anche in questo caso, si apre un form che il medico compila inserendo i diversi dati necessari.

*Slide 5: interfaccia utente – paziente (homepage)*

Questa immagine mostra l'area personale del paziente dalla quale egli può visualizzare i dati personali ed eventualmente modificarli.

Può visualizzare le proprie visite e il rischio cardiologico come mostrato nella seguente slide

*Slide 6: interfaccia utente – paziente (lista visite e rischio cardiologico)*

Nella prima immagine viene mostrata la lista delle visite effettuate al paziente, il quale cliccando l'apposito link può anche scegliere di stamparle; viene generato un file pdf con tutte le visite

La seconda immagine invece riporta il rischio cardiologico calcolato precedentemente dal medico e inserito automaticamente nella cartella del paziente.

*Slide 7: Design Pattern MVC*

Questa applicazione adotta il design pattern MVC, (ovvero Model View Controller.), in quanto è quello maggiormente indicato per la realizzazione delle applicazioni web

I design pattern possono essere definiti come delle soluzioni architetturali riutilizzabili in contesti simili;

più semplicemente il pattern definisce lo "scheletro" che deve avere il software, inteso come i livelli in cui è suddiviso il programma e l'iterazione logica tra di essi.

Come mostrato dall'immagine, il Model View Controller è composto da tre componenti:

il Model: che è il livello principale dell'intera architettura e ad esso è attribuita la gestione dei dati

il View: definito anche livello di presentazione, è costituito da tutte le funzioni che permette di definire le modalità con cui l'utente vede e interagisce con l'applicazione.

il Controller: è il livello di connessione tra il View e il Model. infatti riceve l'input dall'utente, e il view restituisce i dati in base alle regole definite nel model.

*Slide 8: Front-End*

Il front-end dell'applicazione, ovvero l'aspetto grafico, è stato realizzato utilizzando il CSS e il framework AngularJS.

Il CSS è un linguaggio che permette di gestire il design e la presentazione delle pagine web, quindi l'aspetto estetico, lavorando in collaborazione con l'HTML, ovvero il linguaggio con cui vengono realizzate le pagine web.

Attraverso il CSS è possibile stabilire le regole di formattazione dei contenuti all'interno della pagina web.

AngularJS è un framework sviluppato da Google che adotta il pattern architetturale MVC.

Questo framework è stato ha lo scopo di semplificare lo sviluppo del front-end delle applicazioni web.

AngularJS permettere di estendere le funzionalità delle pagine web ed è facilmente implementabile; infatti nella pagina web è sufficiente inserire la direttiva mostrata nella figura e poi "richiamare" le diverse funzionalità che offre questo framework attraverso degli appositi script all'interno della pagina.

*Slide 9: Back-End*

Il back-end dell'applicazione è l'implementazione di tutti i servizi offerti dal sistema.

In questa applicazione il back-end è stato realizzato utilizzando i framework Spring e Hibernate.

Spring è un framework utilizzato per la realizzazione delle applicazioni web implementate tramite il linguaggio di programmazione Java. (ovvero il linguaggio utilizzato per realizzare tale applicazione)

Tale framework offre la possibilità di realizzare delle applicazioni implementando il design pattern MVC.

Hibernate è un framework che si occupa di gestire autonomamente le operazioni e l'interfacciamento con il database.

In particolare Hibernate Gestisce le operazioni di Create, Read, Update e Delete e (inserimenti, lettura creazione ecc)

Inoltre offre la possibilità di mappare un qualsiasi oggetto java su un database relazionale.

Come mostrato in figura, una qualsiasi classe java, mediante i tag Entity e Table, viene gestita da Hibernate come se fosse una tabella del database relazionale.

Quindi tutte le istanze della classe verranno automaticamente trattate da hibernate come se fossero dei record del database.

*Slide 10: Conclusioni e Sviluppi futuri*

Questo tirocinio è stato molto importante per la mia formazione in quanto mi ha permesso di effettuare uno sviluppo full-stack di un'applicazione e quindi di vedere nel dettaglio sia l'aspetto del back-end sia del front-end.

Inoltre mi ha dato la possibilità di studiare e implementare degli strumenti che non avevo mai adoperato in precedenza, ovvero AngularJS, Spring e Hibernate.

Per quanto riguarda l'applicazione realizzata, le possibili modifiche che potrebbero essere apportate riguardano l'inserimento di nuove funzionalità quali, ad esempio, la presenza di un form/casella che permette al paziente di comunicare direttamente con il medico e viceversa.

Inoltre, per motivi di maggiore sicurezza, sarebbe importante integrare nel processo di registrazione all'applicazione, la possibilità di ricevere un’e-mail con la conferma dell'avvenuta registrazione.

Grazie per la vostra attenzione.