****

**POLITECNICO DI BARI**

**DIPARTIMENTO DI INGEGNERIA ELETTRICA E DELL'INFORMAZIONE**

CORSO DI LAUREA TRIENNALE IN

INGEGNERIA INFORMATICA E DELL'AUTOMAZIONE

Work Project

INGEGNERIA DEL SOFTWARE

**Titolo**

**Docente: Codice Gruppo: T6**

Prof.ssa Ing. Marina Mongiello

**Componenti gruppo:**

Damato Luigi Lele

Campo Fabrizio

De Frenza Dario Donato

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**Anno Accademico 2019/2020

Indice

1. Introduzione 3
2. Dominio di interesse 4
3. Tecnologie utilizzate 9
4. Il progetto 10
5. Validazione 17
6. Conclusioni 22
7. Bibliografia 23

**Introduzione**

Il sistema software realizzato consiste in un’applicazione per smartwatch con sistema operativo Android Wear OS e di una sua controparte per dispositivi mobili Android, per necessità di configurazione e settaggio.

L’applicazione Next consente di tenere traccia, sul proprio smartwatch, tramite notifica o gesture, delle informazioni relative all’orario universitario; l’obiettivo iniziale è stato quello di fornire all’utente un’interfaccia su smartwatch semplice ed immediata che consentisse di visualizzare in ogni momento e in ogni luogo le informazioni relative alla successiva lezione del proprio orario universitario.

I capitoli della relazione sono organizzati nel seguente modo: il primo capitolo è dedicato alla fase analisi, con lo scopo generale di specificare e documentare le funzionalità e i servizi che il sistema software deve possedere per inserirsi correttamente nel determinato contesto operativo con cui vuole interagire. A questo proposito il capitolo si apre con la definizione del problema che il sistema software è chiamato a risolvere. Segue poi, l’analisi del dominio applicativo con la specificazione delle regole e delle dinamiche del contesto con cui il sistema software deve interagire. Il capitolo si conclude con la fase di definizione e analisi della specifica dei requisiti.

Il secondo capitolo è incentrato sulle tecnologie, i tools e i linguaggi utilizzati durante l’intera durata del progetto.

Il terzo capitolo descrive i principali step della fase di progettazione che va dalla definizione dei casi d’uso, fino a un accenno alla fase di implementazione dell’applicazione.

Il capitolo conclusivo consiste in una relazione di validazione del progetto, con la descrizione della fase di testing dell’applicazione e un resoconto dei risultati ottenuti.

**Dominio di interesse**

Il team di sviluppo ha cercato di orientarsi verso lo sviluppo di un applicativo che rispondesse alle esigenze degli studenti del Politecnico di Bari, in quanto coerentemente e attualmente operativi in questo contesto.

A tal proposito è stato osservato, a seguito di quasi tre anni di corso nella realtà del Politecnico di Bari, che la maggioranza degli studenti faticano a ricordare la disciplina della lezione successiva che si apprestano a seguire, il nome del docente di riferimento, nonché l’aula in cui si svolge tale lezione.

Il problema nasce dall’ingente quantità di informazioni da ricordare in ogni momento della giornata universitaria e ogni giorno della settimana; informazioni che sono, peraltro, spesso soggette a cambiamento durante l’anno accademico.

In aggiunta, numerosi studenti, devono tenere traccia di più orari contemporaneamente, per esigenze legate alla volontà di seguire esami a scelta non appartenenti al proprio corso di laurea, o di seguire lezioni di anni precedenti.

Va inoltre sottolineato che studenti del primo anno e non solo, oltre alle difficoltà sopracitate, necessitano di conoscere anche la posizione dell’aula relativa a una data lezione, in quanto non ancora familiari con la struttura del Politecnico.

Comportamento comune riscontrato tra gli studenti, al momento della nostra analisi, è stato quello di aprire la galleria foto sul proprio smartphone, cercare la foto dell’orario in uso e consultarlo in ogni momento della giornata universitaria, per sopperire alle difficoltà appena descritte.

A tale fine, l’app Next intende rivolgersi e andare incontro alle esigenze di questa fascia di studenti che rappresentano la maggioranza degli iscritti ai corsi di laurea del Politecnico di Bari. In particolare, intende eliminare le azioni di consultazione dello smartphone, consentendo di disporre in maniera immediata ed efficace delle informazioni della lezione successiva, in ogni momento e a portata di smartwatch.

Come già accennato l’app si rivolge agli studenti del Politecnico di Bari in quanto fornisce supporto esclusivamente per le strutture e le informazioni didattiche di esso. Tuttavia, ciò non implica l’impossibilità di scaricare l’app da parte di studenti esterni ad esso, si pensi ad esempio agli studenti Erasmus che possono necessitare di avere supporto immediato relativamente a un certo numero di corsi e a un certo numero di lezioni.

Di seguito la specificazione degli stakeholder del nostro software con la distinzione in primari e secondari.

**Stakeholder** **principali**

Studenti del Politecnico di Bari: sono i principali utilizzatori del servizio e hanno completo accesso alle funzionalità e ai servizi disponibili, scaricare, salvare e personalizzare l’orario di interesse, visualizzare le informazioni relative a ogni lezione, visualizzare statistiche di frequenza relative alle lezioni e contattare i docenti di interesse.

**Stakeholders** **secondari**

Utente generico: colui che scarica il software per visualizzarne le funzionalità e le caratteristiche.

A seguito dell’analisi del dominio di interesse sono state delineate le funzionalità della Smartwatch App e della Mobile App e i requisiti che esse devono soddisfare. L’utente utilizzatore dell’App Next deve innanzitutto avere a portata di mano le informazioni aggiornate riguardanti la lezione successiva conformemente all’orario scaricato e salvato su Smartphone, ricevere e richiedere quindi le giuste notifiche in ogni momento.

Di seguito sono riportati i requisiti di sistema (SRS) e i corrispondenti requisiti architetturali (SAS).

**REQUISITI** **DI** **SISTEMA**

|  |  |  |
| --- | --- | --- |
| **SRS** | **SAS** | **Descrizione** |
| SRS-01 | 01,02,03,  04,05,07 | Il sistema deve consentire all’utente di aggiungere un nuovo orario nello storage dell’applicazione. |
| SRS-02.1 | 06 | Il sistema deve consentire all’utente di settare un’orario come attivo sul sistema. |
| SRS-02.2 | 06 | Il sistema deve consentire all’utente di visualizzare i file orario salvati nello storage dell’applicazione. |
| SRS-02.3 | 06 | Il sistema deve consentire all’utente di personalizzare un file orario. |
| SRS-02.4 | 06 | Il sistema deve consentire all’utente di eliminare un file orario. |
| SRS-02.5 | 06 | Il sistema deve consentire all’utente di rinominare un file orario. |
| SRS-03.1 | 08,13 | Il sistema deve consentire all’utente di attivare e disattivare le notifiche su smartwatch. |
| SRS-03.2 | 05 | Il sistema deve consentire all’utente di attivare e disattivare l’aggiornamento del file orario corrente. |
| SRS-03.3 | 09,10,11 | Il sistema deve consentire all’utente la modifica della lingua usata dall’applicazione. |
| SRS-03.4 | 13,14 | Il sistema deve consentire all’utente di selezionare i minuti di anticipo con cui ricevere le notifiche su smartwatch. |
| SRS-04 | 12 | Il sistema deve consentire all’utente l’importazione dei file orario. |
| SRS-05 | 06, 12 | Il sistema deve consentire all’utente la condivisione dell’orario. |
| SRS-06 | 13,15,16,  17,18,19 | Il sistema deve consentire all’utente di ricevere le informazioni della lezione successiva. |
| SRS-07 | 20 | Il sistema deve consentire all’utente di visualizzare le informazioni della lezione corrente. |
| SRS-08.1 | 21,22,23 | Il sistema deve consentire all’utente di visualizzare le info del docente della lezione corrente. |
| SRS-08.2 | 21,22 | Il sistema deve consentire all’utente di contattare il docente. |
| SRS-09 | 24,25 | Il sistema deve consentire all’utente di visualizzare le info dell’aula della lezione corrente |
| SRS-10.1 | 26,27,28 | Il sistema deve consentire all’utente di segnare la sua presenza a una lezione. |
| SRS-10.2 | 27, 29 | Il sistema deve consentire all’utente di visualizzare le info relative al registro presenza. |

**REQUISITI ARCHITETTURALI**

|  |  |
| --- | --- |
| **SAS** | **Descrizione** |
| SAS-01 | Il sistema, tramite apposita interfaccia su smartphone, deve consentire la scelta del corso di laurea e dell’anno accademico dell’orario che l’utente intende scaricare. |
| SAS-02 | Il sistema, tramite smartphone, deve scaricare l’orario selezionato dall’utente dal sito “www.poliba.it”. |
| SAS-03 | Il sistema, tramite smartphone, deve salvare il file orario nello storage dell’applicazione. |
| SAS-04 | Il sistema, tramite smartphone, deve gestire lo storage dell’applicazione. |
| SAS-05 | Il sistema, tramite smartphone, deve mantenere aggiornati gli orari selezionati dall’utente, se questi non sono stati personalizzati. |
| SAS-06 | Il sistema, tramite interfaccia grafica su smartphone, deve fornire all’utente una lista di orari salvati. Si può interagire con ogni file in suddetta lista tramite tap e menu fornendo opzioni di visualizzazione, modifica, eliminazione, condivisione, ridenominazione e settaggio come orario attivo sul sistema. |
| SAS-07 | Il sistema deve impostare un orario come attivo sul sistema, se esso è l’unico orario salvato. |
| SAS-08 | Il sistema, tramite apposito bottone su smartphone, consente di abilitare o disabilitare la ricezione di notifiche su smartwatch. L’impostazione è abilitata di default. |
| SAS-9 | Il sistema utilizza di default la lingua italiana. |
| SAS-10 | Il sistema, tramite interfaccia grafica su smartphone, fornisce una lista per la selezione della lingua italiana, inglese, francese e spagnola. |
| SAS-11 | Il sistema modifica il testo visualizzato nei menù di smartphone e smartwatch nella lingua selezionata dall’utente. |
| SAS-12 | Il sistema importa ed esporta file orario tramite il menu di condivisione Android di default. |
| SAS-13 | Il sistema, tramite smartwatch, deve gestire le notifiche. |
| SAS-14 | Il sistema, tramite apposita casella di testo su smartphone, consente di inserire un anticipo personalizzato. |
| SAS-15 | Il sistema deve fornire, tramite apposita interfaccia su smartwatch, un bottone che consente all’utente di ricevere una notifica con le informazioni sulla lezione successiva. |
| SAS-16 | Il sistema, tramite smartwatch, riceve da smartphone le informazioni relative alla lezione successiva. |
| SAS-17 | Il sistema, tramite smartwatch, aggiorna l’interfaccia utente con le info sulla prossima lezione, in base all’anticipo impostato. |
| SAS-18 | Il sistema stabilisce un canale di comunicazione bidirezionale tra smartphone e smartwatch. |
| SAS-19 | Il sistema deve gestire i messaggi di errore e inviare relative notifiche all’utente. |
| SAS-20 | Il sistema, tramite smartwatch, deve aggiornare l’interfaccia grafica dello smartwatch con le informazioni relative alla lezione corrente. |
| SAS-21 | Il sistema, tramite interfaccia grafica su smartwatch, fornisce un’icona con l’immagine di un determinato docente. Tramite click dell’utente, il sistema mostra le informazioni relative e un apposito button per contattare il docente. |
| SAS-22 | Il sistema, tramite smartphone, apre l’applicazione di posta elettronica di default per contattare il docente selezionato dall’utente. |
| SAS-23 | Il sistema, tramite smartphone, gestisce le info relative a ciascun docente. |
| SAS-24 | Il sistema, tramite smartphone, gestisce le info relative a ciascuna aula del Politecnico. |
| SAS-25 | Il sistema, tramite interfaccia grafica su smartwatch, fornisce un’icona che tramite click dell’utente consente di visualizzare la posizione dell’aula della lezione corrente. |
| SAS-26 | Il sistema, tramite interfaccia grafica su smartwatch, fornisce un pulsante per la registrazione della presenza alla lezione corrente. |
| SAS-27 | Il sistema, tramite smartphone, traccia i grafici relativi alle presenze. |
| SAS-28 | Il sistema, tramite smartphone, gestisce un registro delle presenze dell’utente relativo all’intero semestre e a un determinato file orario. |
| SAS-29 | Il sistema, tramite interfaccia grafica su smartphone, visualizzare le info relative al registro presenze. |

**Tecnologie utilizzate**

Per la realizzazione di questa applicazione sono stati utilizzati diversi framework e diversi strumenti software che hanno permesso una semplificazione del lavoro.

**Framework**: È stato utilizzato il framework di supporto sviluppato da Google per Android Wear OS conforme alle specifiche e ai pattern di Android Wear 2.0. Esso supporta le features quali Notification, Watch Faces, scambio dati, sincronizzazione.

**IDE**: L’ambiente di sviluppo del software è stata la versione 3.5.3 di Android Studio.

**Tools**: La stesura della documentazione UML è stata effettuata tramite Draw.io e Visual Paradigm.

**Linguaggio**: L’intero sistema software è scritto in codice sorgente object-oriented linguaggio Java.

**Tecnologie**: Il Bluetooth è lo standard su cui si basa il protocollo di comunicazione e trasmissione dati comunemente utilizzato dagli smartwatch.

**Progetto**

Di seguito è riportato il diagramma dei casi d’uso realizzato a partire dall’analisi delle specifiche dei requisiti e descrive tutti gli scenari di interazione degli attori con il sistema.

Nello specifico si veda che gli attori coinvolti nella modellazione sono l’utente e il sito ufficiale del Politecnico di Bari.

Immagine che contiene testo, mappa

Descrizione generata automaticamente

Nel diagramma delle sequenze sono stati definite le azioni che modellano gli scenari dei casi d’uso.

Immagine che contiene testo

Descrizione generata automaticamente

Segue il diagramma delle classi che ha consentito di descrivere concettualmente le principali classi implentative del sistema e le loro dipendenze.

Nello specifico sono state definite le seguenti classi:

**Orario**:

Modella l’oggetto orario attraverso gli attributi:

* Lista Lezioni: Lista delle lezioni divise per giorno.
* In uso: Specifica se l’orario è attivo sul sistema.
* Cdl: Corso di laurea.
* Registro presenze: Tiene traccia delle lezione frequentate per ogni materia.
* Anno corso: Primo, secondo o terzo.

e attraverso i seguenti metodi:

* Update: Aggiorna l’orario.
* Share: Condivide l’orario selezionato come parametro.
* Update registro: aggiorna il registro delle presenze.
* Get Next: Richiede la lezione successiva dopo tap dell’utente.
* Get Next by Anticipo: Richiede la lezione successiva in base all’anticipo impostato.
* Customize: Personalizza l’orario.
* Set As Main: Imposta l’orario come attivo sul sistema.
* Rename: Rinomina l’orario.
* Get Registro Presenze: Restituisce il registro presenze.

**Impostazioni**:

Modella l’oggetto impostazioni attraverso i seguenti attributi:

* Anticipo: Anticipo con cui ricevere la notifica su smartwatch.
* Attiva Notifiche: Valore che vale true se le notifiche su smartwatch sono attive.
* Lingua: Specifica la lingua di sistema.
* Anticipo Default: Anticipo di default settato a 15 minuti.
* Attiva Notifiche Default: Valore di default settato a true.
* Attiva Sync: Valore che vale true se è attiva la sincronizzazione dell’orario in uso.
* Lingua Default: lingua di default settata su ‘Italiano’

e attraverso i metodi:

* Getter e setter di lingua: Restituiscono e settano l’anticipo.
* Get Attiva Notifiche: Restituisce l’attributo corrispondente.
* Attiva Notifiche: Attiva le notifiche su smartwatch.
* Getter e setter di lingua: Restituiscono e settano la lingua.
* Getter e setter di Anticipo Default: Restituiscono e settano l’anticipo di default.
* Getter e setter di Attiva Notifiche Default: Restituiscono l’attributo corrispondente.
* Attiva Sync: Attiva la sincronizzazione.
* Disattiva Sync: Disattiva la sincronizzazione.
* Disattiva Notifiche: Disattiva le notifiche.

**Storage**:

Modella l’oggetto storage attraverso l’attributo:

* Lista Orari: Lista di tutti gli orari salvati sul sistema.

E attraverso i metodi:

* Get Orario: Restituisce l’orario.
* Add Orario: Aggiunge un orario allo storage.
* Delete Orario: Elimina un orario dallo storage.
* Save Edit: Salva le modifiche dell’utente su un orario.
* Import: Importa un file orario.

**Docente**:

Modella l’oggetto docente attraverso gli attributi:

* Nome
* Mail
* Ufficio
* Discipline
* Orario Ricevimento
* Giorni Ricevimento

e dai seguenti metodi:

* Parse Docente: Salva le info dei docenti.
* Getter e setter degli attributi nome, mail, ufficio, discipline, orario ricevimento, giorni ricevimento.
* Contact: Contatta il docente.

**Lezione**:

Modella l’oggetto lezione attravaerso gli attributi:

* Materia
* Docente
* Aula
* Ora inizio
* Ora fine
* Contatore Presenze: Conta le presenze dell’utente a lezione.

E attraverso i seguenti attributi:

* Getter e setter di materia, contatore, docente, aula, ora inizio, ora fine.
* Update Contatore: Aggiorna il contatore delle presenze.

e **Aula**:

Modella l’oggetto aula attraverso gli attributi:

* Nome
* Descrizione
* PosX, posY: Definiscono la posizione dell’aula su un file immagine salvato nel sistema.

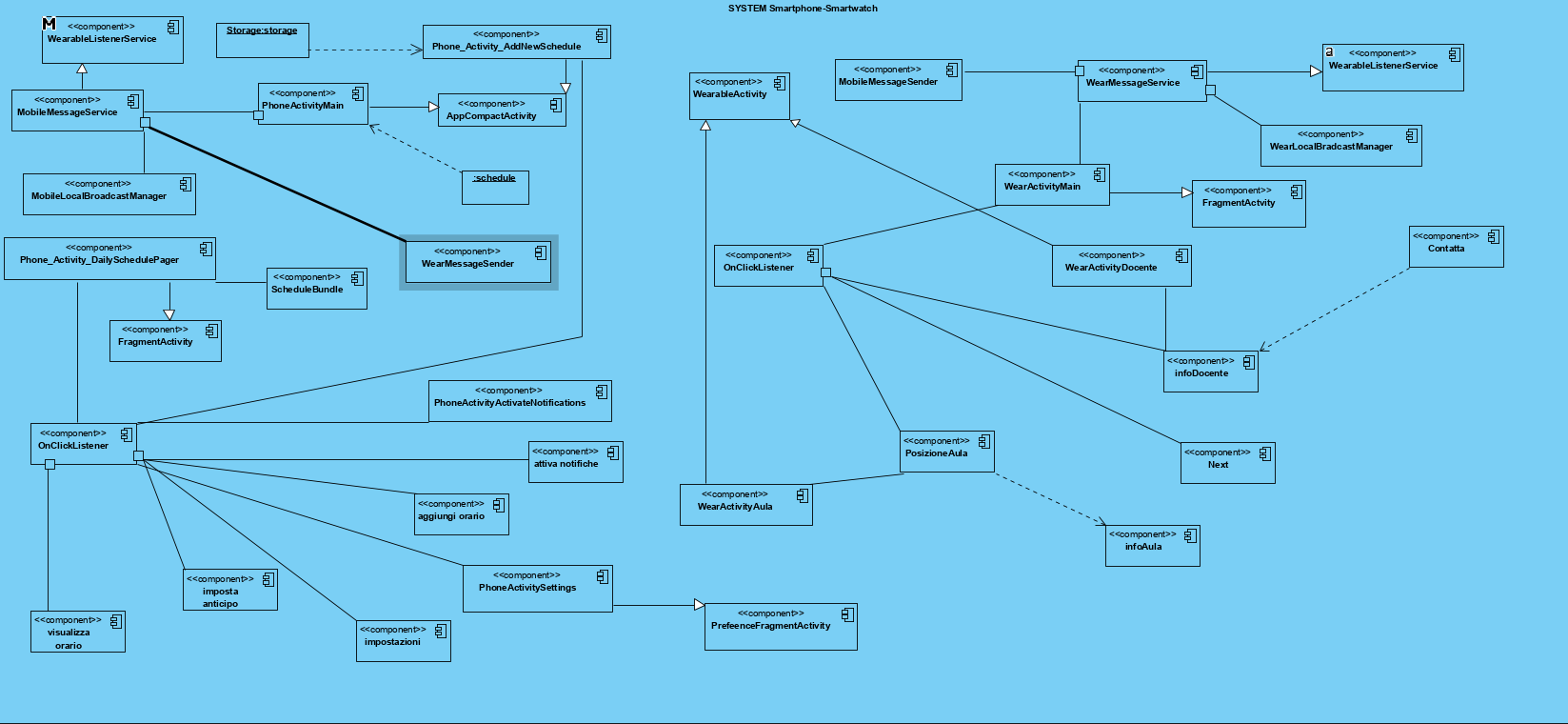
Un oggetto orario può contenere più lezioni e una lezione puo appartenere a più orari.

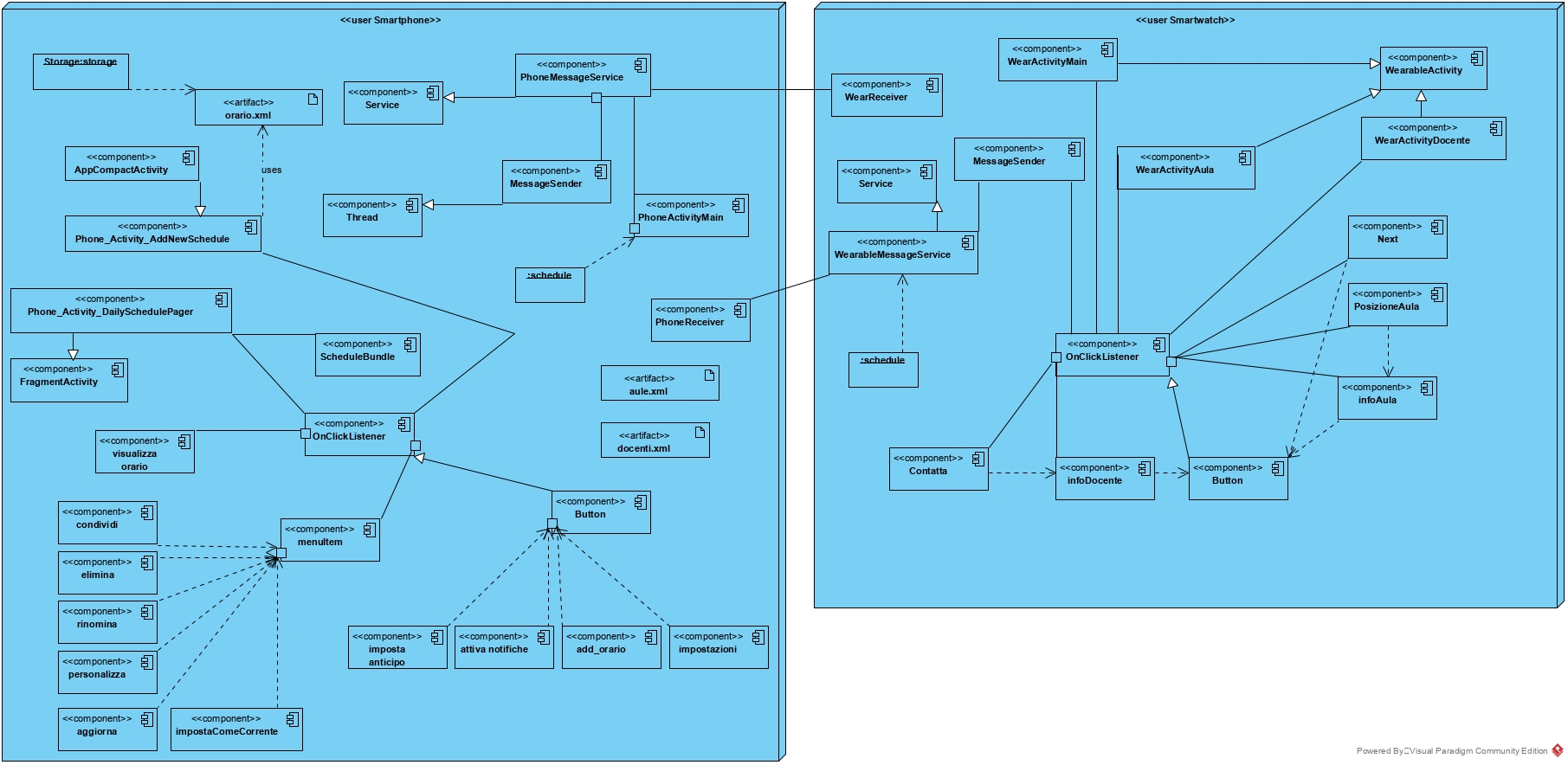
L’oggetto storage puo contenere più orari e un orario appartiene all’unico storage di sistema.

Di seguito è mostrato il diagramma delle classi:

Immagine che contiene screenshot

Descrizione generata automaticamente

Component Diagram: 

Deployment Diagram: 

**Validazione**

Immagine che contiene screenshot

Descrizione generata automaticamente

L’architettura non ha subìto drastici cambiamenti tranne che a livello di immagazzinamento ed elaborazione dei dati: prima delegata alla main Activity del modulo Phone e poi alla classe Schedule inserita nella libreria condivisa tra i moduli. Questa effettua il parsing dell’oggetto Orario dell’utente e ne crea la lista di Stub. Sia Stub che Schedule implementano l’interfaccia Parcelable, permettendo una facile trasmissione dei dati tramite Intent o Message.

Immagine che contiene screenshot, monitor, nero, sedendo

Descrizione generata automaticamente

Ciascun MessageService rileva cambiamenti nel proprio Data Layer e li inoltra tramite LocalBroadCaster al Thread Receiver corrispondente all’Intent contenuto nel Message. Il Receiver quindi elabora i possibili contenuti del messaggio, aggiornando gli oggetti Schedule e Attendance insiti nelle rispettive Main Activities. L’invio è affidato ai MessageSenderThread, estensioni della classe Thread, in modo tale da non occupare il Thread principale

Immagine che contiene screenshot, nero, monitor, schermo

Descrizione generata automaticamente

Dopo la rimodulazione del progetto, che ha previsto l’eliminazione del ViewPager, si è deciso di mantenere Wearable\_Fragment\_Stub di tipo Fragment in modo tale da non attribuire troppe funzioni alla main Activity.

La classe Wearable\_Activity\_Teacher permette all’utente di visualizzare informazioni base sui docenti e di contattarli tramite mail ufficiale, richiamando la propria applicazione di messaggistica di default.

L’Activity Wearable\_Activity\_Map indica tramite una piantina del politecnico la locazione precisa dell’aula in cui si tiene la lezione.

Immagine che contiene screenshot, sedendo, parcheggiato, schermo

Descrizione generata automaticamenteIl modulo Phone è simile al precedente, inoltre colleziona, aggiorna ed elabora i dati scaricati dal sito ufficiale del Poliba, rendendo disponibile la visualizzazione e la modifica degli orari dell’utente. Permette di visualizzare le statistiche e i grafici della propria presenza ai corsi.

Immagine che contiene screenshot, sedendo, parcheggiato, schermo

Descrizione generata automaticamente

**Conclusioni**

Nel complesso imparare ad operare con il framework di Android Wear è stato un compito molto arduo. Si è infatti trattato di un progetto complesso da realizzare sia a livello progettuale che a livello implementativo. Le maggiori difficoltà sono state riscontrate nella comprensione del funzionamento del meccanismo di comunicazione tra smartwatch e smartphone.

Al di là di queste difficoltà, il team si ritiene soddisfatto per i risultati ottenuti. Si è, infatti, avuta la possibilità di mettere in pratica le conoscenze possedute attraverso il corso di ingegneria del software in un progetto concreto, favorendo anche il miglioramento delle capacità di team working.

Obiettivo futuro è quello di sviluppare le feature descritte nella fase di analisi dei requisiti che non si ha avuto tempo e possibilità di implementare.

Primo fra tutti il meccanismo di parsing dell’orario che consentirebbe di prelevare dal sito ufficiale del Politecnico di Bari il file pdf selezionato dall’utente e fare il parsing del pdf in file .xml.

A tal proposito, le maggiori difficoltà sono state relative al difficile riconoscimento delle celle della tabella del file pdf; problematica che si è cercato di ovviare attraverso l’inserimento in fase di implementazione di un file .xml già scritto.

Un secondo problema è stata la difficile implementazione del meccanismo di gestione del registro presenze, per il quale si cercherà sicuramente una soluzione implementativa più ottimale in futuro.

**Bibliografia**

[1] *Visual Paradigm: Component diagram*

https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is- component-diagram/

[2] *Visual Paradigm: Deployment diagram*

https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-deployment-diagram/

[3] *UML: class diagram*

http://disi.unitn.it/~agiordani/ITI/TDP2.pdf

[4] *Visual Paradigm: class diagram*

https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-class-diagram/

[5] *Visual paradigm: sequence diagram*

https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/

[6] *HTML: sequence diagram*

https://www.html.it/pag/15669/i-componenti-dei-sequence-diagrams/

[7] *Casi d’uso*

http://www.federica.unina.it/ingegneria/ingegneria-del-software-ingegneria/casi-uso/

[8] *Visual Paradigm: use case diagram*

https://www.visual-paradigm.com/support/documents/vpuserguide/94/2575/6362\_drawinguseca.html

[9] *Ingegneria del software, Ian Somerville*

[10] *Viewpager:animation*

https://developer.android.com/training/animation/screen-slide

[11] *View pager*

https://developer.android.com/reference/android/support/v4/view/ViewPager

[12] *Communication handheld-wearable*

https://stackoverflow.com/questions/31035976/android-two-way-communication-between-handheld-and-wearable

[13] *Communication: data layer*

https://developer.android.com/training/wearables/data-layer

[14] *Layout: List view*

https://developer.android.com/reference/android/widget/ListView

[15] *Layout: fragments*

https://developer.android.com/guide/components/fragments

[16] *Layout: fragment adding*

https://stackoverflow.com/questions/9245408/best-practice-for-instantiating-a-new-android-fragment