1. Realizzazione primo Mockup della finestra dell'app. Aggiunta della zona di input fornita all'utente per inserire i dati personali come nome, cognome, cod. fiscale, e del bottone per confermare. Aggiunta della sezione per inserire informazioni relative allo stato fisico dell'utente quali peso, altezza, età e gruppo sanguineo. Aggiunta della zona di input per inserire categoria, tipo, data della prestazione medica e dei bottoni per inserire in memoria i dati e per svuotare i campi input precedenti
2. Aggiunta al Mockup di una tabella per tracciare le prenotazioni dell'utente, e di un bottone per cancellare una prenotazione. Inserito grafico per avere una visuale sui vari tipi di prestazioni effettuate nei vari reparti. Aggiunta breve descrizione dell'app. Definizione casi d'uso per l'inserimento dati. Scelta dati modificabili con XML come numero di righe della tabella e dati per il server di log. Alla chiusura la cache salva i dati in input. Scelta dei dati da inviare al server di log remoto.
3. Rimozione dei campi di identificazione, età, peso, altezza e gruppo sanguineo. Rimossi i campi di input per la tabella. Inserimento di un campo email per identificare l'utente. Tabella resa editabile per inserire i dati. Aggiornamento del tipo di grafico utilizzato per il report e introduzione di un datepicker per selezionare il periodo di riferimento. Aggiunto un nuovo caso d'uso. La cache memorizza anche la selezione di una riga della tabella. Aggiornato il file di configurazione locale XML
4. Scrittura delle responsabilità delle classi. Stesura della classe ServerRemotoLog che gestisce il server di log che scrive su XML e di EventoDiNavigazioneGUI che invia gli eventi di log al server. Progettazione di ArchvioPrestazioni con relative funzioni per creare dati, in particolare quelle relative a inserimento, cancellazione e modifica degli elementi del DB. Bozza della classe GraficoPrestazioni e SelezionePeriodo per creare, aggiornare e modificare il grafico in attesa della prototipazione
5. Riguardati laboratori dove sono state trattate le TableView, per rivedere in particolare la loro creazione e il loro utilizzo. Cercati alcuni tutorial sulla selezione di una particolare riga e azioni eseguibili su di essa. Visionati laboratori riguardati l'integrazione e la gestione del Database e il relativo ottenimento dei dati da esso. Cercati alcuni tutorial per quanto riguarda la creazione e l'implementazione di un grafico a barre verticali, in particolare come creare e gestire gli assi.
6. Cercati ulteriori tutorial su come implementare un grafico a barre, riscontrati alcuni problemi per la gestione degli assi, in particolare come utilizzare numeri interi invece che decimali, e come differenziare le varie barre. Visti tutorial per importare dati dal DB e aggiornare il grafico. Tutorial su come implementare un datepicker per selezionare il periodo. Cercati video tutorial su come implementare una tabella editabile, in particolare con menu selezionabili all'interno delle celle stesse
7. Riguardati tutorial sull'utilizzo di HBox e VBox per la gestione dei layout orizzontale e verticale. Creazione della classe principale MyMedicalRecords che istanzia form e lo mostra sullo schermo tramite la scena chiamando i costruttori delle varie classi. Creazione della classe FormIscrizione per l'autenticazione dell'utente tramite email. Creazione della classe InterfacciaMedical per la gestione del layout dell'interfaccia applicativa. Aggiornato il diagramma delle classi UML dopo le modifiche

8. Creazione della classe TabellaPrestazione per istanziare e gestire la tabella contenente le prestazioni degli utenti, con la relativa funzione per aggiungere righe alla tabella. Creazione della prima parte del database su MySQL Workbench con la relativa tabella("prestazioni") per la gestione delle prestazioni degli utenti. Creazione della classe bean Prestazioni per inserire i dati in tabella e definizione delle funzioni di get. Aggiornato il diagramma delle classi UML con le nuove classi create
9. Creazione della classe ArchivioPrestazioni con le relative funzioni per lanciare query sql al database. Introduzione del metodo eventi() e aggiornaElementi() della classe principale per la gestione dei pulsanti e per l'aggiornamento degli elementi presenti sulla scena. Creazione e collegamento del pulsante Carica per caricare i dati dell'utente indicato nel form. Creazione classe EditingCelleTabella per rendere le celle della tabella editabili. Creati e collegati i pulsanti Aggiungi ed Elimina.
10. Collegati pulsati Aggiungi e Rimuovi per aggiungere o rimuovere un record dalla tabella. Creata funzione ottieniDatiInseriti nella classe Prestazioni per prelevare i campi editati dall'utente. Dichiarazione query sql nella classe ArchivioPrestazioni per aggiungere e rimuovere una prestazione dal db. Creata classe GraficoPrestazioni per gestire il grafico a barre e query sql per caricare i dati in esso. Creata la funzione aggiornaGrafico per recuperare i dati dal db e aggiornare live il grafico.
11. Creazione della classe SelezionePeriodo per creare e gestire un DatePicker, da utilizzare per selezionare il periodo di analisi nel grafico. Create le funzioni ottieniDataInizio e ottieniDataFine per prelevare le date. Aggiornata la funzione aggiornaElementi di MyMedicalRecords con l'aggiunta delle funzioni del DatePicker,perché il grafico venga aggiornato in seguito ad ogni evento collegato. Aggiornato il diagramma delle classi UML per verificare la correttezza delle classi aggiunte fino ad ora
12. Creazione della classe Cache che salva e preleva da/su file binario i contenuti compilati del campo EmailUtente e dei campi del DatePicker, per ricaricare l'utente e per avere il grafo nuovamente aggiornato alla riapertura. Creazione di un metodo SalvaDatiBin() che, alla chiusura dell'app, scrive su file binario e di un metodo CaricaDatiBin() che ricarica i dati all'apertura. Aggiunti i metodi importaData() e CaricaData() per esportare e importare le date da/nel file binario come tipo LocalDate.
13. Modificati gli eventi legati ai pulsanti nella classe generale MyMedicalRecords: funzione per l'inserimento di una nuova riga editabile della tabella usata solo dalla aggiornaElementi(), togliendo codice ridondante. Aggiunta del metodo updateItem() alla classe TabellaPrestazioni per gestire il colore delle righe della tabella: riga rossa se la data della prestazione è precedente alla data odierna e quindi già passata, o verde se la data è superiore e quindi la prestazione non ancora effettuata.
14. Aggiunta funzione nella classe Cache per salvare la selezione di una riga se questa non è stata eliminata. Aggiunto padding alla scena per staccare gli oggetti dai bordi dell'applicativo. Creato HBox per contenere il titolo MyMedicalRecords e fare in modo che fosse centrato, eliminate le dimensioni dalla scena in modo che si adatti autonomamente al contenuto. Scena non ridimensionabile con false nel campo setResizable di stage, creato VBox per la tabella e uno per il grafico in modo da centrarli
15. Creazione di SeverLogRemoto che riceve la riga di XML relativa a un evento e la scrive su file. Creata la classe GestoreEventoLog con creaLog che crea un nuovo evento, invio per mandarlo al server e valida per essere certi che l'evento sia stato generato correttamente

prima di essere inviato. Creata la class ParametriDiConfigurazione che ha il compito di salvare i dati caricati con l'apposita funzione carica, dal file config.xml, munita anche di tutti i metodi get per ottenere i dati da passare

16. Creata EventoNavigazioneGui che si occupa di creare effettivamente la stringa XML. Aggiunta di funzioni statiche in ParametriDiConfigurazione per restituire i dati del DB, numero delle righe della tabella e il periodo di default da valutare per grafico e tabella. Modificate le query per far in modo che restituiscano risultati compresi nel periodo, con aggiunta di LIMIT per rispettare il numero di righe. Utilizzo dei dati di connessione al DB. Aggiunta funzione di validazione della riga di log.
17. Modifica della funzione carica per validare config.xml prima di caricarlo, modifica del costruttore di ParametriDiConfigurazione per assegnare parametri di configurazione standard se quelli riportati da config.xml non sono validi. Creata settaGraficaGenerale() in InterfacciaMedical che si occupa di gestire la grafica di tutti gli oggetti di MyMedicalRecords, modificando tutta la grafica nell'apposita classe. Ho quindi collaudato l'applicativo seguendo i casi d'uso riportati nella documentazione.
18. Rimossi i pulsanti Svuota e Invia poiché non utilizzati. Revisione dei casi d'uso dell'applicativo con modifica delle parti riguardanti i precedenti pulsanti e l'utilizzo del DatePicker. Completato il diagramma UML finale. Pulizia del codice per rimuovere parti ridondanti, in particolare nella classe principale Wallet, e aggiunta dei commenti per spiegare eventuali metodi utilizzati. Collaudato l'applicativo sui casi d'uso aggiornati in precedenza e completamento del documento di collaudo finale