ALMA MATER STUDIORUM - UNIVERSITA' DI BOLOGNA CAMPUS DI CESENA

DIPARTIMENTO DI INFORMATICA – SCIENZA E INGEGNERIA

CORSO DI LAUREA IN INGEGNERIA E SCIENZE INFORMATICHE

TITOLO DELL'ELABORATO:

UN SOFTWARE COLLABORATIVO PER LA GESTIONE DELLE ATTIVITA' DI UN TEAM

Elaborato in

INGEGNERIA DEL SOFTWARE

Relatore

STEFANO RIZZI

Presentata da

ALESSANDRO SABATINO

Anno Accademico 2018/2019

Prefazione

L’idea su cui la presente tesi si basa, nasce da un’esigenza reale di un gruppo di lavoro composto da collaboratori del sito web Koinervetti (www.koinervetti.com).

Koinervetti nasce nel 2015 come progetto interno dell’Istituto d’Istruzione Superiore Franchetti-Salviani di Città di Castello (PG) per poi evolversi negli anni fino a “privatizzarsi” (con il sottoscritto come intestatario del dominio) in seguito alla formazione di un team eterogeneo ed autonomo composto da studenti delle più svariate facoltà ed Università d’Italia. Un’unione di intenti frutto di caparbietà e fiducia nella crescita del gruppo. Un collettivo di studenti intraprendenti, un viaggio ambizioso e stimolante, indirizzato originariamente dalla fiducia di due insegnanti professionisti, promotori del progresso dell’apprendimento attraverso esperienze di tutti i tipi. Esperienze orientate al cambiamento e all’evoluzione, sinonimo di crescita.

Figura 1 - logo del sito web Koinervetti

Koinervetti è un insieme di spazi creativi assegnati a degli autori dove gli stessi possono esprimere il proprio talento influenzato

da una qualsivoglia inclinazione artistica o interesse. Uno spazio creativo è, in sostanza, una rubrica che, se Koinervetti corrispondesse ad un palazzo, potrebbe equivalere ad una casa in affitto. Ad attività “in singolo” come la gestione di uno spazio personale si evidenziano anche attività di gruppo nelle quali il collettivo ha il piacere di raccontare storie attraverso la voce dei protagonisti (locali e personalità di rilievo) e l’organizzazione di eventi (massima espressione del lavoro di squadra) in collaborazione con associazioni o altri privati.

Il credo del team è quindi la collaborazione e la condivisione:

«Se tu hai una mela e io ho una mela e ci scambiamo le nostre mele, allora tu ed io avremo ancora una mela a testa. Ma se tu hai un’idea e io ho un’idea e ci scambiamo queste idee; allora ciascuno di noi avrà due idee»

George Bernard Shaw

Indice

1 Obiettivo del progetto

La necessità di collaborare si riversa nella concreta gestione ed amministrazione del sito web attraverso la formazione di un organigramma funzionale, di flussi di lavoro e di pianificazione delle attività. Motivi per cui, in un contesto del genere, il software diventa “groupware”: un applicativo che consente lo svolgimento di un lavoro di tipo collaborativo con processi che regolano la cooperazione.

L’obiettivo equivale perciò allo sviluppare un software collaborativo che faciliti l’interazione fra i collaboratori e lo svolgimento delle mansioni loro assegnate, caratterizzanti delle responsabilità in termini funzionali. Un “groupware” che racchiuda la gestione della conoscenza, la condivisione delle informazioni utili nel contesto cooperativo del sito web Koinervetti ed il flusso di attività (da analizzare sotto differenti punti di vista) e documenti vagliati dall’amministratore costituenti un workflow system.

Nei capitoli successivi si sottolineeranno le caratteristiche del dominio applicativo (Secondo Capitolo) con conseguente specifica dei requisiti ed analisi (Terzo Capitolo), per poi approcciarsi alla progettazione (Quarto Capitolo). Prima dell’implementazione (Sesto Capitolo) si affronterà la questione tecnologie utilizzate (Quinto Capitolo). Il settimo ed ultimo capitolo sarà dedicato alle conclusioni (sezione dedicata anche a possibili migliorie ed aggiornamenti futuri).

2 Descrizione del dominio applicativo

In virtù delle rimarcate considerazioni, si realizza un groupware basato sulla condivisione di informazioni fra collaboratori e sulla gestione di processi caratterizzanti le attività del gruppo di lavoro.

La gestione del sito web da parte del team avviene in modo cooperativo. Tuttavia, il coordinatore (Admin) supervisiona le attività definendo processi ed attori coinvolti negli stessi, con l'obiettivo di raggiungere un risultato soddisfacente in modo efficiente.

Il raggiungimento del risultato dipende fortemente dalle risorse a disposizione:

- *Risorse umane*: insieme dei collaboratori ai quali sono assegnati dei ruoli (utili alla costituzione dell'organigramma). Alcuni collaboratori sono identificati da ruoli funzionali (attualmente riguardanti "come", "quando" e "se" pubblicare un contenuto. Rispettivamente si tratta di prendere decisioni sulla modalità di pubblicazione nel sito web e nei social network e sulla presentazione del contenuto una volta online, sulla pianificazione e sul soddisfacimento dei requisiti di qualità per decretare "idoneo" o meno il contenuto stesso revisionato).

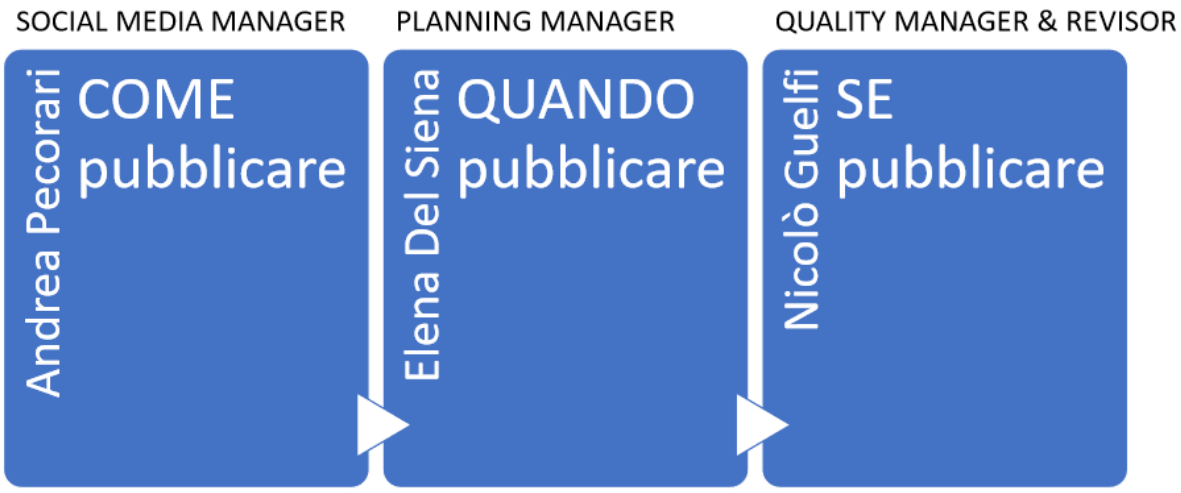

Figura 2 - Organigramma funzionale

- *Risorse hardware*: il servizio hosting del website dispone di un server con database MySQL dedicato di 2GB su SSD, caselle mail illimitate (1GB ciascuna, estendibili a 25GB in numero limitato).
- *Risorse software*: l'amministrazione è attuata tramite WordPress (Content Management System). Vengono effettuati inoltre dei backup (giornalieri) e si può usufruire della modalità "staging" che consente di clonare in un ambiente di prova il sito web, onde evitare di compromettere la versione pubblica in caso di modifiche. Si denota anche l'utilizzo di File Manager come FileZilla, software per il controllo delle statistiche, web mail (oltre a business mail e PEC) e classiche applicazioni mobili per la comunicazione (vedi WhatsApp e Skype).

  Considerando le risorse sopra elencate, i processi attuati con una certa ricorrenza riguardano:

- *Ricerca dei contenuti e scrittura della bozza*: l'autore (ruolo predefinito), una volta scelto l'argomento da trattare, ricerca appropriati contenuti ponendo particolare attenzione alle fonti dalle quali gli stessi vengono reperiti, avendo la necessità di citarle. In seguito, una volta caricata la bozza nel CMS questa sarà revisionata e salvata, pronta per essere "corretta" dal Quality Manager.
- *Pubblicazione contenuti*: il Quality Manager nonché Revisor, a bozza salvata, viene notificato dall'autore per poi procedere alla revisione del futuro articolo. Una volta terminata la revisione (la quale potrà sancire la non idoneità del contenuto, caso in cui l'autore dovrà effettuare le opportune modifiche) viene notificato l'Admin che, successivamente ad un controllo finale, procederà alla pubblicazione in koinervetti.com. La "catena" di notifiche termina con la comunicazione al Social Media Manager, responsabile della pubblicazione nei social network. Contemporaneamente alla revisione, il Planner Manager pianificherà la pubblicazione in sito web e social assegnandole uno slot libero della settimana (pianificazione che potrebbe subire delle variazioni nel caso la revisione sancisca la non idoneità).
- *Gestione delle collaborazioni*: in caso di volontà nell'instaurare rapporti di collaborazione con altri enti, associazioni o privati, l'autore proponente (o, più comunemente, l'Admin) contatta la "terza parte" con una proposta. In seguito allo studio "congiunto" di un piano di collaborazione si proseguirà con la presentazione di un documento contenente la formalizzazione della proposta di collaborazione da parte dell'Admin. In caso di controproposta questa verrà valutata con eventuale conseguente definizione.

- *Gestione delle prestazioni*: caso diametralmente opposto. Nell'eventualità che un esterno al team invii delle proposte, si procederà allo studio in merito alla fattibilità della "prestazione" ed all'opzionale invio di una controproposta. Il flusso termina con definizione di un accordo (potenzialmente equivalente ad un "nulla di fatto").
- *Gestione eventi e progetti interni*: escluse le fasi di realizzazione (non standardizzate) la gestione di un evento corrisponde a quella di una collaborazione. La differenza sostanziale riguarda la nomina di un responsabile che curerà la sua realizzazione.
- *Riunioni*: gli incontri (dalla cadenza non regolare) si basano sulla definizione di un documento (Ordine Della Settimana) presentato nella prima fase. Avrà poi luogo la discussione degli argomenti proposti dall'Admin e la redazione di un verbale associato.

  Buona parte dei documenti gestiti presentano una struttura predefinita. L'Ordine Del Giorno, per esempio, è caratterizzato dalla presenza di una sezione principale con le questioni (di varia natura) più urgenti da trattare e da altre sezioni opzionali (proposte di miglioramento e lo "stato dell'arte" di collaborazioni ed eventi) susseguite dalla presentazione dell'analisi degli insights di sito web e social.

  I documenti proponenti una collaborazione definiscono invece l'idea alla base della proposta e l'obiettivo, mentre, se si tratta di un progetto interno o dell'organizzazione di un evento, oltre a idea di base ed obiettivo è previsto specificare risorse necessarie, tempistiche e ruoli di responsabilità.

  Ogni attività propria dei processi elencati, necessita una programmazione settimanale. Esistono attività di vario genere:

alcune si ripetono ogni settimana nel calendario (soprannominate perciò “fisse”), altre non hanno cadenza regolare (“variabili”). Eventuali investimenti (relativi a pubblicità, per esempio) ricoprono il tipico ruolo di “attività variabile”. Lo svolgimento di attività (sia fisse che variabili) comporta l’impiego di risorse (umane e non) e l’investimento di costi e tempi, motivo per il quale le stesse sono classificate in base ad un livello di criticità.

3 Specifica dei requisiti ed analisi

L’obiettivo che si desidera raggiungere è fornire una serie di servizi atti alla condivisione della conoscenza fra i collaboratori del team ed alla pianificazione delle attività.

Nel contesto del cooperative working, una delle problematiche chiave caratterizzanti il lavoro del team è la pianificazione delle attività (con annessa un’attenzione particolare all’impiego di risorse trasversalmente incidenti in termini di costi e tempi). Il collettivo in questione fa riferimento, per la precisione, alla programmazione di contenuti da pubblicare, investimenti da compiere ed all’organizzazione e partecipazione ad eventi e meeting.

Allo stesso tempo, la collaborazione rende necessaria la condivisione, amministrata attraverso dei processi di gestione della conoscenza corrispondenti ad un insieme e metodi.

Il progetto richiede perciò realizzazione e gestione di:

- *Workflow collaborativi* come automazione parziale di processi ricorrenti e caratterizzanti il gruppo di lavoro, nei quali documenti ed informazioni vengono condivisi fra i collaboratori (tipicamente identificati da ruoli funzionali) per svolgere attività regolamentate ed in grado di velocizzare il processo di “apprendimento” tramite la collaborazione.
- *Gestione documentale* come garanzia della corretta produzione e conservazione dei documenti, con l’obiettivo di rendere gli stessi inalterabili ed in ogni caso reperibili.
- *Sistema di registrazione delle competenze individuali e collettive* dei collaboratori per la definizione della struttura organizzativa tramite organigramma e di gruppi di lavoro in base ai ruoli.

Durante la fase corrente si ha lo scopo di mantenere un certo tasso di qualità attraverso *chiarezza*, *consistenza* e *non ambiguità* delle specifiche.

E’ per questo necessario indicare chiaramente operazioni e soggetti descritti dal processo, il quale, a sua volta, dovrà essere definito da specifiche non contraddittorie, in modo completo e dettagliato.

I requisiti funzionali si differenziano in base agli stakeholder del sistema. I cosiddetti “portatori di interesse” sono solitamente dei soggetti influenti nei confronti di società o progetti. Nel caso analizzato, si tratta dell’insieme di collaboratori facenti parte del gruppo di lavoro del sito web (definibile quindi un “gruppo di interesse locale”). I collaboratori, a loro volta, si dividono in due categorie:

- *Stakeholder forti*: utenti aventi un ruolo funzionale e di responsabilità, senza i quali il team non funzionerebbe. Social Media Manager (responsabile del “come” pubblicare i contenuti), Planner Manager (responsabile del “quando” pubblicare) e Quality Manager & Revisor (responsabile del “se” pubblicare). Sono inoltre definibili come “forti” anche gli stakeholder “di progetto”, ovvero quei collaboratori che generalmente non sono presenti nell’organigramma (non avendo uno dei tre ruoli funzionali sopra indicati) ma che nel contesto di una specifica attività possono essere investiti di una responsabilità se ritenuti idonei.
- *Stakeholder deboli*: utenti corrispondenti agli autori degli spazi a disposizione nel sito web e che entrano in contatto con il gruppo saltuariamente. Ogni collaboratore definibile come tale è uno stakeholder debole. In caso di assegnazione di un progetto uno stakeholder di questo tipo può diventare “forte” e ritornare “debole” alla fine dell’attività verso la quale aveva delle responsabilità.

Tutti collaboratori (di ambedue le tipologie di stakeholder) avranno accesso al sistema di workflow per la “condivisione della conoscenza” ma la modifica dei processi sarà prerogativa dei soli utenti con ruoli funzionali, nel rispettivo raggio di azione.

Ogni utente avrà la possibilità di:

- Consultare documenti e calendario delle attività per informazioni circa la pianificazione
- Consultare i workflow
- Effettuare proposte relative a progetti o attività di vario genere
- Consultare proposte di altri collaboratori

I collaboratori facenti parte dell'organigramma funzionale (stakeholder forti) avranno ulteriori privilegi in quanto:

- Il *Planner Manager* avrà la possibilità di pianificare le attività (e quindi apporre modifiche al calendario)
- Il *Quality Manager & Revisor* potrà modificare i workflow di revisione e pubblicazione
- Il *Social Media Manager* sarà in grado di modificare il workflow di pubblicazione e definire strategie per i social network (sempre attraverso dei workflow).

L'amministratore e coordinatore del team e del groupware dovrà occuparsi di:

- *Gestione documentale* (creando proposte, Ordini "Della Settimana" ed inviando mail per ogni genere di comunicazione da recapitare ai collaboratori)
- *Gestione dei flussi di lavoro* (costituendo workflow collaborativi corrispondenti ad ogni processo ed attività caratterizzanti il gruppo di lavoro e definendo i necessari organigrammi)
- *Pianificazione degli investimenti e delle attività* (programmando ed analizzando costi e tempi e risorse da investire per la creazione e pubblicazione di contenuti, piuttosto che per

l’organizzazione di eventi o l’attuazione di progetti, affiancato dal *Planner Manager*).

I requisiti non funzionali riguardano invece l’*usabilità* (facilità d’uso), definita dall’ISO come “l’efficacia, l’efficienza e la soddisfazione con la quale determinati utenti raggiungono determinati obiettivi in determinati contesti”. In questo dominio applicativo la definizione è traducibile con la “necessità di progettare un’interfaccia semplice che sia fruibile ai collaboratori per effettuare facili operazioni e dalla buona operabilità”.

3. 1 Modellazione

Le specifiche vengono modellate per mezzo del “linguaggio di modellazione unificato” UML, il quale dispone di una notazione standard basata su un metamodello (in quanto rende possibile la definizione di un diagramma attraverso un altro). La notazione viene espressa tramite diagrammi grafici (particolari visualizzazioni di alcuni tipi di elementi di un modello come entità e relazioni). I diagrammi utilizzati sono:

- *Activity Diagram:* diagramma che modellano un processo come un’attività (associabile a qualunque elemento) nonché insieme di nodi connessi da archi.
- *Sequence Diagram:* diagramma di interazione che mostra le relazioni fra le cosiddette “linee di vita” degli elementi come sequenza di messaggi ordinati temporalmente.
- *State Diagram:* esaustivo descrittore dell’evoluzione temporale delle istanze di un classificatore. Sostanzialmente descrive il cambiamento di stato di un oggetto.
- *Collaboration Diagram:* diagramma che mostra l’interazione fra le linee di vita senza enfatizzare il tempo.
- *Use Case Diagram*: descrittore dell’interazione che gli attori hanno con il sistema.
- *Class Diagram*: diagramma fondamentale che descrive la struttura statica del sistema raffiguranti classi e relazioni fra le stesse.
- *Component Diagram*: descrittore delle componenti software del sistema e delle loro relazioni.
- *Deployment Diagram*: diagramma in grado di specificare l’hardware sui cui verrà eseguito il software e la modalità di dislocamento dello stesso nell’hardware.

Di seguito sono riportati quelli maggiormente significativi:

- *Activity Diagram (gestione pubblicazioni)*

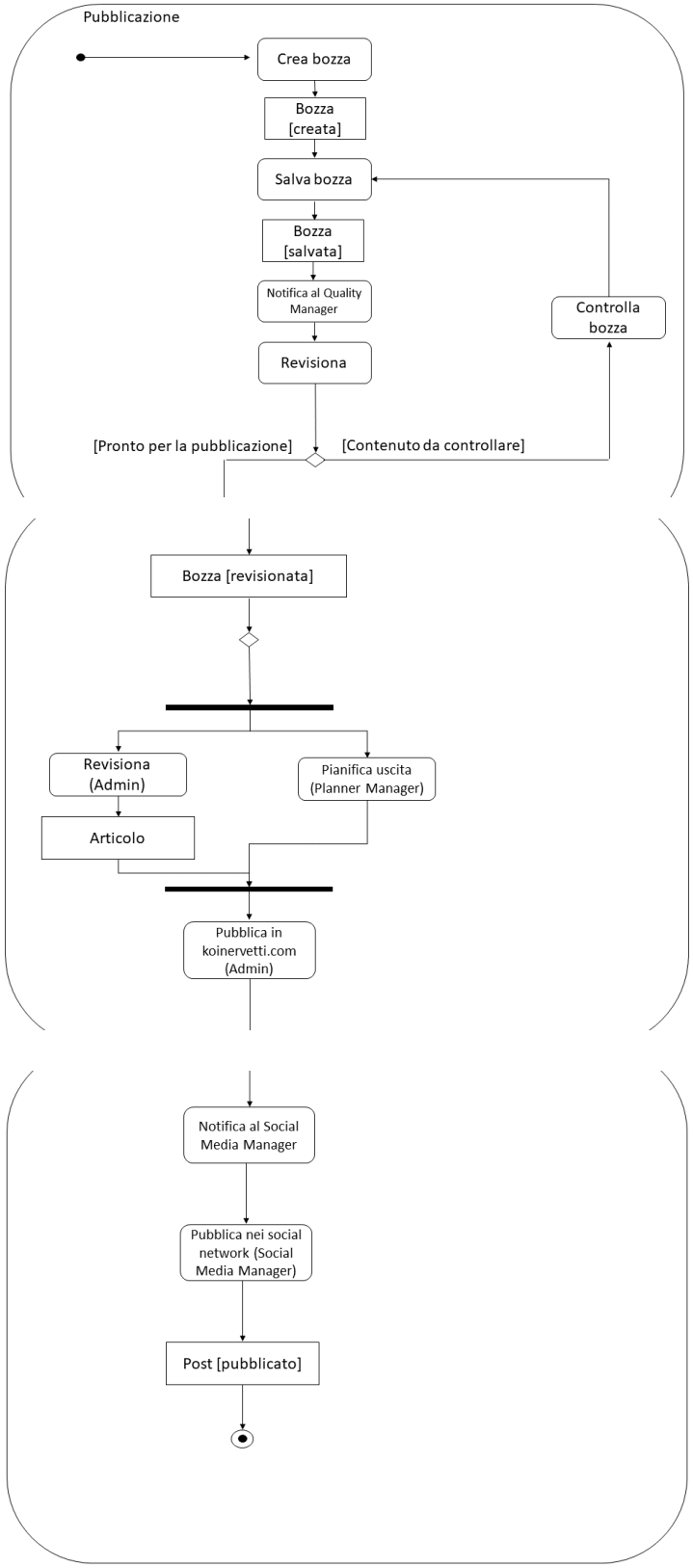

Figura 3.1.1 - Activity diagram del processo di pubblicazione di contenuti

- *Sequence Diagram* (gestione pubblicazioni):

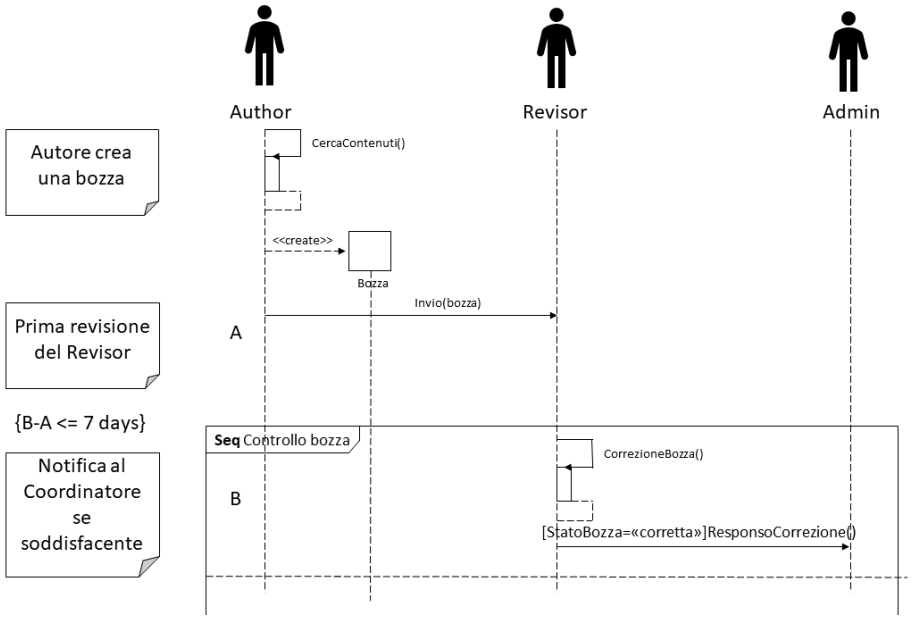

Figura 3.1.2 - Sequence diagram del processo di pubblicazione dei contenuti (parte 1)

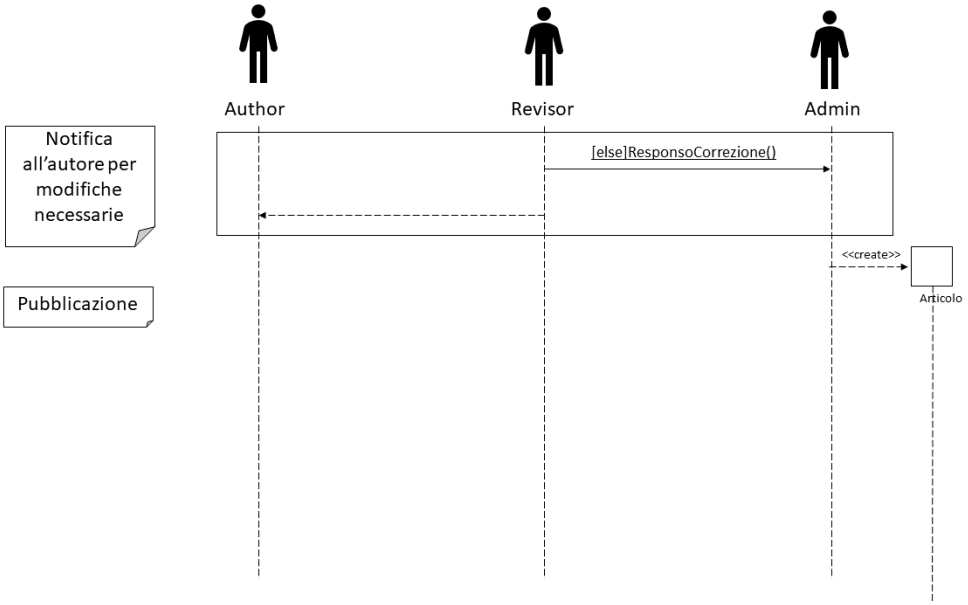

Figura 3.1.3 – Sequence diagram del processo di pubblicazione dei contenuti (parte 2)

- *State Diagram* (gestione pubblicazioni):

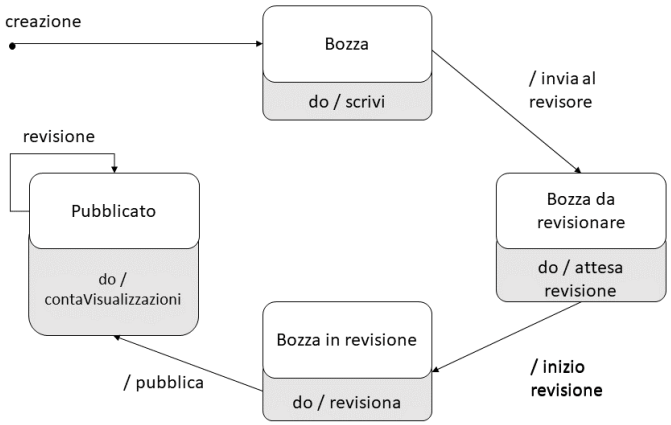

Figura 3.1.4 - State diagram del processo di pubblicazione dei contenuti

- Sequence Diagram (gestione ed organizzazione degli eventi):

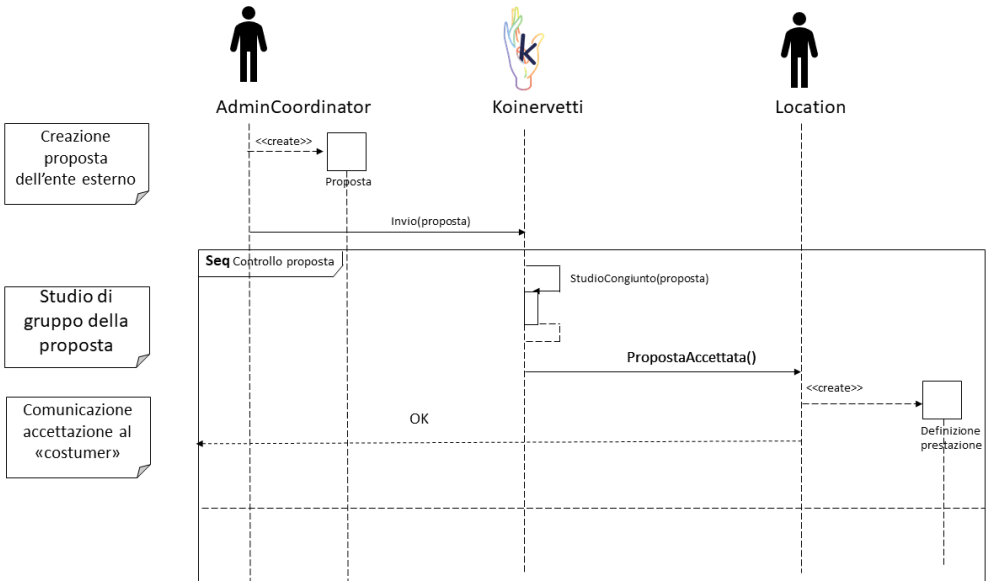

Figura 3.1.5 - Sequence diagram del processo di organizzazione e gestione di eventi (parte 1)

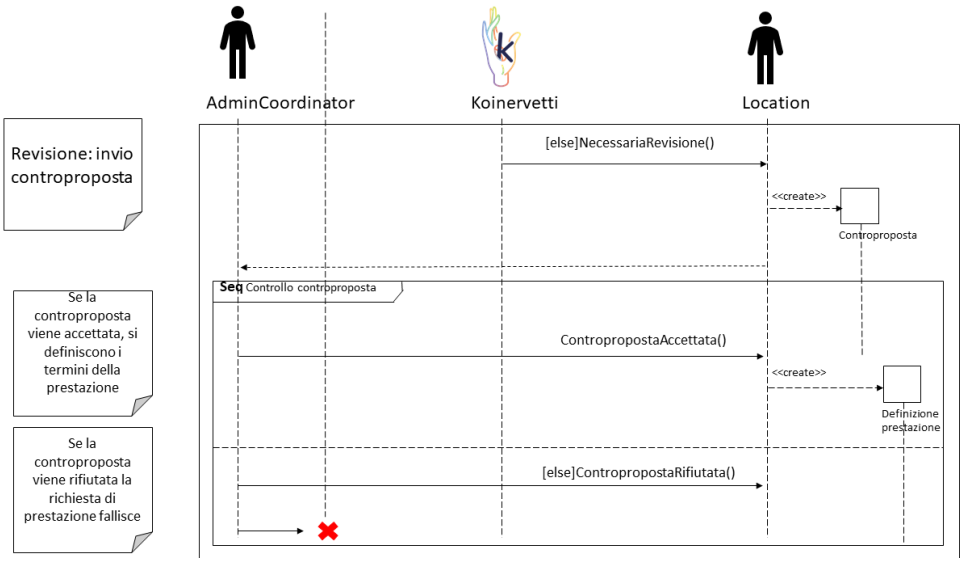

Figura 3.1.6 - Sequence diagram del processo di organizzazione e gestione di eventi (parte 2)

- Use Case Diagram della gestione del flusso documentale (figura 3.1.7) e della pianificazione delle attività (figura 3.1.8)

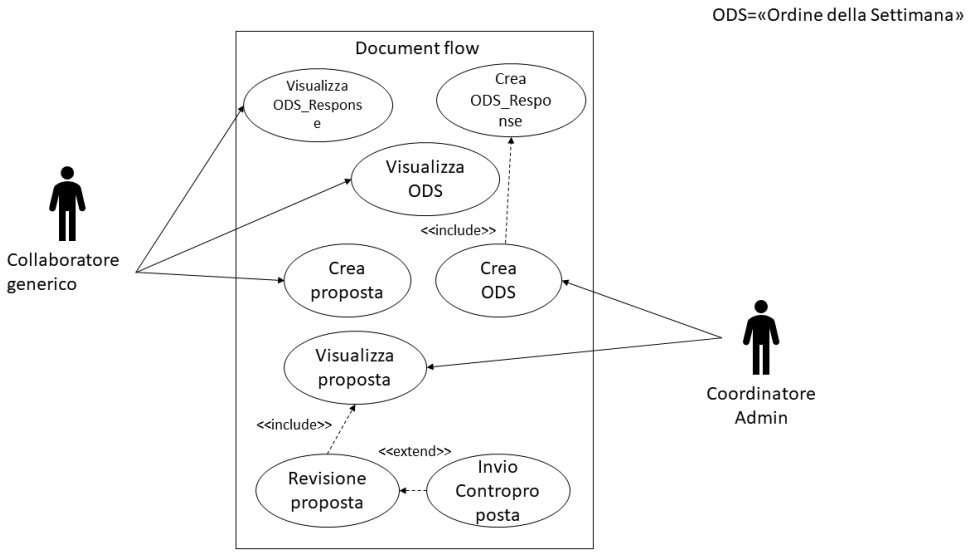

Figura 3.1.7 - Use case diagram per la gestione del flusso di documenti

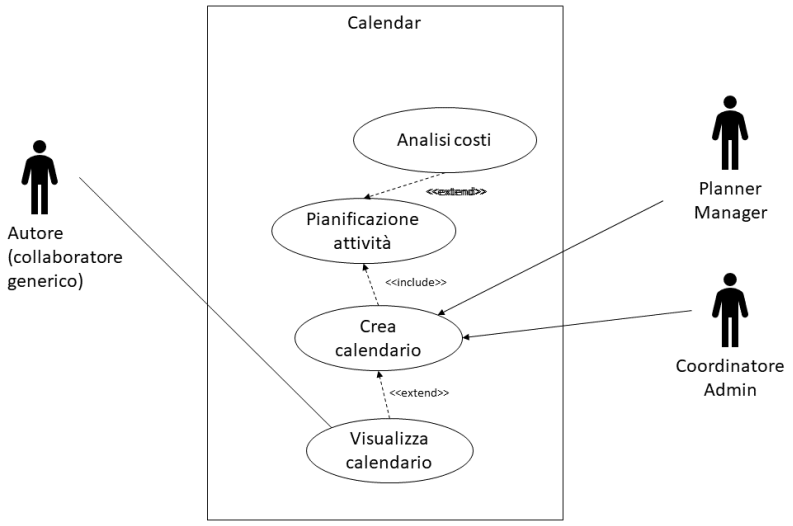

Figura 3.1.8 - Use case diagram per la pianificazione delle attività

- *Class Diagram*

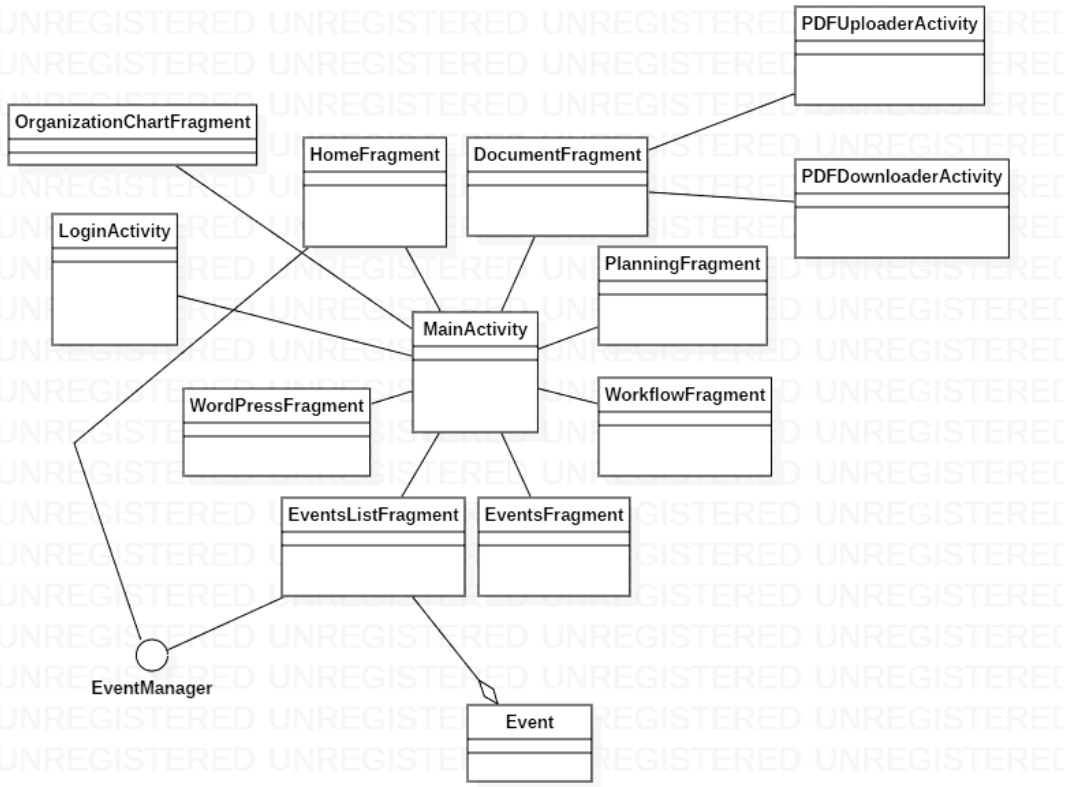

Figura 3.1.9 - Class diagram

- *Component Diagram*:

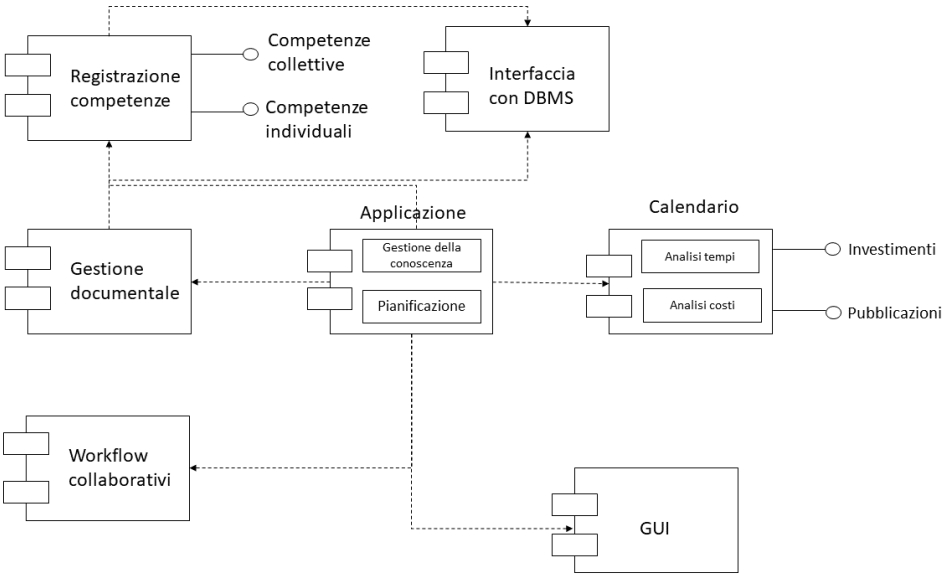

Figura 3.1.10 - Component diagram

- *Deployment Diagram*:

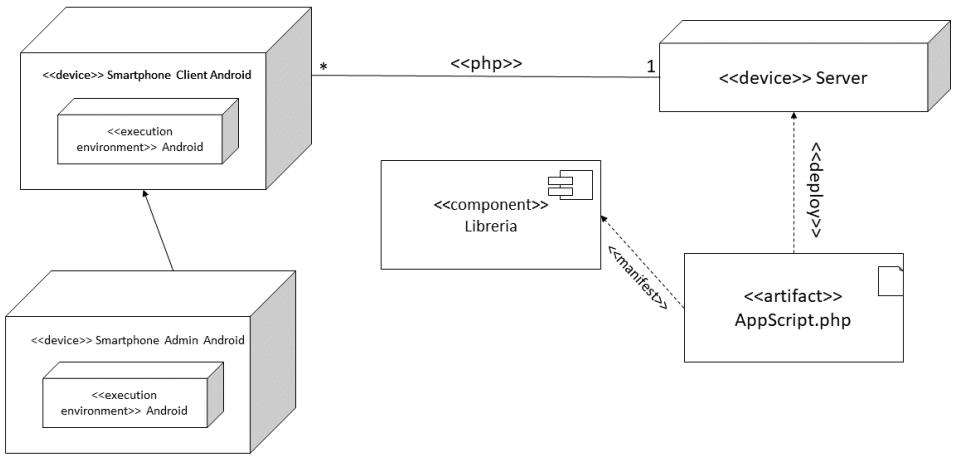

Figura 3.1.11 - Deployment diagram

4 Progettazione

Compito della fase di progettazione è definire come il sistema dovrà approcciarsi alle operazioni discusse nella precedente fase di analisi. In questo processo di trasformazione, conosciuto anche come “software design”, le specifiche dell’utente mutano in specifiche direttamente utilizzabili dai programmatori. La risultante della trasformazione sarà un insieme di moduli con una certa funzionalità che, relazionati fra di loro, costituiranno l’architettura del sistema e dei quali si definirà struttura e comportamento ai fini della corretta esecuzione del sistema.

Per la progettazione del software è buona norma:

- *Utilizzare formalismi e metodologie standardizzate* onde ridurre sensibilmente errori tipici della fase progettuale come inconsistenza ed ambiguità.
- *Saper anticipare i cambiamenti* (software o hardware che siano) prevedendo necessità future, con l’obiettivo di massimizzare riusabilità e facilità di manutenzione del software. Spesso, un cambiamento non è una semplice evoluzione del software paragonabile ad un servizio che si prevede di includere in futuro. Per questo, uno degli obiettivi è rendere il progetto facile da modificare.

- *Trattare separatamente le diverse peculiarità del sistema modularizzandolo*. La modularizzazione consiste in una suddivisione in base alla dimensione. Il sistema, a partire dalla sua forma più complessa e generica viene suddiviso in moduli (componenti aventi in comune funzionalità strettamente correlate fra loro), una scomposizione benefica dividente il sistema complesso in parti più semplici (beneficio evidente anche nel caso sia necessario apportare modifiche).
- *Rappresentare la divisione in moduli attraverso dei grafi* che descrivano relazioni “di utilizzo” e “di composizione”.
- Ignorare i dettagli del problema enfatizzando invece gli aspetti fondamentali (*principio di astrazione*)
- *Evidenziare in primis il problema più generale* (in quanto probabilmente più semplice o, addirittura, già risolto in un’applicazione esistente)

  Architetturalmente, il software in questione, previa applicazione della modularità, è descritto dai seguenti grafi, rispettivamente “is part of” (figura 4.1) e “uses” (figura 4.2):

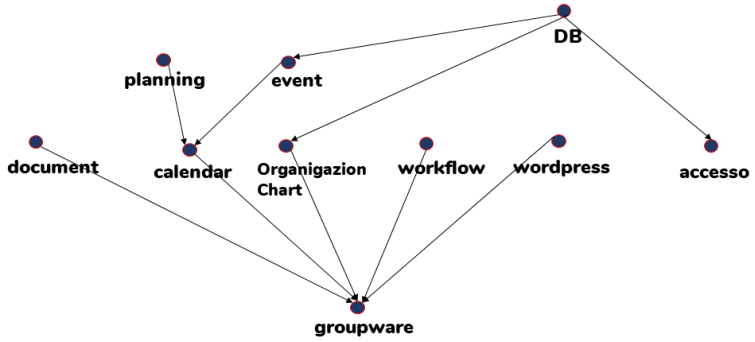

Figura 4.1 - Grafico "IS PART OF"

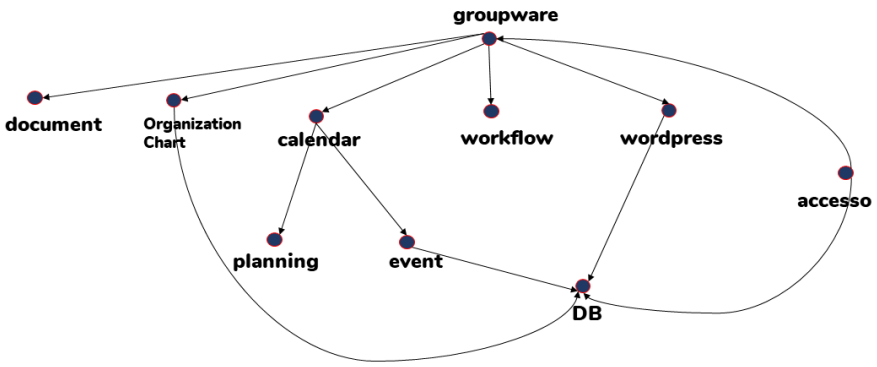

Figura 4.2 - Grafico "USES"

L’approccio scelto per la progettazione logica del sistema è quello ad oggetti. In realtà, il paradigma in questione, caratterizza ogni fase a partire dall’analisi fino all’implementazione, indicando rispettivamente “cosa” il sistema deve fare e “come” lo deve fare. I concetti fondamentali che lo caratterizzano sono:

- *Oggetti*: “individui sostanziali che possiedono un’identità ed un insieme di proprietà che ne definiscono stato e comportamento” nonché entità del dominio applicativo ed istanza di una classe;
- *astrazione*: struttura per i dati e interfaccia che definisce i servizi che un insieme di oggetti può implementare;
- *classe*: realizzazione di tipi di dati astratti (astrazioni);
- *incapsulamento*: “protezione” per l’oggetto che “nasconde” il suo stato e l’implementazione dei servizi che offre;
- *ereditarietà*: maccanismo che permette di basare la definizione e l’implementazione di una classe su quelle di altre classi;
- *polimorfismo*: capacità di un oggetto di assumere forme molteplici. Applicabile tramite i meccanismi di overload ed override;
- *late binding*: istanziamento dinamico grazie al quale si è svincolati, fino a run-time, dall’implementazione;
- *delegazione*: definizione di oggetti complessi che contengono al loro interno un riferimento ad un altro oggetto in modo da delegare ad esso alcune funzionalità.

I benefici più noti del paradigma ad oggetti riguardano la modellazione (in particolare la decomposizione del sistema complesso in moduli interagenti fra di loro), la flessibilità (conseguenza dell'incapsulamento che rimanda i dettagli implementativi nel tempo e li nasconde nelle classi), l'alta produttività, la possibilità di sviluppare prototipi rapidamente e, per ultimo ma non meno importante, la riduzione dei costi di manutenzione.

4.1 Specifiche del software

Considerando la natura del progetto ed in particolare i suoi attori principali, l'implementazione del software corrisponde ad un'applicazione sviluppata nativamente in Android per rendere più agevole la quotidiana interazione a distanza. L'applicativo si appoggerà al database MySQL ospitante il sito web Koinervetti (da precisare che l'applicazione attingerà al database esclusivamente per reperire informazioni su utenti, eventi programmati e documenti) e ad uno storage ospitato da Firebase. In virtù dell'obiettivo di massimizzare la collaborazione e l'efficienza del flusso informativo circolante fra i collaboratori (con particolare riguardo ai responsabili), i requisiti da soddisfare saranno:

- Accesso al sistema tramite le stesse credenziali che permettono di accedere al sito web;
- comunicare con il database per verificare accessi ed aggiornare informazioni, considerando che sarà popolato in corrispondenza della pianificazione di attività ed in caso di arrivo di nuovi collaboratori che necessiteranno di credenziali;
- possibilità di caricare (in uno storage remoto) e scaricare documenti;
- possibilità di programmare attività (da parte di Admin e Planner Manager);
- possibilità di comunicare direttamente con l'Amministratore di sistema;
- possibilità di accedere direttamente al backend (WordPress) del sito web per mettere in pratica un'attività;
- possibilità di inviare un messaggio, in caso di bozza ultimata, per attivare il processo di correzione;
- costante aggiornamento ai collaboratori circa cambiamenti riguardanti i flussi di lavoro, pubblicazione di documenti e pianificazione di attività e progetti;
- presentare un'interfaccia semplice e graficamente esplicativa

4.2 Workflow

I processi funzionali e decisionali di un team, in particolar modo se complessi, presentano la necessità di essere automatizzati. Un flusso di lavoro è l'automazione totale o (nel caso del gruppo di lavoro di Koinervetti) parziale di un processo aziendale, durante il quale i collaboratori si scambiano informazioni e prendono delle decisioni (in base al ruolo che li contraddistingue) rispettando un insieme di regole procedurali.

Esistono principalmente 3 tipologie di workflow:

- *Ad Hoc*: non presentano uno schema predefinito e risultano spesso complessi vista la necessità di cooperazione e coordinazione tra diverse risorse umane.
- *Administrative*: semplici e ripetitivi, sono processi più facilmente automatizzabili nella loro interezza.
- *Production*: ripetitivi ma complessi, nonostante possano essere completamente automatizzati. Definiti anche "mission critical", richiedono l'accesso a molte risorse eterogenee.

I flussi di lavoro ai quali il team fa riferimento sono Administrative Workflow:

- Ricerca contenuti e creazione bozza

Figura 4.2.1 - Workflow per la ricerca dei contenuti e la creazione della bozza che diventerà articolo

- Gestione pubblicazioni

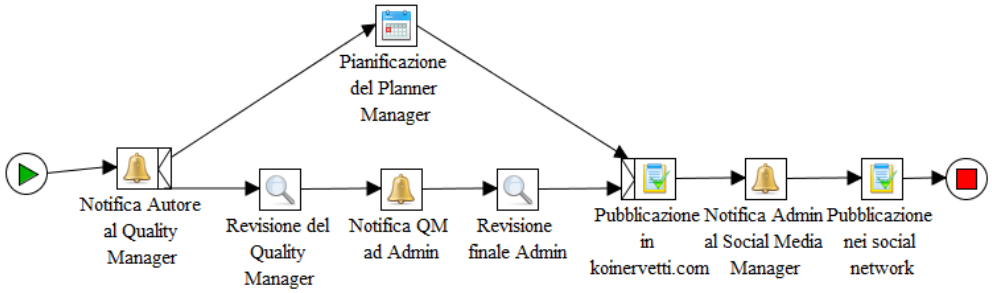

Figura 4.2.2 - Workflow di gestione delle pubblicazioni

- Gestione collaborazioni (equivalente alla gestione delle prestazioni)

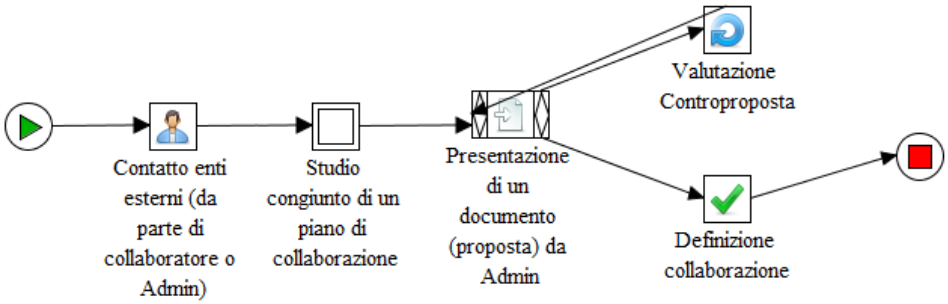

Figura 4.2.3 - Workflow di gestione delle collaborazioni

- Organizzazione eventi

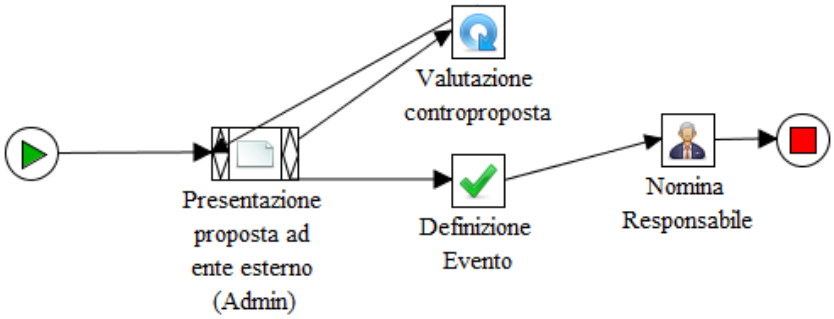

Figura 4.2.4 - Workflow di gestione ed organizzazione degli eventi

Le componenti delle cosiddette “net” (rappresentazioni dei workflow) sono le seguenti:

- *Atomic Task*: singolo task eseguibile da un collaboratore
- *Composite Task*: contenitore di una sottorete (nonché insieme di atomic task e/o composite task a sua volta), perciò non eseguibile da un singolo componente
- *Flow Relation*: rappresentazione di relazioni fra task
- *Split e join*: condizioni che personalizzano il task (utilizzati AND per identificare la separazione in flussi paralleli ed OR per separare flussi alternativi). I flussi si ricongiungono poi in un task tramite join.

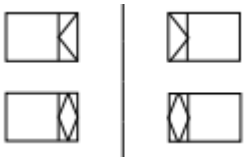

Figura 4.2.5 - Rispettivamente split AND ed OR a sinistra e join a destra

4.3 Focus group e mockup

Considerando caratteristiche, ruolo dell'utente finale (corrispondente ad un collaboratore del team) e specifiche predisposte in precedenza, come anticipazione di gradini successivi della progettazione e con lo scopo di discutere la struttura ancor prima di presentare un prototipo, si è resa indispensabile la formazione di un focus group. Assieme ai membri del team, sono stati esplorati vantaggi e svantaggi associabili ad una serie di opzioni riguardanti la rappresentazione del sistema. Un insieme di considerazioni rilevate come input per l'inizio della progettazione vera e propria, atte alla semplificazione del design ed al raggiungimento di un grado di soddisfazione accettabile da parte dell'utente finale.

In seguito alla discussione con il gruppo di lavoro, per rafforzare e rispettare il concetto di usabilità (facilità d'uso di un software, in breve), una fase topica della progettazione di un'applicazione mobile è quella di mockup. Si tratta della riproduzione di un oggetto "originale" (in questo caso dello smartphone e dell'interfaccia grafica dell'applicazione) a scopo dimostrativo e comunicativo. Questa rappresentazione grafica mostra come sarà il prodotto senza realizzarlo, ed è un buon approccio per anticipare eventuali discrepanze con le richieste del committente (ed in generale, con quelle che sono le sue aspettative nei confronti del software).

Di seguito sono mostrate degli sketch (ovvero dei mockup non eccessivamente dettagliati ma focalizzati sulle funzionalità da offrire all'utente):

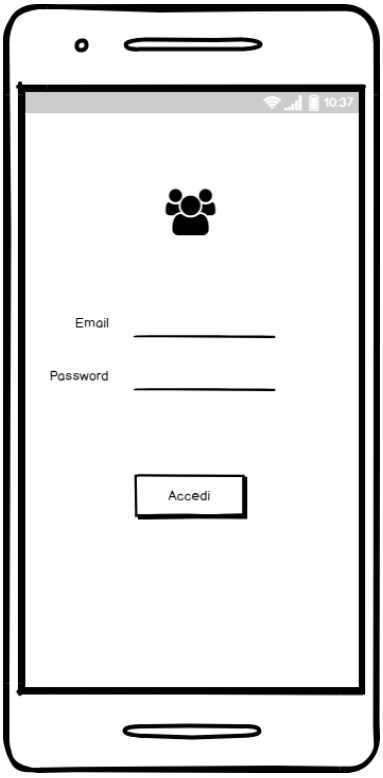

Figura 4.3.1 - mockup della schermata di login

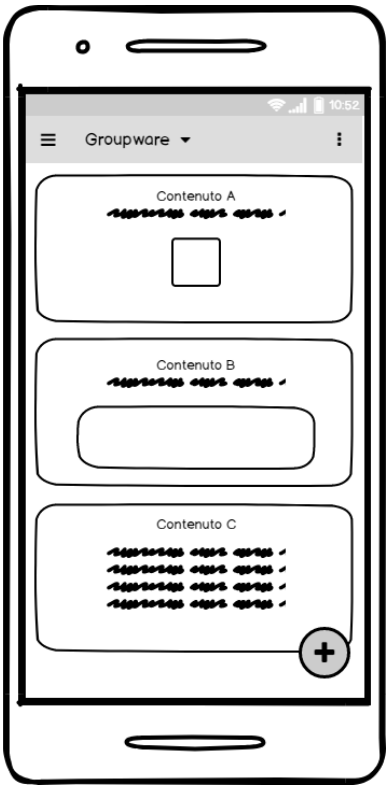

Figura 4.3.2 - Mockup della home

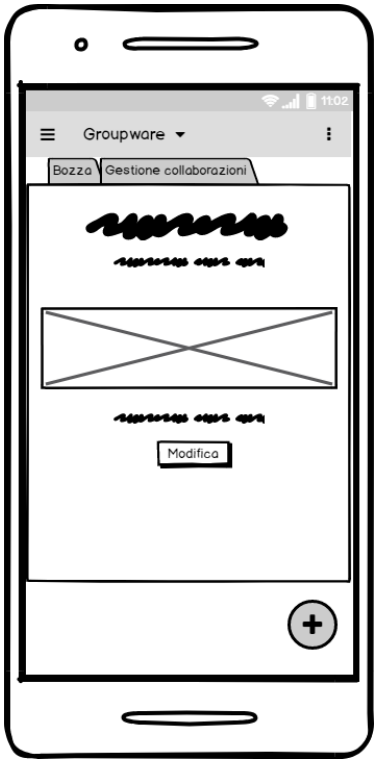

Figura 4.3.3 - Mockup schermata di workflow

5 Tecnologie utilizzate

Progettazione e sviluppo del groupware hanno previsto di adottare sia tecnologie apprese durante il corso di studi che tecnologie da considerare come background personale.

Essendo l'applicativo un supporto all'amministrazione di un team formato per la gestione di un sito web, l'operatività del gruppo di lavoro è basata sull'utilizzo di un Content Management System.

Il CMS utilizzato dal team è WordPress, piattaforma software open source in grado di gestire dinamicamente contenuti testuali e multimediali propri di un sito internet.

In un contesto di questo genere la pianificazione (con annessa analisi di tempi e costi associati alle attività programmabili e programmate) gioca un ruolo fondamentale. Per un approccio analitico, da integrare a considerazioni prettamente informative,

è stato adottato il diagramma GANTT (tipico strumento di supporto alla gestione di progetti), seppur in versione semplificata.

I processi di lavoro descritti dal workflow system hanno inoltre previsto lo studio di base di un linguaggio di supporto fondato sui workflow patterns. Il linguaggio in questione (XML based) è YAWL (Yet Another Workflow Language) ed è supportato da un software (open source) che include un ambiente esecutivo, un editor grafico ed un gestore di liste di lavoro.

5.1 WordPress

Amministratore e collaboratori del sito web necessitano di un uso massiccio del software, in relazione ad aspetti tecnici e puramente gestionali. Si denotano, rispettivamente, le necessità di adottare tecniche SEO per l'ottimizzazione nei motori di ricerca (abbinate alla conoscenza dei vari linguaggi di programmazione lato server e lato client per lo sviluppo e manutenzione di siti web) e di assegnare ruoli funzionali attraverso la gestione di permessi. WordPress, inoltre, è noto per la disponibilità di numerosi plugin (estensioni di funzionalità originarie) e temi per la personalizzazione del sito web, permette la creazione di pagine statiche (con la possibilità di costruirle a partire da modelli predefiniti), la gestione delle categorie e dispone di un editor per la formattazione dei testi. La gestione dell'utenza risulta flessibile (vedi la funzionalità di supporto multi-autore) in via generale, ma

può “irrigidirsi” grazie alla possibilità di blocco utente in base all’indirizzo IP.

Nell’ambito del sito web Koinervetti, come esplicitato, è quotidiano l’uso di software per l’ottimizzazione nei motori di ricerca (per cui sono previste verifiche su SEO e leggibilità). Si tratta di un insieme di attività volte a migliorare la scansione, l’indicizzazione e la catalogazione di un documento presente in un sito web da parte degli spider (bot che analizzano i contenuti di un database, o più in generale di una rete) dei motori di ricerca. Altri plugin supportano la gestione degli spazi pubblicitari (automatizzati da Google AdSense), delle statistiche (sfruttando la possibilità di log degli utenti), delle notifiche push e delle performance.

5.2 Diagramma GANTT

Grazie a questo approccio è possibile rappresentare graficamente un calendario di attività per pianificare, coordinare e tracciare le varie attività alle quali si associano delle risorse con relativo costo (orario e/o monetario). E’ costituito da due assi: quello orizzontale equivale all’arco temporale del progetto (che nel contesto in questione sarà una settimana), mentre quello verticale rappresenta le attività che lo costituiscono.

Ad ogni attività sono associati:

- *Risorse* (umane e non): quanto serve per realizzare e completare l’attività (nell’ambito di un sito web, per esempio, per pubblicare un articolo servono, banalmente, autore, dispositivo e connessione)

- *Costi*: non sempre presenti, possono corrispondere al tempo impiegato (un'attività richiedente tipicamente un costo è un'intervista in trasferta)
- *Tempi*: si tratta della durata dell'attività. Il tempo (approssimato) che si dovrà impiegare per portarla a termine.

5.3 Linguaggio YAWL

In questo ambito, l'utilizzo dei pattern ha una notevole valenza risolutiva essendo gli stessi atti alla risoluzione di problematiche ricorrenti in particolare nel campo delle applicazioni orientate ai processi.

Il linguaggio YAWL è fortemente influenzato dalle reti di Petri (rappresentazioni matematiche di un sistema distribuito discreto, nonché linguaggio di modellazione), supportando queste la maggior parte dei pattern. I tre costrutti base sono:

- Or-join (diramazione dei flussi)

- Cancellation sets (cancellazione di elementi dalla "rete")

- Attività multi-instance (insieme di attività fra loro correlate)

Di seguito è mostrata l'architettura dei moduli di YAWL:

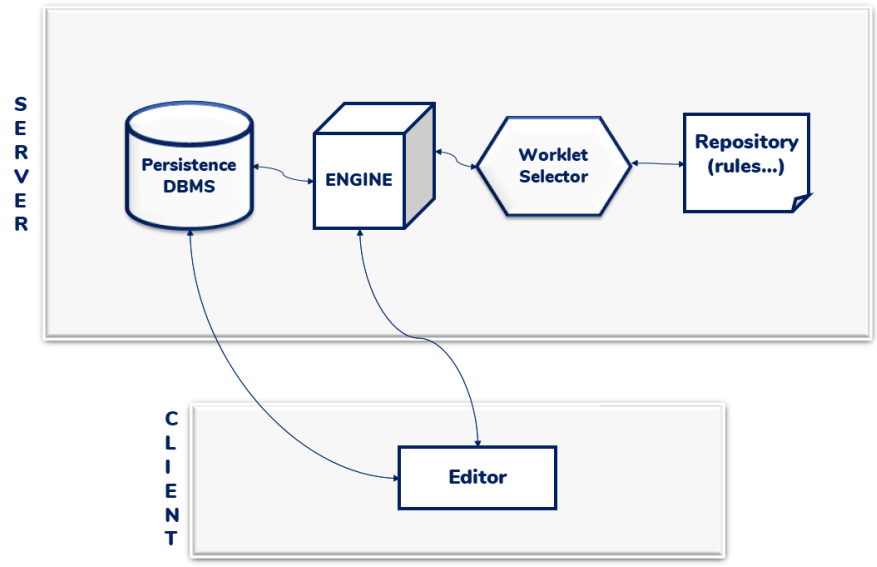

Figura 5.3.1 -Architettura del sistema YAWL

5.4 Android Studio ed applicazioni Android Material Design

Lo sviluppo di applicazioni Android offre la possibilità di usufruire di una vasta gamma di tecnologie (dipendenti ed indipendenti da software ed hardware esecutore dell'applicazione).

Il software utilizzato per lo sviluppo è Android Studio, IDE (basato su IntelliJ) progettato per l'implementazione di applicazioni Android. Questo offre la possibilità di utilizzare due linguaggi di programmazione: Kotlin (sviluppato da JetBrains) e Java (prodotto da Sun). Entrambi sono orientati agli oggetti ed interpretati dalla Java Virtual Machine (componente in grado di eseguire i programmi in bytecode dopo una prima fase di compilazione ed inclusa nel Java Development Kit). Tuttavia, per conoscenza pregressa e considerando come sia il linguaggio più

usato, la scelta è ricaduta sull'utilizzo di Java. Nel linguaggio suddetto "everything is an object", il che porta l'utilizzo di Java ad essere (in molti casi) una scelta conseguente alla progettazione di un sistema secondo le linee guida del paradigma ad oggetti.
L'accesso al database MySQL (RDBMS open source utilizzato da gran parte dei CMS sul mercato) ha inoltre reso necessario la scritturo di script appositi per la connessione. Il linguaggio lato server utilizzato per il loro sviluppo è il PHP (un linguaggio di scripting interpretato nato per sviluppare pagine web dinamiche). A differenza di Java, la sua tipizzazione è debole ma supporta comunque il paradigma di programmazione ad oggetti.

La scelta di sviluppare il software in Android ha come principale motivazione quella della diffusione del sistema operativo stesso (circa tre dispositivi su quattro al mando eseguono il sistema di Google), oltre alla possibilità di utilizzare un IDE ad-hoc per lo sviluppo nella piattaforma sopra citata ed in grado di disporre di funzionalità come:

- Un editor visuale utilizzabile anche in modalità drag & drop
- *SDK* ed *AVD Manager* per la gestione degli emulatori (dispositivi virtuali atti a simulare l'esecuzione dell'applicazione in ambiente Android)

- *Inline debugging* per una più immediata ispezione del codice in fase di debugging
- Il monitoraggio delle risorse (memoria e CPU su tutte) utilizzate dall'applicazione.

La struttura di un'applicazione Android si basa sull'Application framework, un insieme di API e componenti per gestire risorse, file system ed applicazioni di base del sistema. In generale, un progetto Android è strutturato come segue:

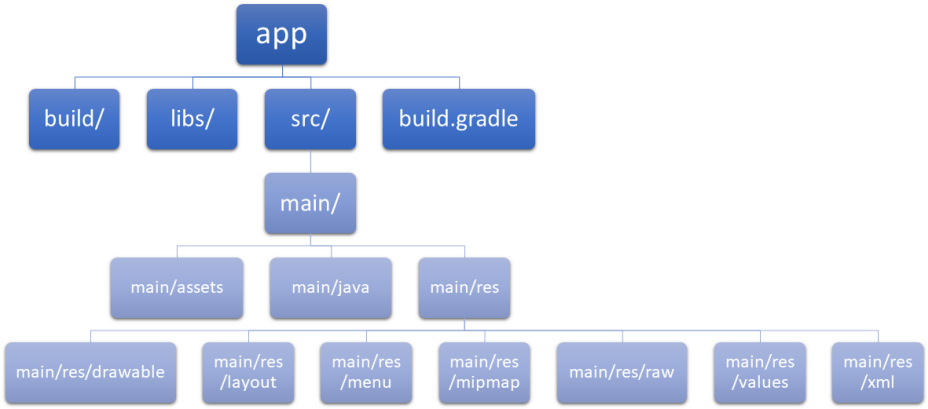

Figura 5.4.1 - Tipica architettura di un'applicazione Android

Da notare come in "libs/" sono presenti le librerie (JAR) in uso dall'applicazione, in "main/res" icone, layout, stringhe e temi, ed in "main/src" il codice sorgente. In "src" si trova inoltre anche il file "AndroidManifest.xml", descrittore di permessi, activity, service e molto altro.

Classi ed interfacce di Java, in Android possono corrispondere ad activity e fragment (oltre a service, async task ed altre componenti).

Un’activity corrisponde ad un’interfaccia grafica che permette all’utente di visualizzare ed interagire con essa secondo i principi previsti dallo sviluppatore. Più activity possono essere connesse fra loro, inoltre Android le organizza secondo un principio di stack, secondo il quale l’activity utilizzata in quel momento si trova in cima alla “pila”. Il richiamo di altre activity porrà quella corrente in pausa portando quella richiamata in cima allo stack. Ad ogni activity è associato un layout (file .xml), senza il quale non potrebbero essere visualizzate le informazioni. Grazie ai suddetti file si può determinare la disposizione della view e delle sue componenti (che, in base ad esso, varieranno anche la loro modalità di interazione).

Un fragment, invece, è collegato ad un’activity e permette la combinazione di molteplici schermate in una sola view. A differenza delle activity, i fragment possono essere statici o dinamici ed utilizzano il back stack (il che significa che il sistema non aggiunge i fragment automaticamente in uno stack, ma che una transazione viene memorizzata nella pila solo previo richiamo esplicito di un metodo).

Spesso, in Android, onde evitare il blocco dell’interfaccia grafica, è buona prassi far eseguire operazioni particolarmente onerose (temporalmente) ad un thread diverso da quello principale. Il lancio di uno script PHP che permetta l’accesso al database è un tipo di operazione breve, ma che potrebbe richiedere tempo. La soluzione a questa problematica corrisponde all’implementazione di un AsyncTask, processo asincrono in

grado di eseguire operazioni in background (come la lettura delle righe di una tabella tramite lo script).

Approccio differente è quello dei service, utilizzati invece per l'esecuzione di lunghe operazioni in background. Esistono due tipo di service:

- *Started*: vengono eseguiti in background indefinitamente (fino all'interruzione esplicita), anche se la componente che li ha avviati è terminata.
- *Bound*: vengono utilizzati solo in presenza di client interagenti e si interrompono nel momento in cui non esistono più client collegati ad esso.

Una delle applicazioni più frequenti dei service sono i sistemi di notifiche (vedi invio di messaggi broadcast tramite broadcast receiver).

La progettazione di un'applicazione mobile (indipendentemente dalla piattaforma) è fortemente orientata agli utenti. Nel web e nel mondo delle applicazioni per dispositivi mobili il design gioca un ruolo fondamentale e così la "User Experience" come base della fase di progettazione del software richiede che il processo antecedente l'implementazione abbia una forte affinità con le aspettative dell'utente finale.

Il material design (annunciato da Google nel 2014) anticipa le intenzioni dell'utente "materializzando" la dimensione digitale con la quale interagire attraverso la rappresentazione di bordi e di una superficie fisica caratterizzata da ombre e sovrapposizioni che "danno informazioni su ciò che si può toccare e come si muoverà".

In generale, il material design si applica a colori (solitamente vivaci, in contrasto con aree tenui come sfondo), font (tenendo presente la necessità di aumentare la leggibilità aumentando il contrasto), immagini (mantenute prive di filtri) ed icone (dal design minimale).

Android Studio dispone delle componenti “materiali” descritte sopra, motivo per il quale la progettazione dell’applicazione ha delle buone fondamenta costituite dalle generali linee guida in materia predisposte da Google.

5.5 HMTL, CSS, PHP e JSON

Come già dichiarato, gli script per la comunicazione con il database usufruiscono del linguaggio lato server PHP. In particolare, il risultato prodotto dalle interrogazioni al database MySQL sarà a disposizione dell’applicazione grazie all’incapsulamento dello stesso in un oggetto JSON (non prima di essere stato codificato). Entrando nei dettagli, JSON è un formato (basato sul linguaggio JavaScript) per l’interscambio di dati fra applicazioni client-server. Risulta utilizzato di frequente in quanto facile da manipolare per i programmatori ed altrettanto semplice da generare ed analizzare per le macchine. HTML (HyperText Markup Language), invece, è un linguaggio di markup nato per la formattazione ed impaginazione di documenti ipertuastuali presenti nel web. In conclusione, CSS, (Cascading Style Sheets) il quale è un insieme di regole atte alla regolazione dello stile (resa grafica) di un documento.

5.6 Firebase – Storage remoti

La web application in questione, oltre alle funzionalità di cloud messaging, realtime database ed hosting, è in grado di gestire storage per upload e download sicuri (indipendentemente dalla qualità della rete). Il sistema può essere sfruttato per archiviare immagini, audio, video o altri contenuti generati dagli utenti registrati all'applicazione. Firebase richiede tuttavia di essere associato all'applicazione che usufruirà dei suoi servizi tramite un processi di autenticazione.

5.7 API varie

Lo sviluppo del software prevede, inoltre, anche l'utilizzo di alcune Application Programming Interface (API) per la gestione del calendario, del download ed upload dei documenti e della resa grafica dei workflow.

6 Implementazione

Il progetto è diviso in package che raggruppano classi per funzionalità.

Il seguente workflow mostra come le componenti del sistema interagiscono:

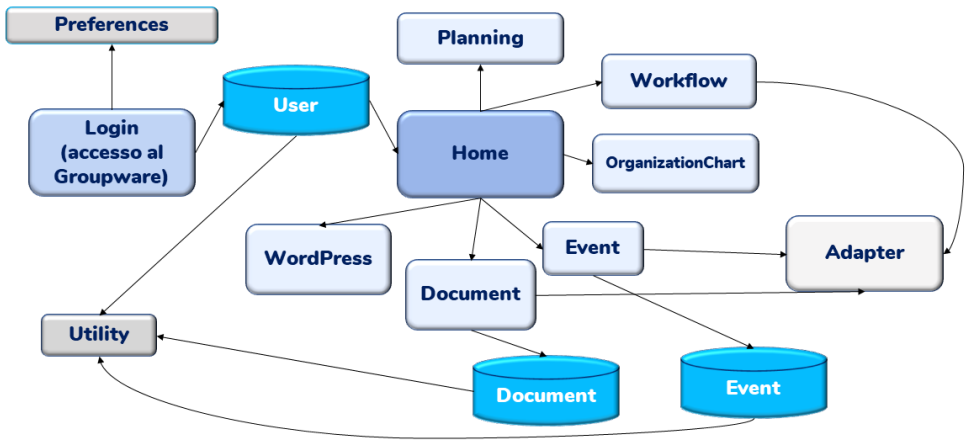

Figura 6 - Workflow delle componenti del sistema

6.1 LoginActivity, MainActivity e HomeFragment

Il collaboratore avrà la possibilità di utilizzare l'applicazione previo accesso tramite le stesse credenziali utilizzate per accedere al sito web. Una volta effettuato l'accesso sarà in grado di usufruire delle funzionalità compatibili con i suoi privilegi.

Inserimento e verifica delle credenziali sono caratteristiche della *LoginActivity.*

La classe in questione preleva (in background) gli utenti dal database MySQL (tramite uno script PHP eseguito da un task asincrono) per confrontare i dati inseriti dall'utente con quelli presenti nella base di dati. I dati ottenuti (nonché risultato della query eseguita dallo script) vengono manipolati dall'applicazione tramite l'incapsulamento in oggetti JSON. Il metodo addetto è *downloadJSON(final String urlWebService),* il quale ha come parametro una stringa che corrisponderà all'URL identificante lo script PHP. Il metodo suddetto implementa una classe innestata come specializzazione di AsyncTask, la quale attraverso *doInBackground(Void… voids)* apre una connessione HTTP per connettersi allo script e leggere il contenuto del JSON risultante.

```
private void downloadJSON(final String urlWebService) {

    class DownloadJSON extends AsyncTask<Void, Void,
String> {
```

```
        @Override
        protected void onPreExecute() {
            super.onPreExecute();
        }


        protected void onPostExecute(final String s) {
            super.onPostExecute(s);
            try {
                addUser(s);
            } catch (JSONException e) {
                e.printStackTrace();
            }
        }


        protected String doInBackground(Void... voids) {
            try {
                URL url = new URL(urlWebService);
                //Open HTTP connection
                HttpURLConnection con = (HttpURLConnection)
url.openConnection();
                StringBuilder sb = new StringBuilder();
                BufferedReader bufferedReader = new
BufferedReader(new
InputStreamReader(con.getInputStream()));
                String json;
                //Read JSON like a file
```

```
            while ((json = bufferedReader.readLine()) != null) {
                sb.append(json + "\n");
            }
            return sb.toString().trim();
        } catch (Exception e) {
            return null;
        }

    }
  }
  DownloadJSON getJSON = new DownloadJSON();
  getJSON.execute();
}
```

I JSON object restituiti verranno poi caricati in un JSONArray. Il vettore verrà scorso da un ciclo determinato.

All'interno del ciclo, da ogni JSON object si preleveranno oggetti del tipo corretto (*String*, in questo caso) per poi aggiungerli ad una lista di tipo *User.* Gli oggetti di tipo User presentano tre stringhe equivalenti ai dati dell'utente (email, nome e password), tuttavia, l'omonima classe implementa due costruttori ed in questa fase viene utilizzato quello al quale vengono passati solamente due parametri (i soli necessari a verificare l'accesso). A richiamare il metodo garante del caricamento della lista è *onPostExecute(final String s)* (il cui tipo del parametro deve necessariamente corrispondere a quello specificato nella definizione della classe attraverso l'AsyncTask). In questo caso l'esecuzione non avviene in background ed i risultati dell'esecuzione sono visibili dalla view.

Eseguito l'accesso, si avrà la possibilità di interagire con la Home (*MainActivity*). Il layout dell'activity principale prevede la presenza di:

- Una *NavigationView (figura 6.2)* contenente il menù principale per accedere alle varie funzionalità del groupware ed informazioni dell'utente acceduto (come la mail, il ruolo all'interno del team e l'immagine del profilo).
- Un'*AppBarLayout* (manipolata con un *ActionBarDrawerToggle*) con un menù "secondario" per feedback ed istruzione per l'uso (guida)
- Un *frameLayout* come schermata principale per raffigurare a sua volta il layout dei fragment (dinamici) "contenitori" delle varie funzionalità messe a disposizione dei collaboratori. Uno di questi è l'HomeFragment (figura 6.1):

Figura 6.1 - HomeFragment e cardView

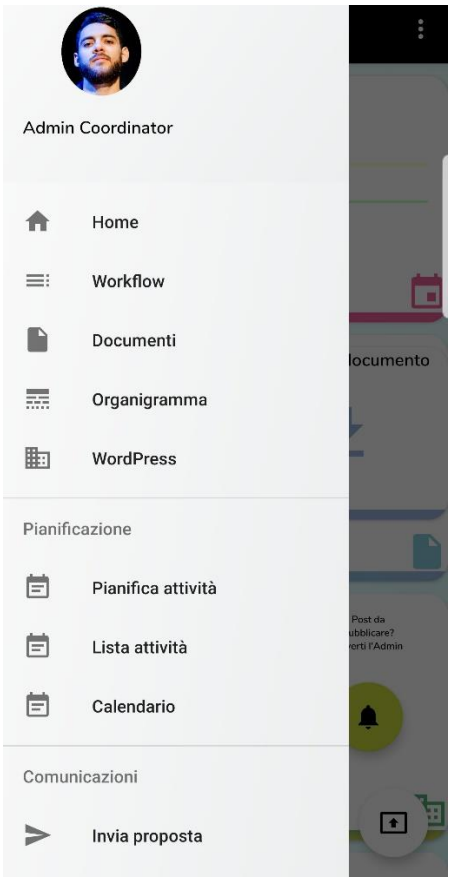

Figura 6.2 - Menù (NavigationView)

La *MainActivity,* seppur non in modo molto evidente è personalizzata in base all'utente che vi ha fatto accesso. Il collaboratore infatti potrà visualizzare la propria immagine di profilo (nella home come nell'activity precedente, dopo il primo accesso) grazie al download della stessa memorizzata in un storage remoto su Firebase. La libreria di cui si usufruisce è *Glide:*

```
Glide.with(this)

.load("https://firebasestorage.googleapis.com/v0/b/koinervetti-groupware.appspot.com/o/images%2FAdmin_round.png?alt=m
```

```
edia&token=91d5770a-c4a1-4934-b57c-cae8afd8c93a")
        .into(navImg);
```

Il suo uso prevede di specificare il contesto (activity o fragment), l'URL dal quale scaricare l'immagine ed in quale ImageView caricarla.

La personalizzazione riguarda anche mail e ruolo, motivo per il quale le credenziali inserite nelle EditText della LoginActivity saranno passati come parametri alla MainActivity al momento del suo lancio. Per adempire a questa necessità si utilizzano *Intent* (con putExtra(), che richiede l'inserimento di un valore etichettato con una chiave) ed il metodo *StartActivityForResult():*

```
private final int LOGIN_REQUEST = 1;
public static final String VALUE_PASSED_KEY =
"VALUE_PASSED_KEY";

Intent intent = new Intent(this, MainActivity.class);
intent.putExtra(VALUE_PASSED_KEY,email);

startActivityForResult(intent, LOGIN_REQUEST);
```

Il dato può poi essere ottenuto dalla ricevente con:

```
if(getIntent()!=null){
    email =
getIntent().getStringExtra(LoginActivity.VALUE_PASSED_KEY)
;
}
```

I dati utente, una volta convalidati, rimarranno memorizzati anche in seguito alla chiusura dell'applicazione. Questa feature è resa possibile dalle *preferences* (implementate per mail, nome e password). Di seguito un esempio di codice per il salvataggio della password:

```
public class SetPreferencesPassActivity {
    private static final String LOG_SESSION_MOOD =
"LOG_SESSION_MOOD";
    private static final String PASS_LOGGED_CAT = "
PASS_LOGGED_CAT";

    public static boolean isLogged(Context context){
        final SharedPreferences preferences =
getPrivateSharedPreferences(context);
        return preferences.contains(PASS_LOGGED_CAT) &&
!preferences.getString(PASS_LOGGED_CAT, "").isEmpty();
    }

    public static String getUserLogged(Context context){
        return
getPrivateSharedPreferences(context).getString(PASS_LOGG
ED_CAT, "");
    }

    public static void logIn(Context context, String categoria){
```

```
getPrivateSharedPreferences(context).edit().putString(PASS_LOGGED_CAT, categoria).apply();
    }

    public static void logOut(Context context){

getPrivateSharedPreferences(context).edit().remove(PASS_LOGGED_CAT).apply();
    }
}
```

Come per *putExtra()* si denota la presenza di chiavi associate al valore, mentre i metodi utili riguardano la verifica del logging (previo controllo del contenuto della "preferenza" che stabilisce quindi se l'utente è loggato o no), l'ottenimento di informazioni dell'utente acceduto ed i metodi di *login* (con *putString()* del metodo *edit()* per modificare le preferenze) e *logout* ((con *remove()*).

Per completare la panoramica sulla MainActivity è bene evidenziare come il frame appartenente al "content main" ospiti i vari fragment implementati, eseguibili tramite il metodo *setFragment()*:

```
private void setFragment(Fragment fragment){
    FragmentManager fragmentManager=getSupportFragmentManager();
    FragmentTransaction fragmentTransaction=fragmentManager.beginTransaction();
```

```
    fragmentTransaction.replace(R.id.frameLayout, fragment);
    fragmentTransaction.commit();
}
```

Per gestire i fragment le activity possono avvalersi della classe FragmentManager (i quali oggetti sono recuperabili tramite *getSupportFragmentManager()*) che permette l'apertura di una FragmentTransaction chiamando il metodo *beginTransaction()* attraverso la quale aggiungere o rimuovere fragment dinamicamente.

Il layout della home è "relative" e presenta delle *materialCardView (*contraddistinte da una toolbar*).* Degna di nota è la "card" che presenta l'ultimo evento in ordine temporale, "misurato" in termini di tempo e costo con delle *seekBar.*

Le componenti della view vengono disabilitate ed assumono un colore differente in base al valore monetario e temporale dell'evento.

Le funzionalità dichiarate sono sviluppate tramite dei fragment:

6.2 DocumentFragment

La gestione del flusso documentale passa attraverso alcuni step. In particolare, il fragment principale presenta delle tracce da seguire per creare dei documenti standard e la possibilità di caricare e scaricare documenti (come Ordine Della Settimana e proposte di vario genere) attraverso *PDFUploaderActivity()* e *PDFDownloaderActivity().*

I documenti PDF d'interesse sono memorizzati nello storage virtuale di Firebase, motivo per il quale si è reso necessario includere l'omonima libreria:

```
implementation 'com.google.firebase:firebase-storage:16.0.4'
```

L'upload di un documento (PDF) implica di istanziare un nuovo Intent con definita la specifica azione di ottenere contenuti dalla memoria del dispositivo e la funzionalità in grado di far selezionare all'utente un file (in questo caso di un tipo predeterminato) identificato dal suo URI per essere passato all'activity chiamante (metodo *onActivityResult()*).

```
private void showFileChooser() {
    Intent intent = new Intent();
    //kind of data you want to choose
    intent.setType("application/pdf/*");
    //kind of action you want to take
    intent.setAction(Intent.ACTION_GET_CONTENT);
    //Opening a view to select the PDF
    startActivityForResult(Intent.createChooser(intent, "Seleziona
un PDF"), PICK_DOC_REQUEST);
}
```

Il file selezionato sarà caricato nel server previa esecuzione del seguente metodo:

```
private void uploadFile() {
    if (filePath != null) {
        final ProgressDialog progressDialog = new
ProgressDialog(this);
        progressDialog.setTitle("Sto caricando il file...");
        progressDialog.show();

        //The randomUUID() method is used to retrieve a type 4
(pseudo randomly generated) UUID
        StorageReference ref = storageReference.child("ODS/" +
UUID.randomUUID().toString());

        //"ref" uses putFile() to upload the PDF on server
        ref.putFile(filePath)
                .addOnSuccessListener(new
OnSuccessListener<UploadTask.TaskSnapshot>() {
                    @Override
                    public void onSuccess(UploadTask.TaskSnapshot
taskSnapshot) {
                        progressDialog.dismiss();
                        Toast.makeText(PDFUploaderActivity.this, "File
caricato!", Toast.LENGTH_SHORT).show();
                    }
                })
                .addOnFailureListener(new OnFailureListener() {
                    @Override
                    public void onFailure(@NonNull Exception e) {
                        progressDialog.dismiss();
                        Toast.makeText(PDFUploaderActivity.this, "File
```

```
non caricato! " + e.getMessage(),
Toast.LENGTH_SHORT).show();
                    }
                })
                .addOnProgressListener(new
OnProgressListener<UploadTask.TaskSnapshot>() {
                    @Override
                    public void onProgress(UploadTask.TaskSnapshot
taskSnapshot) {
                        double progress = (100.0 *
taskSnapshot.getBytesTransferred() / taskSnapshot
                                .getTotalByteCount());
                        progressDialog.setMessage("Stato caricamento:
" + (int) progress + "%");
                    }
                });
    }
}
```

L'oggetto *StorageRefence* è un riferimento allo storage di Firebase, ed implementa *putFile()*, il quale gestisce le casistiche di successo e fallimento (con visualizzazione di un messaggio a scomparsa (*toast)* come "notifica") del caricamento e definisce il comportamento della view nella fase intermedia con un *progressDialog* (per visualizzare in percentuale lo stato dell'upload). La percentuale di caricamento non è simulata grazie al metodo *getBytesTransferred()* del task responsabile dell'upload, che indica quanti byte sono stati trasferiti in tempo reale.

Diametralmente opposto il downloader. Firebase dispone di una classe denominata DownloadManager, utile a definire il servizio e la richiesta (per via della quale si avrà la possibilità di impostare anche la directory di destinazione):

```
private void downloadFile(Context context, String fileName,
String fileExtension, String destinationDirectory, String url){
    DownloadManager downloadManager = (DownloadManager)
context.getSystemService(Context.DOWNLOAD_SERVICE);
    Uri uri=Uri.parse(url);

    //download request from the specified URI
    DownloadManager.Request request=new
DownloadManager.Request(uri);

    //set destination folder and a notification upon completion

request.setNotificationVisibility(DownloadManager.Request.VIS
IBILITY_VISIBLE_NOTIFY_COMPLETED);
    request.setDestinationInExternalFilesDir(context,
destinationDirectory, fileName+fileExtension);


    //there may be more than one request, so they are queued
    downloadManager.enqueue(request);
}
```

Il documento da scaricare viene selezionato da una listView presentante l'elenco di documenti contrassegnati nel database e memorizzati in Firebase, ottenuto tramite l'esecuzione di uno

script PHP (simile al precedente) atto alla produzione di un JSON contenente i documenti. Al tocco di un elemento della lista, previo passaggio (al metodo sottostante) del titolo del documento come parametro, avrà luogo il download del corrispettivo PDF.

Il metodo *child(),* infine, indica l'interesse verso un elemento dello storage e permette di ottenere il suo URL (*getDownloadUrl()*) per il download:

```
private void download(final String titolo) {

storageReference.child("ODS/"+titolo+".pdf").getDownloadUrl().
addOnSuccessListener(new OnSuccessListener<Uri>() {
        @Override
        public void onSuccess(Uri uri) {
            String url=uri.toString();
            downloadFile(PDFDownloaderActivity.this,
titolo,".pdf",DIRECTORY_DOWNLOADS, url);
        }
    });
}
```

6.3 EventsFragment

Il fragment sopra nominato è addetto all'aggiunta di un evento tramite web view (visualizzazione in-app di una pagina web, figura 8). Considerando la polivalenza del database che ospita anche alcune tabelle utilizzate dall'applicazione, si è deciso di sfruttare una semplice form già implementata ed utilizzata nel sito web.

La form invia i dati (via metodo POST) ad uno script PHP che effettuerà l'inserimento nel database con un INSERT INTO. Piuttosto tipica anche la definizione dello stile (CSS).

Da denotare la presenza di transizioni ed animazioni implementate con il pacchetto webkit-keyframes che definisce lo stile che sarà applicato all'oggetto al momento dell'animazione.

Per visualizzare una pagina web è necessario un client che "sovrascriva" la view del layout con un apposito metodo che sarà utilizzato dal fragment:

```
public class MyWebViewClient extends WebViewClient {
    @Override
    public boolean shouldOverrideUrlLoading(WebView view,
String url) {
        view.loadUrl(url);
        return true;
    }

    public boolean onKeyDown(int keyCode, KeyEvent event,
WebView myWebView) {
        // Check if the key event was the Back button and if there's
history
        if ((keyCode == KeyEvent.KEYCODE_BACK) &&
myWebView.canGoBack()) {
            myWebView.goBack();
            return true;
        }
        return onKeyDown(keyCode, event, myWebView);
```

```
    }
}

private void startWebView(String url){
    webView.setVisibility(View.VISIBLE);
    wvClient=new MyWebViewClient();
    webView.loadUrl(url);
    webView.setWebViewClient(wvClient);
    //to “go away” from web view and to go on the previous
activity/fragment
    webView.canGoBack();
    //client’s method
    wvClient.shouldOverrideUrlLoading(webView,url);
}
```

6.1 EventsListFragment

La lista degli eventi programmati viene disposta dal fragment in questione, il quale implementa ancora una volta un async task per adempire al suo compito. Ogni elemento della list view (figura 9) ha una sua struttura personalizzata resa possibile dall’uso congiunto di una classe “modello” ed apposito *adapter*.

La classe Event modella un evento definendo campi, costruttore ed annessi “getter” e “setter”. Come già evidenziato, gli elementi della lista sono oggetti di tipo Event.

La classe EventAdapter (specializzazione di ArrayAdapter<Event>) adotta una conversione della vista per adattare gli elementi della lista ad un layout ad-hoc per oggetti Event:

```
public View getView(int position, View convertView, ViewGroup parent) {
    LayoutInflater inflater = (LayoutInflater) getContext().getSystemService(Context.LAYOUT_INFLATER_SERVICE);
    convertView = inflater.inflate(R.layout.event_layout, null);

    final EventViewHolder eventViewHolder;

    TextView txtNomeEvento=convertView.findViewById(R.id.txtNomeEvento1);
    TextView txtData=convertView.findViewById(R.id.txtData1);
    TextView txtRisorse=convertView.findViewById(R.id.txtRisorse1);
    TextView txtCosto=convertView.findViewById(R.id.txtCosti1);
    TextView txtTempi=convertView.findViewById(R.id.txtTempi1);

    eventViewHolder=new EventViewHolder(txtNomeEvento,txtData,txtRisorse,txtCosto,txtTempi);

    Event c = getItem(position);
```

```
    txtNomeEvento.setText(c.getNome());
    txtData.setText(c.getData().toString());
    txtRisorse.setText(c.getRisorsa());
    txtCosto.setText(c.getCosto());
    txtTempi.setText(c.getTempo());

    return convertView;
}
```

Il costruttore formerà la view apposita attraverso il layout dedicato e la lista di eventi “popolata” dal task asincrono in background:

```
EventAdapter arrayAdapter = new EventAdapter(getContext(),
R.layout.event_layout, events);
//It assigns the adapter to the list
listView.setAdapter(arrayAdapter);
```

6.4 OrganizationChartFragment

Il fragment per rappresentare l'organigramma (figura 10) è statico a meno del caricamento dell'immagine di profilo dei responsabili attraverso Glide.

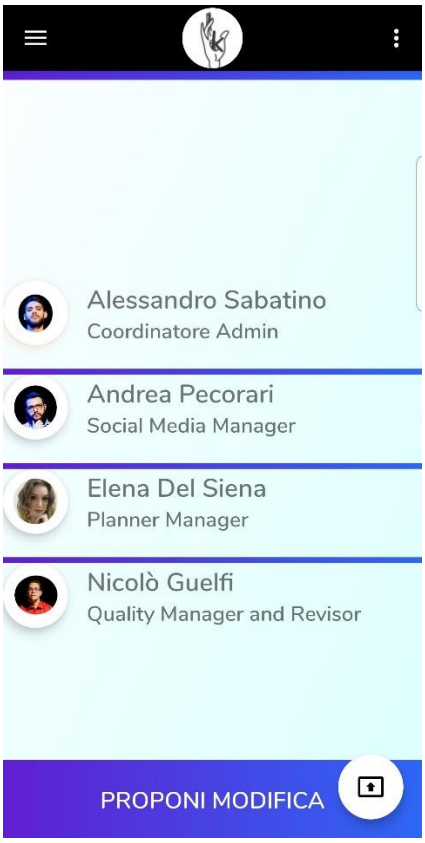

Figura 6.4.1 – Organigramma

6.5 WordPressFragment

Come per aggiungere un evento, anche per accedere a WordPress si fa uso di una web view che reindirizza alla pagina di login del CMS.

6.6 WorkflowFragment

Il layout del *WorkflowFragment* si discosta dagli altri per l'implementazione di *TabLayout* e *ViewPager.*

Seppur per motivi differenti, si utilizza nuovamente un adapter. In particolare è stato necessario "estendere" un FragmentPagerAdapter:

```
public class WorkflowAdapter extends FragmentPagerAdapter {

    private Context myContext;
    int totalTabs;

    public WorkflowAdapter(Context context, FragmentManager
fm, int totalTabs) {
        super(fm);
        myContext = context;
        this.totalTabs = totalTabs;
    }

    // this is for fragment tabs
    @Override
    public Fragment getItem(int position) {
        switch (position) {
            case 0:
                return new DraftFragment();
            case 1:
                return new PublicationsFragment();
            case 2:
                return new CollaborationsFragment();
```

```
            case 3:
                return new EventsManagingFragment();
            default:
                return null;
        }
    }

    // this counts total number of tabs
    @Override
    public int getCount() {
        return totalTabs;
    }
}
```

Ad ogni “tab” corrisponderà un fragment raffigurante (attraverso la view) il workflow. L’immagine del flusso di lavoro può essere ingrandita e rimpicciolita. Ci si è per questo serviti di una libreria apposita chiamata “PhotoView”:

```
implementation 'com.github.chrisbanes:PhotoView:1.2.6'
```

Un oggetto di tipo PhotoViewAttacher viene creato ed associato all’immagine che si desidera manipolare ed il metodo update(), in background, aggiorna la view e dimensioni dell’imageView stessa (figure 12 e 13):

```
PhotoViewAttacher photoView=new
PhotoViewAttacher(imgPublicationsWF);
photoView.update();
```

Alcuni utenti hanno la possibilità di proporre delle modifiche ai workflow. Verificando quale utente ha effettuato l'accesso si attiva o meno la funzionalità:

```
if(SetPreferencesMailActivity.getUserLogged(getContext()).equals("nicolo.guelfi@koinervetti.com") || SetPreferencesMailActivity.getUserLogged(getContext()).equals("admin@koinervetti.com")){
    btnUpload.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            sendMail("admin@koinervetti.com");
        }
    });
}else{
    btnUpload.setActivated(false);
    Toast.makeText(getContext(),"Non hai i provilegi per poter modificare i workflow",Toast.LENGTH_LONG).show();
}
```

Per il groupware, la comunicazione è centrale, motivo per il quale la possibilità di inviare mail è un must:

```
public void sendMail(String mail) {
    Intent i = new Intent(Intent.ACTION_SEND);
    i.setType("message/rfc822");
    i.putExtra(Intent.EXTRA_EMAIL  , new String[]{mail});
    i.putExtra(Intent.EXTRA_SUBJECT, "Proposta modifica Workflow");
```

```
    i.putExtra(Intent.EXTRA_TEXT   , "");
    try {
        startActivity(Intent.createChooser(i, "Inviando la mail..."));
    } catch (android.content.ActivityNotFoundException ex) {

    }
}
```

Il metodo riportato sopra è un esempio di come poter inviare una mail attraverso un Intent (similarmente a quanto descritto nel caso si debba scegliere un file da caricare sul server). *sendMail(String mail)* accetta un parametro in input, il quale corrisponde alla mail di destinazione (mentre quella sorgente è la predefinita impostata nello smartphone). Lo stesso "popola" poi di informazioni come "oggetto" e "corpo" la mail, passando dei parametri con *putExtra()* come fosse un'activity. Sono numerosi i casi in cui questo è richiesto (e quindi molte le classi che implementano l'interfaccia che definisce *sendMail()*).

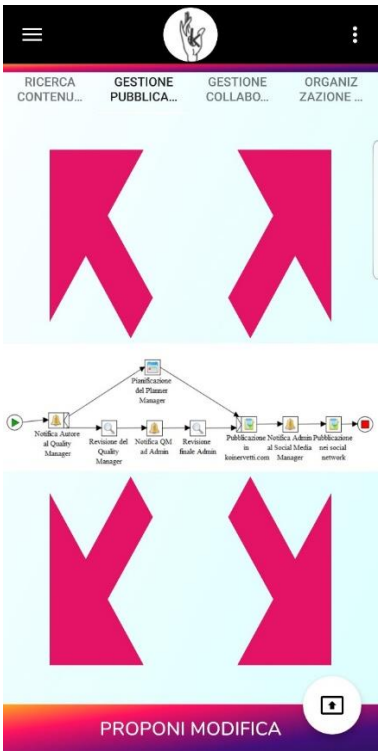

Figura 6.6.1 - Workflow di pubblicazione

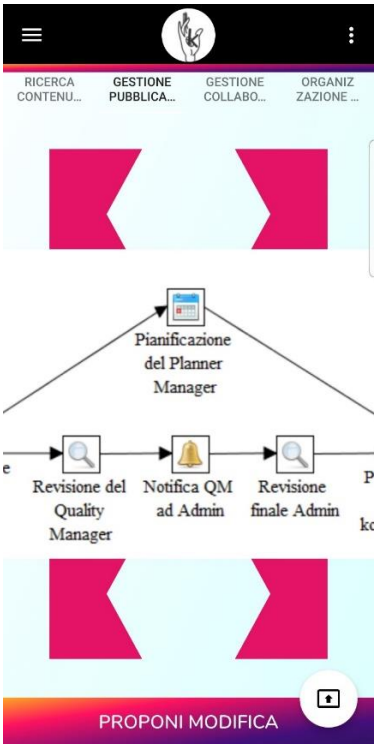

Figura 6.6.2 - Workflow di pubblicazione ingrandito

6.7 Notifiche

Nel contesto della comunicazione si denota la ricezione di notifiche (cloud Messaging). Firebase dispone di un'interfaccia grafica per la creazione di messaggi da inviare ad ogni dispositivo che abbia l'applicazione installata. E' possibile impostare nome, testo, icona, target (nonché tipologia di utenti alla quale inviare il messaggio) e programmare la cadenza dell'invio.

6.9 Script PHP e query SQL

L'interazione con il database MySQL ha, come già dichiarato, richiesto l'implementazione di script PHP che eseguissero dell'interrogazioni (query SQL).

Si apre una connessione con il database tramite mysqli_connect() ed una volta verificata, si esegue la query il cui risultato viene scorso e caricato su un array con *fetch_assoc()*. Il vettore risultante verrà poi incapsulato e codificato in JSON per poter essere utilizzato dall'applicazione Android.

7 Conclusioni

Il groupware è stato sviluppato parallelamente alle canoniche attività del sito web. La contemporaneità delle due attività ha facilitato la ricezione di feedback da parte dei collaboratori ed ha reso più stimolante progettazione ed implementazione considerando che l'applicazione sarà utilizzata realmente dal gruppo di lavoro.

Buona parte delle funzionalità sono state pensate lato amministratore (considerando come un coordinatore vorrebbe gestire un gruppo di caratteristiche similari), mentre l'aver assegnato dei privilegi ad alcuni responsabili è dipeso sia dalla necessità di redistribuire il carico di lavoro sia da esplicite richieste dei collaboratori stessi. Nel contesto medesimo, in futuro, l'obiettivo sarà quello di diversificare maggiormente i privilegi e di rendere autonomi i responsabili. Per la precisione, si conta di evolvere la gestione degli eventi da parte del manager addetto e di sincronizzare le attività con un calendario.

L’applicazione ha già subito una fase di test interno (equivalente ad una rapida distribuzione dell’applicazione ai collaboratori per controlli di qualità) per mezzo del quale sono emerse le prime necessità di miglioramento. In particolare, si è cercato di porre l’attenzione sull’utilizzo dei responsabili possedenti dei privilegi, in modo tale da poter verificare il soddisfacente funzionamento della cooperative working.

Il software è quindi destinato a continui aggiornamenti, i quali dipenderanno dall’evoluzione delle attività del sito web e dalle richieste dei collaboratori, con il costante obiettivo di massimizzare l’efficienza e l’efficacia della collaborazione.

Bibliografia

Koinervetti - https://www.koinervetti.com/

Wikipedia - https://it.wikipedia.org/

YAWL - http://www.yawlfoundation.org/

WordPress - https://wordpress.com/

Android Developer - https://developer.android.com/

Material Design - https://material.io/

Firebase - https://firebase.google.com/