Piano di progetto

Rev 1.1 (10/11/2016)

Progetto Crimson
Ingegneria del Software (A.A. 2016/17)

Gruppo MALG
Cavallo Mattia (845900@stud.unive.it)
Ceolin Andrea (846559@stud.unive.it)
Naccari Laura (846196@stud.unive.it)
Patti Giulio (847665@stud.unive.it)

Sommario

1. Introduzione

1.1. Overview del progetto

Nell'ambito dello sviluppo e della gestione di un progetto software per dispositivi mobili, oggetto del corso di Ingegneria del Software, il nostro gruppo (MALG) ha avanzato come proposta una "app" che permetterà all'utente finale di ottenere informazioni circa la sicurezza di determinate sostanze additive presenti negli alimenti e nei prodotti cosmetici, in particolare quelli acquistati in (o provenienti da) paesi extra-UE, facendo riferimento ai data-set pubblicati dall'Unione Europea relativi all'approvazione o all'applicazione di un divieto sull'utilizzo di questi ultimi.

A tal proposito, l'applicazione riceverà in input una sequenza di ingredienti, e per ognuno di essi verrà generato un responso che può essere positivo - nel caso la sostanza fosse stata esplicitamente approvata dall'UE per l'uso su prodotti destinati all'alimentazione e/o l'utilizzo umano, negativo se vi fosse stato applicato un qualsiasi divieto, o neutro se non vi fossero informazioni disponibili relative lo specifico ingrediente e/o se l'ingrediente in oggetto non rientrasse nella categoria degli "additivi".

1.1.1. Origine dei dati

Tutti o alcuni tra i seguenti *dataset* saranno periodicamente acquisiti e utilizzati localmente dall'applicazione al fine di operare il controllo:

- http://data.europa.eu/euodp/en/data/dataset/1gXgb0Yj73R4ttDChQ5Wyg
- https://data.europa.eu/euodp/it/data/dataset/cosmetic-database-list-of-restricted-substances
- http://data.europa.eu/euodp/en/data/dataset/pghvp0e6GbDsDYrIrlESYg

La lista dei dataset qui sopra potrebbe subire alterazioni o inserimenti in futuro, qualora venissero pubblicati ulteriori dati rilevanti all'ambito specifico dell'app prima della data di rilascio prevista.

1.2. Deliverables del progetto

Le seguenti scadenze sono state fissate per le varie consegne del progetto:

- **08/10/2016**: Proposta di progetto
- **21/10/2016**: Piano di progetto
- **11/11/2016**: Documento di analisi specifica
- **10/12/2016**: Documento di progettazione
- **20/12/2016**: Piano di testing
- **28/02/2017**: APK pronto alla pubblicazione sul *Google Play Store*.

1.3. Evoluzione del progetto

Lo sviluppo del progetto seguirà quanto più possibile la documentazione fornita, incluso il presente documento. Potranno tuttavia essere apportate modifiche ed adattamenti, di minore o maggiore rilevanza, qualora il team lo ritenga necessario.

1.4. Materiale di riferimento

Nella stesura dei vari documenti, si farà riferimento a:

- Slides del corso di Ingegneria del Software (2015-16, 2016-17), a cura del prof. Cortesi
- R. Pressman «*Principi di Ingegneria del Software*», McGraw-Hill, quinta edizione, 2008
- *Proposta di progetto*, gruppo MALG.

1.5. Definizioni e Abbreviazioni

Nella stesura della presente documentazione, si fa uso dei seguenti termini e abbreviazioni:

- **Ambiente di sviluppo**: Raccolta di strumenti destinati agli sviluppatori nell'ambito della produzione di progetti software.
- **Android**: Sistema operativo per dispositivi mobili, in particolare smartphone e tablet.
- **APK**: Pacchetto applicazione installabile su dispositivi con sistema operativo *Android*.
- **App**: Neologismo che sta per "applicazione (software)". Solitamente, con questo termine, si riferisce ad applicazioni aventi per target dispositivi mobili.
- **Backup**: Copia di sicurezza di dati informatici.
- **Bug**: Comportamento imprevisto di un applicazione software dovuto ad un errore (semantico) di programmazione da parte degli sviluppatori.
- **Cloud (computing)**: Insieme di tecnologie che permettono la condivisione e la fruizione di risorse, ad esempio spazio di archiviazione, agli utenti, attraverso Internet.
- **Codice sorgente**: Insieme di file(s) testuali scritti in un linguaggio di facile comprensione al programmatore che, una volta processati dal *compilatore*, genereranno un programma eseguibile (in linguaggio binario).
- **Compilatore**: Programma che si occupa di generare un file in linguaggio binario, eseguibile da parte della CPU di destinazione, a partire dal *codice sorgente* ricevuto in ingresso.
- **Controllo versione**: Sistema in grado di tenere traccia di ogni modifica effettuata ad un insieme di files e/o cartelle, e di rendere notevolmente più semplice la collaborazione tra più persone ad uno stesso progetto.
- **Database**: Archivio organizzato di dati, nel quale si definiscono relazioni, e si operano interrogazioni e manipolazioni.
- **Dataset**: Insieme di dati statici (o aggiornati periodicamente in toto), organizzati in maniera analoga ad un *database*.
- **DB**: Abbreviazione per *database*.
- **Deliverables**: Risorse e artefatti che verranno consegnati al committente del progetto.
- **Diagramma di Gantt**: Rappresentazione grafica delle attività necessarie al completamento di un particolare progetto, del tempo (in termini di data di inizio e di completamento) e delle risorse allocate per ognuna di esse.
- **Diagramma PERT (Project Evaluation and Review Technique)**: Rappresentazione grafica delle attività e delle relazioni di dipendenza tra esse, da cui si possono desumere informazioni utili per una determinazione più accurata dei probabili costi, anche in termini di tempo e risorse da dedicarvi.
- **Emulatore**: Particolare software in grado di simulare un'architettura hardware diversa da quella su cui è in esecuzione. Concettualmente simile alle *macchine virtuali*.
- **Git**: Implementazione di sistema di *controllo versione* ampiamente diffusa.
- **Google Play Store**: Negozio online con sezione dedicata alle *app* gestito da Google.

- **Grafica raster**: Immagine grafica memorizzata, bit per bit, esattamente nella forma in cui dovrà apparire sul visualizzatore dell'utente.
- **Grafica vettoriale**: Immagine grafica costruita dinamicamente a partire da entità geometriche definite. Ideale per la creazione di loghi e altre grafiche che potranno essere ridimensionate in ogni momento senza minima perdita di qualità.
- **HTTP(S)**: Protocollo di rete solitamente utilizzato per reperire pagine web da un server remoto, ma facilmente adattabile a situazioni in cui si desidera interagire con un servizio online per mezzo di richieste e risposte.
- **IDE (Integrated Development Environment)**: Termine inglese per l'*ambiente di sviluppo*.
- **Interfaccia utente**: Insieme degli elementi del programma con cui l'utente può interagire.
- **Java**: Linguaggio di programmazione orientato al *paradigma oggettivo*, utilizzato in particolare nella scrittura della parte algoritmica delle *app* con target *Android*.
- **JavaDoc**: Sistema che permette ai programmatori *Java* di incapsulare porzioni della documentazione tecnica direttamente all'interno del codice sorgente.
- **(Lato) Server**: Parte centralizzata dell'infrastruttura che accetta connessioni in ingresso e si occupa di soddisfare le richieste inoltrate. Si contrappone al **lato client**, che nel caso specifico di questo progetto è costituito dalle installazioni locali dell'*app* sui dispositivi degli utenti.
- **Libreria software**: Collezione di funzioni e strutture dati predisposte per essere incorporate o collegate in altro software.
- **Linguaggi di Markup**: Linguaggio utilizzato per la rappresentazione di dati strutturati.
- **Macchina virtuale**: Ambiente virtuale che emula tipicamente il comportamento di una macchina fisica, grazie all'assegnazione di risorse hardware (porzioni di disco rigido, RAM e risorse di processamento), ed in cui alcune applicazioni possono essere eseguite come se interagissero realmente con tale macchina.
- **Milestone**: Letteralmente "pietra miliare"; punto ben definito dello sviluppo di un progetto. Solitamente coincide con il completamento di un'attività cardine e/o il raggiungimento di una scadenza.
- **Open-Source**: Software che viene pubblicato non solo nella forma binaria (eseguibile compilato), ma anche corredato di codice sorgente e documentazione interna, cosicché possa essere rivisto, modificato e talvolta addirittura re-distribuito da altri.
- **Overview**: Riepilogo, sunto.
- **Paradigma oggettivo**: Modello di programmazione in cui vengono rappresentati gli oggetti della realtà di riferimento, e l'interazione (scambio di messaggi) che avviene tra questi.
- **Pattern (programmazione)**: Tecniche di programmazione ben collaudate volte ad ottimizzare l'efficienza e la semplicità del codice scritto.
- **Query**: Operazione effettuata su un dataset.
- **Repository**: Radice remota di un progetto gestito attraverso un sistema di *controllo versione*.
- **Script**: Tipo particolare di programma; solitamente caratterizzato da relativa semplicità, uso di linguaggio interpretato (ovvero che non richiede *compilazione*), mancanza di interfaccia, e spesso dedito a mansioni accessorie.
- **SDK (Software Development Kit)**: Dall'inglese "kit di sviluppo software", che solitamente si compone di *librerie*, *compilatori*, *emulatori* e talvolta anche di un *ambiente di sviluppo*.
- **Snapshot**: Immagine statica di un set di dati, ad esempio il disco rigido delle VM, acquisito in un determinato momento.

- **Telegram**: Piattaforma di messaggistica open-source basata sul cloud.
- **Template**: Termine inglese per modello.
- **UI (User interface)**: Termine inglese per l'*interfaccia utente.*
- **URL (Uniform Resource Locator)**: Sequenza di caratteri che identifica univocamente una risorsa in rete. L'URL è parte fondamentale di ogni richiesta effettuata attraverso *HTTP*.
- **User Experience (UX)**: Ciò che l'utente prova quando utilizza un prodotto o un servizio.
- **VirtualBox**: Uno dei molteplici software che si occupa di gestire *macchine virtuali*.
- **WBS (Work Breakdown Structure)**: Detta anche "struttura di scomposizione del lavoro" (traduzione letterale) o "struttura analitica di progetto", è l'elenco di tutte le attività di un progetto. Le WBS sono usate nella pratica della gestione del progetto e coadiuvano il project manager nell'organizzazione delle attività di cui è responsabile.
- **VM (Virtual Machine)**: Abbreviazione del termine inglese che sta per *macchina virtuale*.
- **Web-API**: Insieme di funzioni - elaborate a *lato server* - che vengono richiamate mediante semplici richieste *HTTP(S)*.
- **XML**: Particolare *linguaggio di markup*, con enfasi su dati organizzati in gerarchie.

2. Organizzazione del progetto

2.1. Modello di processo

Il gruppo si è espresso in maniera unanime a favore riguardo l'adozione di un modello di processo che non distingua in maniera rigida le fasi di prototipazione e di implementazione. Pertanto, la scelta è infine ricaduta sul **modello evolutivo** con prototipi ad uso interno.

2.2. Struttura organizzativa

Il gruppo incaricato (MALG) è attualmente composto da quattro (4) membri, i quali hanno deciso di mantenere un'organizzazione quanto più possibile democratica e decentralizzata.

2.3. Interfacce organizzative

Il team si relazionerà periodicamente con:

- **Prof. Agostino Cortesi**, al fine di ottenere opinioni e feedback sui progressi dello sviluppo;
- **Campione di utenti selezionati**, caratterizzato da diversi livelli di competenza nell'ambito di uso di tecnologie informatiche su dispositivi mobili, al fine di identificare eventuali problematiche nell'ambito dell'usabilità.

2.4. Responsabilità di progetto

Ogni membro del gruppo sarà tenuto a completare la parte di lavoro assegnatagli nei tempi previsti, e di notificare almeno due (2) degli altri membri qualora ciò non fosse possibile. In mancanza di un leader, l'assegnazione dei vari compiti avviene per mutua decisione durante i vari incontri tenuti tra i membri dal team stesso, secondo le modalità descritte al paragrafo 3.4.

3. Processi gestionali

3.1. Obiettivi e priorità

A partire dalla proposta, i seguenti obiettivi sono stati definiti:

- Rispetto dei termini e delle scadenze concordate con il docente.
- Realizzazione di un'app Android, installabile dagli utenti finali, che offra un responso positivo/negativo/neutro in base alla sequenza di ingredienti dati in input.

3.2. Assunzioni, dipendenze, vincoli

3.2.1. Assunzioni

Si assume che:

- Gli utenti dispongano di un dispositivo mobile basato su Android 4.2 (o versione successiva);
- Gli utenti vivano in un paese in cui il servizio *Google Play Store* viene fruito regolarmente;
- Il team disponga delle risorse enumerate al paragrafo 5.3.

3.2.2. Dipendenze

Il completamento del progetto e la ricchezza di funzionalità dipenderà dal tempo che ogni membro del gruppo riuscirà a dedicare al progetto, dal livello di competenza che egli o ella riuscirà a maturare durante l'apprendimento, e da quanto efficientemente saranno suddivisi i vari compiti.

3.2.3. Vincoli

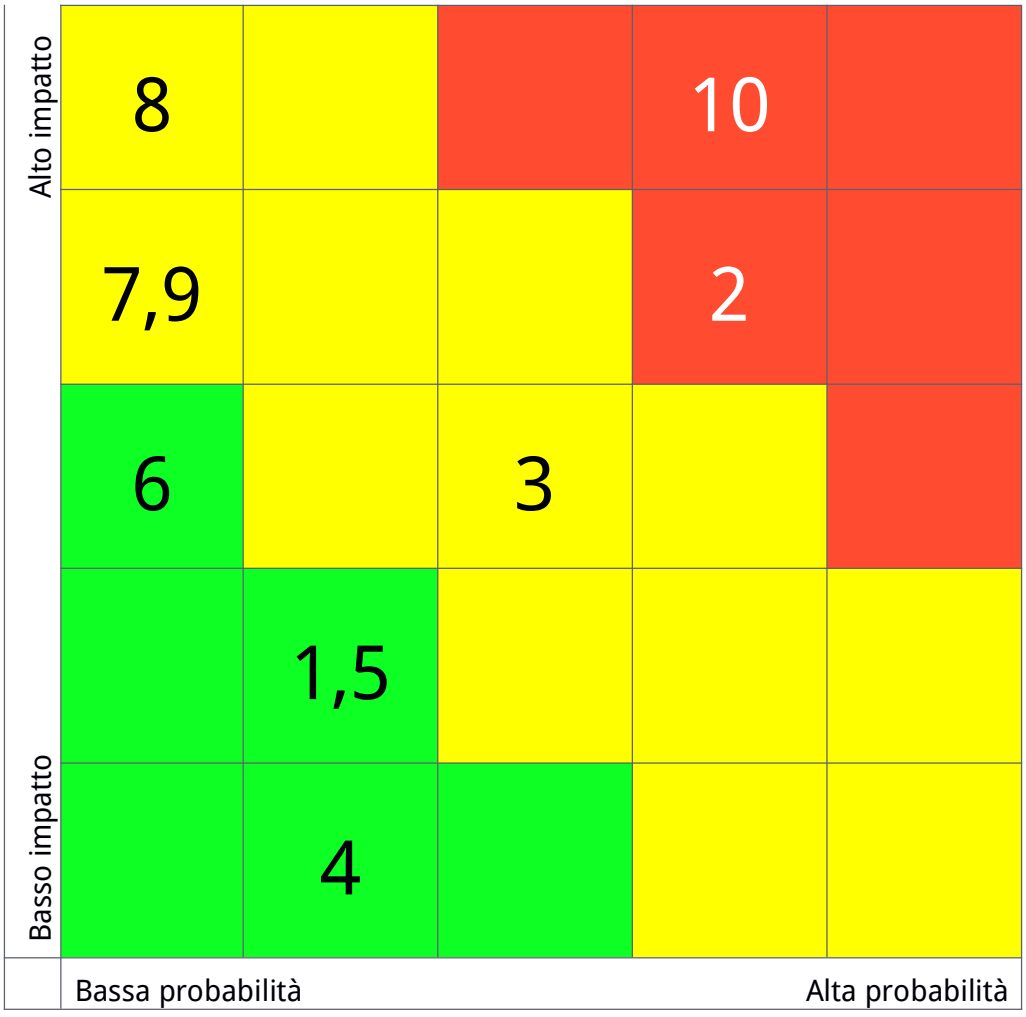
- **Temporali**: Dettati dalle scadenze stabilite al paragrafo 1.2.
- **Qualitativi**: L'app dovrà rispettare le linee guida sull'usabilità dettate dalle euristiche di Nielsen (vedi http://www.far.unito.it/usabilita/Cap5.htm), ed almeno il 50% degli utenti che ne testeranno la versione pre-rilascio dovranno essere in grado di compiere almeno le operazioni elementari in totale autonomia (inserimento di un ingrediente e capire il responso dell'applicazione).
- **Progettuali**: I vari obiettivi (paragrafo 3.1) devono essere raggiunti prima della pubblicazione.

3.3. Gestione dei rischi

I seguenti rischi sono stati individuati:

ID	Rischio	Categoria*	Probabilità	Impatto
1	Necessità di librerie software aggiuntive	DE	25%	2
2	Difficoltà o bug riscontrati dagli utenti	CU	75%	4
3	Problemi di visibilità nel mercato	BU	50%	3
4	Problemi con la VM di sviluppo	EV	25%	1
5	Indisponibilità di alcuni membri del team	ST	25%	2
6	Incomprensioni tra membri del team	ST	5%	3
7	Perdita di dati o codice	EV	5%	4
8	Tempo insufficiente per il completamento	PS	5%	5
9	Competenze insufficienti dello staff	ST	12%	4
10	Dataset obsoleti	CU	80%	5

* DE= Development (sviluppo), EV= Environment (ambiente di lavoro), ST=Staff, PS= Product Size (dimensione del progetto), CU= Customers base (utenti/clienti), BU= Business impact.



Dunque, le seguenti misure verranno messe in atto per risolverli:

- **Rischio 1**: Ripiegare su librerie di terze parti
- **Rischi 1, 4, 8, 9**: Analisi approfondita
- **Rischio 2**: Test e sessioni di debug al completamento di ogni micro e macro funzionalità, servizio di supporto e manutenzione post-rilascio.
- **Rischio 3**: Possibile campagna di marketing post-rilascio
- **Rischio 4**: Sistema interno di sincronizzazione ed aggiornamento, al fine di uniformare le impostazioni e il software della VM tra i membri del gruppo. Creazione snapshot di ripristino e backup di emergenza del disco fisso virtuale sul cloud.
- **Rischi 5, 6, 8**: Incontri frequenti tra i membri del team.
- **Rischio 6**: Si prediligerà la comunicazione testuale e ci si atterrà sempre alla documentazione.
- **Rischio 7**: Utilizzo di un sistema di controllo versione per il codice, la documentazione ed i backup della repository.
- **Rischio 9**: Sessioni di apprendimento collettive durante la fase preliminare del progetto.
- **Rischio 10**: Ripiegare su eventuali altri dataset in futuro.

3.4. Meccanismi di monitoraggio e controllo

Lo staff si incontrerà pressappoco ogni una o due settimane, virtualmente e talvolta fisicamente, in maniera compatibile con gli eventuali impegni che possono sorgere. Durante questi incontri verrà monitorato l'avanzamento del progetto, e in caso di rallentamenti accertati, si deciderà mutualmente come agire.

Il materiale di lavoro verrà condiviso tra i membri dello staff mediante una repository GIT privata, la quale fungerà anche da sistema di backup.

Il gruppo potrà comunicare testualmente e/o verbalmente per mezzo del software open-source Telegram.

3.5. Pianificazione dello staff

I membri dello staff non saranno assegnati in maniera rigida ai vari sotto-progetti; l'allocazione del personale potrà variare a seconda delle necessità, previo mutuo accordo durante i vari incontri. L'unico vincolo sarà quello imposto dalle competenze dei singoli membri del team.

Le conoscenze necessarie per lo sviluppo dell'app (parte algoritmica) sono, in ordine decrescente di importanza:

- Linguaggio di programmazione Java
- Librerie Android SDK (documentazione online disponibile)
- Paradigma oggettivo e pattern più diffusi
- Conoscenza del protocollo HTTP(S) e fondamenti di XML, CSV e JSON.
- Sistema di controllo versione "git"
- Ambiente di sviluppo AndroidStudio

Mentre, quelle per lo sviluppo dell'interfaccia grafica dell'app:

- Linguaggio di markup XML
- Linee guida del design di interfacce per app Android (documentazione online disponibile)
- Uso degli strumenti di disegno di AndroidStudio
- Sistema di controllo versione "git"

- Uso di software grafico raster (GIMP, PhotoShop, ecc.) e/o vettoriale (InkScape, Illustrator, ecc.)

4. Processi tecnici

4.1. Metodi, Strumenti, Tecniche

Al fine di uniformare l'ambiente di sviluppo utilizzato da ciascun membro del team, è stata predisposta una macchina virtuale (in formato VirtualBox), basata su Ubuntu 14.04 LTS, con i seguenti pacchetti preinstallati:

- IDE AndroidStudio (per lo sviluppo e il design dell'app)
- Android SDK (compilatori ed emulatori - target Android 4.2, livello API 16).
- Telegram (per la comunicazione tra i membri del team)
- Repository dedicata (per la collaborazione alla stesura del codice, della documentazione, altri files e/o backup di codice e risorse)
- WPS Office, per la redazione della documentazione.
- Script di auto-configurazione ed aggiornamento (per mantenere la configurazione della VM sincronizzata con quella utilizzata dal resto del gruppo)

4.2. Documentazione del Software

La documentazione di progetto verrà pubblicata sulla wiki del gruppo secondo le date stabilite al paragrafo 1.2. La restante parte di documentazione verrà pubblicata in concomitanza con il rilascio finale, e sarà comprensiva di:

- Manuale utente
- Documentazione del codice dell'app (da JavaDoc)

La documentazione verrà prodotta utilizzando i seguenti strumenti, e avverrà a pari passo con lo sviluppo delle componenti software del progetto:

- Doxygen (o alternative) per la generazione della documentazione relativa al codice
- Microsoft Project e Microsoft Visio (o alternative) per la progettazione dei diagrammi
- WPS Office (o alternative) per la stesura della documentazione progettuale
- mkdocs (o alternative) per la stesura del manuale utente.

4.3. Funzionalità di supporto al progetto

Verranno effettuati molteplici test manuali durante lo sviluppo dell'app;

idealmente almeno uno al completamento di ogni micro o macro funzionalità, questo per verificare che anche la user-experience rispetti i criteri previsti, cosa che sarebbe difficile da

attuare per mezzo dei soli test automatici.

Contemporaneamente allo sviluppo delle componenti e delle sotto-componenti del progetto,

verranno preparati dei test automatici finalizzati a verificare il corretto funzionamento di queste ultime (o almeno di quelle principali, compatibilmente con le scadenze ed il tempo a

disposizione), utilizzando l'AndroidTestingFramework nell ' ambito dell ' app. I suddetti test saranno eseguiti inizialmente in maniera isolata (ovvero testando una sola componente alla volta ed emulando le altre), fintanto che non viene ultimato lo sviluppo di tutte queste, e dunque in maniera integrata, partendo dalle singole sotto-componenti e finendo con il test dell ' intera infrastruttura (metodica bottom-up).

Il gruppo revisionerà periodicamente tutto il lavoro, al fine di assicurarsi che il codice, ed in generale tutto il materiale prodotto, rispettino i vincoli definiti al paragrafo 3.2.3, e che la presente documentazione non venga contraddetta.

5. Pianificazione lavoro, risorse umane e budget

5.1. Work Breakdown Structure (WBS)

Le attività di progettazione sono state organizzate come segue. In futuro, la seguente struttura potrebbe venire ampliata con informazioni più dettagliate:

1. Produzione documentazione progettuale
    1.1. Proposta
    1.2. Piano di Progetto
    1.3. Analisi Specifica
    1.4. Documento di testing
    1.5. Documento Progettaz.
2. Project Management
3. Preparazione ambiente di sviluppo condiviso
4. Apprendimento
5. Prototipazione
    5.1. Interfaccia (UI)
    5.2. Diagramma classi App
    5.3. Diagramma classi Server Dataset
    5.4. API di interconnessione App/Open Data
6. Produzione
    6.1. Codice App
    6.2. Codice UI
    6.3. Codice query sui Dataset
7. Produzione documentazione di accompagnamento
    7.1. Manuale utente
    7.2. Documentazione interna
8. Controllo qualità / Test
9. Collaudo
10. Rilascio e pubblicazione

5.2. Dipendenze

Si faccia riferimento ai diagrammi di Gantt e PERT in calice al documento.

5.3. Risorse necessarie

Tutti gli strumenti software (compilatori, emulatori, ambienti di sviluppo) necessari allo sviluppo del progetto, ad eccezione di VirtualBox, sono stati installati in un'apposita macchina virtuale, distribuita ad ogni membro del gruppo. Per informazioni dettagliate sul software incluso, si faccia riferimento al paragrafo 4.1.

Si richiede inoltre che ogni membro del gruppo abbia a disposizione un PC dotato di CPU compatibile con l'architettura x86-64, che supporti le tecnologie di virtualizzazione assistita mediante hardware (Intel VT-x o AMD-V), e che disponga di accesso ad Internet durante tutte le fasi dello sviluppo.

5.4. Allocazione del budget e delle risorse

Il costo in termini finanziari del progetto sarà pressoché nullo, dal momento che si tenterà di fare uso - nei limiti del possibile - di software libero e di infrastrutture che ci vengono messe a disposizione dall'ateneo Ca'Foscari.

Qualora lo sviluppo e/o il mantenimento del progetto proseguissero al di fuori dell'ambito accademico, si stima che dovranno essere allocati non meno di 50.00€/annuali per le spese di hosting (sulla base del prezzo medio nazionale relativo ai servizi richiesti), e circa 25.00€ una tantum per la registrazione di un account *developer* presso Google Play. A ciò si dovranno aggiungere le eventuali spese burocratiche, di costituzione societaria e salariali.

Il personale tuttavia, essendo piuttosto esiguo (quattro individui), è un elemento critico. Motivo per cui, come accennato al paragrafo 2.4, è fondamentale che in caso di malattia o indisponibilità di uno qualsiasi dei membri, gli altri vengano tempestivamente notificati, al fine di attuare le dovute compensazioni.

5.5. Pianificazione

Al fine di favorire l'alternanza fra prototipazione ed implementazione, e di parallelizzare lo sviluppo delle varie componenti del progetto, si è deciso di non fissare scadenze o *milestones* per le fasi successive a quelle di apprendimento e stesura della documentazione di progetto e precedenti a quella di collaudo finale.

Dunque, le uniche scadenze/milestones previste sono quelle coincidenti alle date di consegna espresse al paragrafo 1.2, e la data di inizio del collaudo finale (20/02/2017), entro e non oltre la quale si presuppone che l'intera fase di produzione venga completata.

Il rilascio, anch'esso, non può avvenire oltre la data prestabilita (28/02/2017).

Appendice A: *Diagramma di Gantt*

Si veda la risorsa *PDP_Gantt.pdf* consegnata assieme al presente documento.

Appendice B: *Diagramma PERT*

Si veda la risorsa *PDP_PERT.pdf* consegnata assieme al presente documento.

Appendice C: *Elenco dei cambiamenti*

Segue l'elenco delle modifiche apportate al presente documento, revisione per revisione.

Rev 1.1 - 10/11/2016

- Correzione delle scadenze fissate per i deliverables, in relazione alle disposizioni date.
- Correzioni cosmetiche sul frontespizio.

Rev 1.0 - 22/10/2016

Stesura iniziale del piano.