



Dipartimento di Ingegneria

Laurea Triennale in Ingegneria Informatica

RICEZIONE E CONTROLLO DI DOCUMENTI

FOTOGRAFATI VIA WHATSAPP

Candidato

Vlad Turno (535396)

Relatore

Prof. Paolo Atzeni

Correlatore

Dott. Gherardo Vittoria

Anno Accademico 2021/2022

INDICE

INDICE	2
RINGRAZIAMENTI	4
INTRODUZIONE	5
1. APPLICAZIONI WEB	8
1.1 Progressive Web App: nozioni generali	9
1.2 Un caso specifico: l'infrastruttura di YayaMedia.....	12
2. LO SVILUPPO AGILE DI APPLICAZIONI.....	16
2.1 Obiettivi, valori e pratiche	16
2.2 Un framework specifico: lo Scrum	20
3. REQUISITI DEL MODULO	24
3.1 Analisi del problema	24
3.2 Requisiti funzionali.....	26
4. PROGETTAZIONE DEL MODULO	28
4.1 Architettura del modulo	28
4.2 Scelta delle API	30
5. REALIZZAZIONE DEL MODULO.....	34
5.1 Controllo di integrità delle acquisizioni.....	35
5.2 Controllo semantico delle firme.....	37
5.3 Estrapolazione dei codici di spedizione.....	39
5.3.1 Codici QR	40
5.3.2 Codici XAB	41
6. VERIFICHE FUNZIONALI	45
6.1 Il dataset oggetto di testing	46

6.2 Imprevisti e difficoltà incontrate	48
7. CONCLUSIONI	51
INDICE DELLE FIGURE	52
BIBLIOGRAFIA	53

RINGRAZIAMENTI

È doveroso rivolgere il primo tra tutti i ringraziamenti a *YayaMedia S.r.l.* per avermi reso partecipe allo sviluppo della piattaforma *Intergroup*. In particolar modo, vorrei ringraziare *Annalisa*, *Gherardo*, *Marco* e *Riccardo* sia per avermi dato fiducia che per avermi istruito e guidato durante questa esperienza.

Ringrazio inoltre il professor *Paolo Atzeni* per la disponibilità e i consigli che mi ha fornito durante questa particolare fase del mio percorso formativo.

Un ringraziamento speciale va rivolto a *Sonia* e alla mia famiglia, ai miei genitori *Dana* e *Claudio*, ai miei fratelli *Victor* e *Sophia*, ai miei zii *Oana* e *Mauro* e ai miei cuginetti *Gioia* e *Flavio* per avermi dato supporto durante questo tortuoso ma soddisfacente percorso.

Vorrei ringraziare particolarmente i miei colleghi *Jimmy*, *Matteo*, *Alessandro* e *Davide* per avermi più volte offerto il loro prezioso aiuto durante questi anni. Senza di voi probabilmente in questo momento non mi troverei qui a scrivere questa tesi.

Vorrei infine ringraziare *Edward Aloysius Murphy Jr.* per aver teorizzato e formulato l'omonimo assioma noto ai più come *Legge di Murphy*, poiché tale principio calza alla perfezione molte delle vicende che si susseguono nella mia vita universitaria da quattro anni a questa parte.

INTRODUZIONE

Sebbene risalga all'inizio degli anni Ottanta la commercializzazione del primo telefono cellulare sviluppato da *Motorola*, è senza dubbio grazie all'evoluzione subita nel corso dei successivi decenni che questo dispositivo ha assunto un ruolo preponderante nella vita quotidiana di molte persone, indipendentemente dallo scopo per cui viene adoperato.

Si stima infatti che nell'anno 2020 solo in Italia il numero degli smartphone risulta essere addirittura superiore a quello degli abitanti del paese, circa 80 milioni di dispositivi mobili a disposizione di poco più di 60 milioni di italiani.^[1]

Per quanto riguarda invece la situazione a livello mondiale, si stima che la percentuale di persone in possesso di tale dispositivo corrisponda, sempre per l'anno 2020, a circa il 78,05% della popolazione globale.

Può essere facilmente notato come tale fenomeno abbia subito una crescita vertiginosa nel corso degli ultimi anni, basta pensare che nel 2016 la percentuale di persone in possesso di un dispositivo mobile ammontava al 49,35% della popolazione.^[2]

È indubbiamente complice di tale diffusione l'integrazione in questi dispositivi, sempre più moderni e all'avanguardia, di molteplici funzionalità dalla varia natura.

Queste ultime trovano applicazione in un ampio ventaglio di ambiti, partendo da quello di carattere ludico come può essere il mondo dell'intrattenimento e arrivando a quello della produttività in situazioni di carattere lavorativo.

I dispositivi mobili moderni disponibili ad oggi sono, di fatto, dotati di risorse di calcolo spesso paragonabili a quelle possedute da molti computer, con le dovute limitazioni dettate da fattori quali le piccole dimensioni dell'apparecchio e la necessità di dover operare nei limiti dei consumi energetici imposti dalla batteria integrata nello stesso.

È senza dubbio sulla base di tali osservazioni che *Intergroup*, un'azienda che opera nel mondo della logistica dei trasporti, ha pensato di far sviluppare la propria piattaforma gestionale in maniera tale da rendere facile l'integrazione con i dispositivi cellulari.

In particolar modo tale integrazione si è resa necessaria dal momento in cui, come verrà spiegato nel corso della seguente tesi, alcuni degli operatori che adoperano il gestionale di *Intergroup* eseguono determinate operazioni per mezzo del proprio smartphone.

Per lo sviluppo della propria infrastruttura l'azienda ha quindi deciso di rivolgersi a *YayaMedia S.r.l.*, un'impresa che dal 2002 si occupa della realizzazione di soluzioni informatiche su misura.

Tali soluzioni intendono accogliere le sfide lanciate dall'avanzamento sempre più rapido e impetuoso della tecnologia.

Operando in questo senso, *YayaMedia* si pone come una vera e propria sartoria del software in grado di proporre soluzioni quanto più possibile dinamiche e flessibili, caratterizzate da una grande efficienza e scalabilità.^[3]

Per perseguire tale scopo, alla base degli applicativi sviluppati dall'azienda vi è sempre un approccio agile combinato all'adozione di tecnologie *web-based* volte ad ottenere una buona integrazione con i dispositivi mobili.

Nel corso di questa tesi verranno affrontati i vari aspetti inerenti allo sviluppo di questo tipo di applicazioni, facendo particolare riferimento al gestionale creato per *Intergroup* e in maniera più specifica a un modulo adibito alla ricezione e al controllo di documenti acquisiti tramite dispositivo mobile.

Il presente documento è articolato nel seguente modo: vi è una prima parte, di carattere introduttivo, composta da due capitoli volti ad illustrare varie tecnologie e aspetti metodologici che sono stati fondamentali durante il processo di sviluppo che ha caratterizzato il lavoro svolto dal candidato nell'ambito del tirocinio presso l'azienda.

In una seconda parte, composta dai capitoli finali, si entra nel vivo degli argomenti e viene illustrato il processo vero e proprio che ha portato allo sviluppo dell'applicativo.

Il primo capitolo è dedicato alle tecnologie web che sono alla base del software sviluppato dal candidato, con particolare riferimento alle applicazioni web progressive e all'infrastruttura di *YayaMedia*.

Il secondo capitolo è di carattere metodologico e affronta lo sviluppo agile di applicazioni, facendo particolare riferimento a *Scrum*, il framework adottato dall'azienda e i cui principi hanno guidato lo sviluppo dell'applicativo oggetto della tesi.

Nel terzo capitolo sono affrontati nel dettaglio i requisiti del modulo, mentre nel capitolo seguente vengono spiegate le scelte di progettazione alla base della realizzazione dello stesso.

Il quinto capitolo è invece dedicato alla vera e propria realizzazione del modulo, seguito da un ultimo capitolo in cui vengono affrontate le verifiche funzionali a cui è stato sottoposto il programma al termine della fase di sviluppo.

1. APPLICAZIONI WEB

In ambito informatico si indica con il termine “Applicazione Web” un programma applicativo memorizzato su un server remoto e pertanto fruibile tramite un qualsiasi dispositivo munito di un browser web (o di un qualsiasi altro programma dalle funzionalità analoghe) e di una connessione a una rete.

Le applicazioni web differiscono dalle applicazioni “tradizionali” per il fatto che implementano il paradigma *client-server*: il programma che viene eseguito non dipende quindi strettamente dalla piattaforma sulla quale viene installato ma vede l’interazione tra un client atto a richiedere servizi e un server atto a offrire servizi.

La diffusione di questo approccio è ampiamente motivata dai molteplici vantaggi che procura, primo tra tutti quello di poter apportare modifiche alla propria applicazione senza dover necessariamente aggiornare ognuno dei client che vi si interfacciano.

Si noti infatti come a differenza di un’applicazione nativa che necessita dello sviluppo di una versione differente per ogni sistema operativo su cui viene installata, un’applicazione web si appoggia su un *browser* che vede continui aggiornamenti.

Ciò solleva gli sviluppatori dall’onere di dover implementare tutte le varie distribuzioni e consente la stesura di un codice unico valido per tutti i dispositivi, peculiarità che le rende vantaggiose laddove non c’è bisogno di un accesso completo a determinate funzionalità dei dispositivi su cui l’applicazione deve operare.

In questa tesi viene affrontato un particolare tipo di applicazioni web, le *progressive web app*, ovvero una sorta di ibrido tra pagine web e applicazioni native nonché categoria alla quale appartengono il client di *Intergroup* e una buona parte degli applicativi sviluppati dall’azienda *YayaMedia*.

1.1 Progressive Web App: nozioni generali

Il termine *Progressive Web App*, spesso abbreviato in *PWA*, è stato coniato da Google e intende inquadrare quelle applicazioni che sono sviluppate e fruite come delle comuni pagine web ma che hanno la peculiarità di assumere caratteristiche simili a delle applicazioni native quando vengono visitate da smartphone o da altri dispositivi mobili.^[4]

Al fine di comprendere il funzionamento di tale ibrido è necessario innanzitutto sottolineare la differenza tra un sito web statico, come la maggior parte dei siti web presenti in rete, e un sito web dinamico, come può essere il client di un'applicazione in cui l'accesso ai servizi offerti avviene tramite *browser web*.

Un sito web si definisce statico quando il contenuto di tale sito rimane costante e può essere modificato solo manualmente da un moderatore, che in quanto tale ha accesso ai file presenti sul server. È il contenuto di tali file, tipicamente codice *HTML*, a determinare cosa vedrà il visitatore della pagina web in questione.

Risulta da subito evidente il fatto che tale approccio presenti determinate limitazioni.

Dobbiamo infatti considerare scenari in cui il contenuto della pagina web non deve essere statico, ma deve cambiare dinamicamente in funzione di determinate richieste effettuate dal visitatore del sito, che tipicamente è un utente e non un moderatore.

Ci sono molteplici situazioni in cui si profilano scenari di questo tipo, banalmente basti pensare a qualsiasi social network in cui ogni utente visualizza contenuti multimediali diversi in base ai propri interessi personali oppure a qualsiasi sito di commercio in cui i prodotti messi in vendita cambiano nel corso del tempo, gli esempi possibili sono vari e innumerevoli.

È impensabile che in questi casi possa esserci un moderatore che di volta in volta si occupa di creare e aggiornare le risorse richieste da ogni utente, è quindi necessario che certe operazioni vengano automatizzate e che il contenuto di determinate risorse sia determinato a tempo di esecuzione, che è esattamente ciò che caratterizza un sito web dinamico.

Nello sviluppo di siti con queste caratteristiche una soluzione obbligatoria è l'adozione di un'architettura multistrato in cui le varie funzionalità del programma sono divise su più livelli messi in comunicazione tra loro secondo il paradigma *client-server*.

Nel caso specifico delle applicazioni web è pratica comune adottare un'architettura a tre livelli, articolata in livello di presentazione, livello applicativo e livello di persistenza.

Il livello di presentazione è adibito alla comunicazione con l'utente. Ricopre infatti il ruolo di interfaccia dell'intera applicazione e consente sia la visualizzazione delle informazioni desiderate che la raccolta dei dati necessari all'elaborazione e alla produzione delle stesse.

Il livello applicativo o logico consiste in un componente o una serie di componenti che si occupano dell'elaborazione delle informazioni raccolte a livello di presentazione.

Per perseguire tale scopo spesso interagisce con il livello sottostante in termini di aggiornamento, modifica o eliminazione di dati e con il livello sovrastante per acquisire e fornire informazioni.

Gli altri due livelli, a differenza di quello applicativo, non sono quasi mai messi in comunicazione diretta tra di loro.

Il livello di persistenza corrisponde al punto in cui vengono archiviati e gestiti i dati e le informazioni elaborate dall'applicazione.

Una precisazione doverosa riguarda l'utilizzo e la presunta intercambiabilità dei termini "strato" e "livello". Quest'ultimo infatti fa riferimento a una suddivisione del programma effettuata sul piano puramente logico mentre il primo termine assume connotati funzionali.^[5]

Parlando in termini più concreti, lo strato di presentazione spesso consiste in una pagina o un sito web che viene sviluppato utilizzando generalmente strumenti informatici quali *HTML*, *CSS* e *JavaScript*.

Lo strato applicativo viene tipicamente sviluppato su un motore *Java*, *PHP*, *Python* o simili e comunica con lo strato di persistenza o con altri componenti esterni tramite chiamate *API* (acronimo di *Application Programming Interface*).

Infine, lo strato di persistenza consiste in un sistema di gestione di database (che sia esso relazionale o meno) come *PostgreSQL*, *MySQL*, *Microsoft SQL Server* e così via.

Il vantaggio principale di tale architettura consiste nel fatto che ognuno dei tre strati può essere eseguito su una propria infrastruttura, può essere sviluppato contemporaneamente agli altri e può essere aggiornato o scalato in base alle necessità senza effetti sugli altri strati, che risultano spesso separati fisicamente oltre che logicamente.

Tornando alle applicazioni web progressive, ci sono alcuni aspetti tecnici caratterizzanti che sono stati teorizzati dagli sviluppatori di *Google* e sono qui di seguito elencati.

Le PWA sono:

- *Progressive* in quanto il funzionamento è identico per ogni utente a prescindere dal browser adoperato
- *Responsive* in quanto si adattano alla risoluzione di qualsiasi schermo adoperato, sia esso mobile, desktop o altro
- *Offline* in quanto grazie all'interazione con il *Service Worker*, uno strumento del browser atto a salvare i dati di navigazione in memoria cache, l'utilizzo di determinate funzioni dell'applicazione è consentito in situazioni di connessione debole o assente^[6]
- *App-like* in quanto l'interazione e la navigazione su dispositivi mobili è molto simile a quella di un'applicazione nativa in termini di presentazione e gestualità
- *Secure* in quanto adoperano il protocollo *HTTPS*
- *Discoverable* in quanto possono essere trovate dai motori di ricerca
- *Linkable* in quanto facilmente condivisibili tramite *URL*
- *Installable* in quanto possono essere salvate da un utente come icona sulla schermata home del dispositivo

Inoltre, si deve al W3C (*World Wide Web Consortium*) l'introduzione, per ogni applicazione web progressiva, di un manifesto consistente di una specifica in formato *JSON* (acronimo di *JavaScript Object Notation*, formato adatto allo scambio di dati fra applicazioni *client-server*) in cui sono conservati i metadati associati a un'applicazione, come ad esempio il nome, l'icona, il link di collegamento e altre impostazioni varie ed eventuali.^[7]

Nonostante quella delle applicazioni web progressive possa sembrare una soluzione destinata ad un utilizzo sempre più largo da parte degli utenti grazie alle caratteristiche che ne fanno uno strumento all'avanguardia sotto molti punti di vista, tale tecnologia non ha vissuto periodo roseo nei primi anni che hanno seguito la nascita dei dispositivi mobili come li conosciamo oggi.

Nonostante fosse possibile da subito intravedere le potenzialità di questo strumento, le applicazioni native tenevano il primato in quanto a performance, usabilità e possibili funzionalità.

Tuttavia, da qualche anno a questa parte, complici l'avvento di *HTML5*, *CSS3* e il miglioramento subito dai *browser web* in generale, le applicazioni web progressive sono diventate una valida alternativa alla controparte nativa.^[8]

1.2 Un caso specifico: l'infrastruttura di YayaMedia

L'azienda *YayaMedia*, a cui si deve lo sviluppo del gestionale di *Intergroup* come applicazione web progressiva, implementa una piattaforma basata su *cloud computing*.

Questo significa che l'applicativo è sviluppato su un sistema distribuito in cui determinate risorse e servizi non sono fornite direttamente da *YayaMedia* ma bensì da terzi, come verrà illustrato nel seguito di questo capitolo.

Il nucleo della struttura è il server *ColdFusion*, una tecnologia multipiattaforma distribuita da *Adobe* che interpreta linguaggio *CFML* (acronimo di *ColdFusion Markup Language*) ed esegue codice *Java*.

Il CFML è un linguaggio di programmazione lato server che può essere eseguito su JVM (acronimo di *Java Virtual Machine*), in ambiente *.NET* (di proprietà di *Microsoft*) o su motore *Google App*. Il linguaggio CFML è fondamentalmente un *HTML* potenziato con comandi per eseguire operazioni su banche dati, costrutti per programmare secondo il paradigma orientato a oggetti, può generare codice *XML*, *HTML*, *CSS*, *JS* e vanta altri numerosi strumenti utili nell'ambito dello sviluppo web. Il *Java* invece non ha bisogno di presentazioni.^[9]

Lo strato di persistenza è implementato tramite il servizio di banca dati *MySQL* fornito dalla *Oracle*. L'interfaccia utente si articola in due diverse modalità: un *Front-End* basato su tecnologia *PWA* precedentemente trattata e un *Back-End* dedicato agli “addetti ai lavori” che ha la peculiarità di potersi interfacciare con la banca dati dell'applicazione.

La piattaforma è in hosting a Francoforte presso i server di *AWS* (acronimo di *Amazon Web Services*). Tale soluzione ha risvolti vantaggiosi in termini di affidabilità e scalabilità, in quanto all'occorrenza è possibile ampliare o ridurre le risorse computazionali da dedicare alla piattaforma.^[10]

Segue uno schema esemplificativo utile a inquadrare visivamente l'architettura base di *YayaMedia*.

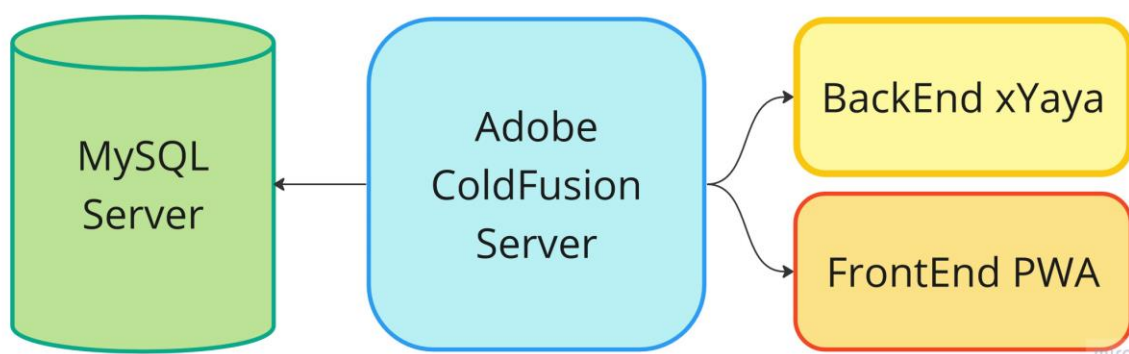


FIGURA 1 – IL SISTEMA YAYA

Nell'ambito del progetto *IGP* tale architettura è stata ampliata per far fronte alle esigenze specifiche dell'azienda che ha commissionato il progetto.

Nella fattispecie si è reso necessario attingere in tempo reale alla banca dati propria di *Intergroup*, motivo per il quale la banca dati *MySQL* che costituisce lo strato di persistenza dell'applicazione è stata collegata alla banca dati *MS-SQL Server* dell'azienda.

Tale collegamento utilizzava inizialmente una *VPN* crittografata ma tale scelta, che in un primo momento sembrava essere la più logica, è stata in seguito oggetto di ripensamento a causa di disservizi e attese nel rispondere alle richieste degli utenti.

A tal fine si è optato per la pianificazione di una routine adibita all'importo, a intervalli di tempo regolari, dei dati presenti nell'archivio esterno.

Gli accessi al server sono regolamentati da politiche di sicurezza e il backup dei dati viene effettuato regolarmente presso un sito differente (Dublino) per avere ridondanza geografica. L'archivio dati non presenta limiti di capienza.

L'accesso alle applicazioni di front-end e back-end avviene su un protocollo crittografato e previa autenticazione tramite credenziali rilasciate ai singoli utenti.

Sulla base delle credenziali assegnate al singolo utente e delle impostazioni stabilite dall'amministratore del sistema si determina quali funzionalità rendere disponibili o meno all'interno dell'applicazione.^[11]

La piattaforma, per far fronte a necessità funzionali dialoga con servizi esterni per mezzo di connessioni dirette (come nel caso dei server *FTP* per il trasferimento di dati) e chiamate *API*, come nel caso delle integrazioni con i vari servizi esterni.

A titolo di completezza segue uno schema esemplificativo utile a inquadrare visivamente l'architettura dell'applicativo *Intergroup*.

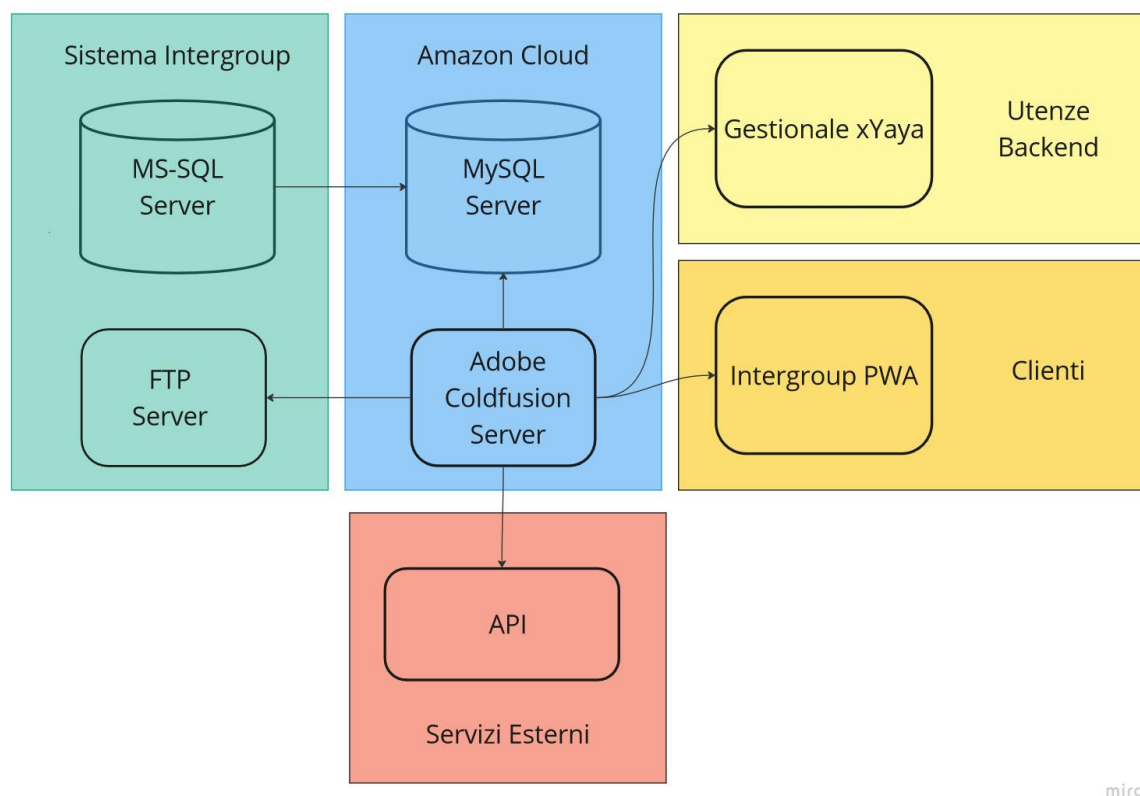


FIGURA 2 – IL SISTEMA IGP

2. LO SVILUPPO AGILE DI APPLICAZIONI

Si definisce con il termine “metodologia agile” un approccio allo sviluppo del software basato sulla distribuzione frequente e continua di parti funzionanti di programma realizzate in maniera rapida e iterativa.

Il termine, che racchiude in se un ampio ventaglio di metodologie e pratiche, fa la sua apparizione per la prima volta nel 2001 con la pubblicazione del “*Manifesto per lo Sviluppo Agile del Software*” (traduzione dall’inglese di “*Manifesto for Agile Software Development*”) ad opera di un gruppo di progettisti software tra i quali è bene citare *Kent Back* (mentore di pratiche come l’ *Extreme Programming* e il *test driven development*), *Ward Cunningham* (ideatore del concetto di Wiki ^[12]), *Robert Martin* e *Martin Fowler*.

Il concetto di sviluppo agile intende contrapporsi a quelli che all’epoca erano i tradizionali modelli di sviluppo detti metaforicamente “a cascata” e caratterizzati da una sequenzialità delle fasi da seguire rigidamente.

Tali metodologie di sviluppo sono state progressivamente abbandonate a causa degli effetti collaterali che sortivano, i quali verranno affrontati successivamente nel corso di questa tesi.

2.1 Obiettivi, valori e pratiche

La metodologia agile è concepita come una guida specifica per programmatori e progettisti che suggerisce un approccio meno rigido allo sviluppo in favore di uno focalizzato sull’obiettivo di produrre e consegnare frequentemente al cliente piccole parti di software funzionanti.

Nel *Manifesto Agile* sono esplicitamente indicati quattro valori ^[13] su cui si basa tale metodologia di sviluppo:

- Gli individui e le interazioni sono più importanti dei processi e degli strumenti
- È più importante avere un software funzionante che una documentazione esaustiva
- È bene preferire la negoziazione con il cliente piuttosto che la negoziazione di contratti
- È meglio rispondere in maniera adeguata ad un cambiamento piuttosto che seguire rigidamente un piano

L'obiettivo dei metodi agili è quello di eliminare il rischio di sviluppare programmi che non rispecchino quanto richiesto dal cliente e per perseguire tale scopo si avvale di una serie di pratiche. Alcune di queste sono di seguito descritte:

- Consegne frequenti di versioni incomplete ma funzionanti del software al fine di avere un riscontro su quanto realizzato e su eventuali anomalie o modifiche da apportare
- Coinvolgimento del cliente, possibile in misura differente a seconda dello specifico paradigma adottato
- Comunicazione costante tra tutti i soggetti che prendono parte allo sviluppo del progetto, cliente compreso, al fine di ottenere una migliore collaborazione tra le parti coinvolte
- Automatizzazione di attività che non riguardano strettamente lo sviluppo del programma, come ad esempio la produzione di documentazione
- Sviluppo iterativo tramite cicli di attività al fine di migliorare di volta in volta il prodotto finale
- Modellizzazione e rappresentazione visuale dei problemi e delle possibili soluzioni
- Lavoro di squadra, tipicamente fondamentale nelle varie pratiche di sviluppo agile
- Incontri frequenti, al fine di trasmettere costantemente a tutti gli attori che prendono parte alla realizzazione del progetto le informazioni necessarie alla buona riuscita dello stesso

- Definizione di priorità da assegnare a diverse parti (logiche o funzionali) del progetto
- Definizione di gerarchie all'interno del gruppo di sviluppo, laddove si palesa necessario avere una figura di riferimento per svolgere un determinato compito
- Produzione e l'aggiornamento di documentazione relativa al software in via di sviluppo
- Ristrutturazione nel corso del tempo di determinate parti di codice
- Semplicità e ordine in tutti gli aspetti inerenti allo sviluppo, dalla progettazione e modellazione alla produzione di documentazione
- Sviluppo guidato dai test, per definire quali caratteristiche dovrà assumere il programma ed evitare di andare fuori strada durante lo sviluppo dello stesso
- Suddivisione del progetto in intervalli di tempo definiti da eventi e scadenze
- Controllo, tramite appositi strumenti, delle versioni del prodotto rilasciate nel corso del tempo

Le pratiche messe in atto sono varie e diverse tra loro. Tra le molte è opportuno citare *extreme programming*, *feature driven development* o *Scrum*, che verrà affrontato in dettaglio nel corso del paragrafo seguente.

A titolo di completezza è doveroso citare le peculiarità dello sviluppo tradizionale detto “a cascata” al quale le metodologie agili intendono contrapporsi.

Tale approccio consiste nel seguire rigidamente una sequenza di passi, quasi come se ci si trovasse a dover affrontare un processo industriale.

Di larga diffusione nel corso degli anni '70, tale filosofia di sviluppo rimane una pietra miliare nell'ambito dei processi produttivi informatici e pur essendo stata ormai praticamente abbandonata conserva inalterata la sua rilevanza storica.^[14]

Il modello tradizionale per lo sviluppo sequenziale prevede una prima fase di analisi dei requisiti che ha lo scopo di determinare cosa dovrà fare e soprattutto cosa non dovrà fare il sistema che si intende realizzare. Tale fase può talvolta essere preceduta da uno studio di

fattibilità atto a stabilire costi, benefici e sostenibilità di un progetto prima ancora di decidere se intraprenderne o meno la realizzazione.

Il passo successivo consiste in una fase di progettazione il cui scopo è quello di determinare come verrà messo in atto ciò che è stato stabilito durante la precedente fase di analisi dei requisiti.

La progettazione è seguita dallo sviluppo vero e proprio del programma, che viene successivamente collaudato per verificarne il corretto funzionamento.

Chiude la cascata dei processi la fase di manutenzione, che comprende tutte le attività volte a mantenere un corretto funzionamento dell'applicativo.

Si noti come la sequenzialità di tali fasi non possa essere in alcun modo alterata in quanto ad ogni passo è necessario rifarsi a quanto fornito dal passo precedente. Ad esempio risulta poco fattibile collaudare un modulo che non si è ancora implementato, implementare un modulo che non si è ancora progettato, progettare un modulo che non si è ancora definito e così via.

L'approccio sequenziale allo sviluppo di applicazioni è corredato di pregi e difetti. Sebbene sia decisamente una buona pratica sviluppare un approccio regolamentato da istruzioni precise sui passi da seguire, questa pratica produce spesso effetti collaterali che minano la buona riuscita del progetto.

Primo tra tutti il fatto che qualsiasi errore che occorre in una determinata fase va inevitabilmente a compromettere l'integrità delle fasi successive.

Si prenda in esame la situazione in cui l'analisi dei requisiti presenta un errore di valutazione: tutto il processo di progettazione deve essere rivisitato e l'errore si propaga inesorabilmente sulle successive fasi di sviluppo e collaudo.

Tale paradigma di sviluppo presenta quindi una rigidità invalidante che, in concomitanza della frequenza con cui gli errori di valutazione occorrono durante lo sviluppo di un progetto, fa sì che il vantaggio vada naturalmente a favore dei più sofisticati paradigmi di sviluppo agile.

2.2 Un framework specifico: lo Scrum

Tra le varie metodologie di sviluppo agili che sono state teorizzate, una in particolare viene adottata dagli sviluppatori di *YayaMedia* durante il processo di realizzazione delle applicazioni: lo *Scrum*.

Il nome ha le sue radici in ambito sportivo e deriva dal nome che assume la mischia nello sport del rugby e intende riprendere la coesione con cui la squadra cerca di arrivare alla meta percorrendo il campo come un'unica entità.^[15]

Al di fuori della metafora sportiva si intende sottolineare come sia necessaria la coordinazione di tutti gli attori partecipanti nel muoversi verso la stessa direzione.

Il termine *Scrum* applicato allo sviluppo di prodotti appare per la prima volta nel 1986 in un articolo pubblicato su *Harvard Business Review* dai giapponesi *Hiroataka Takeuchi* e *Ikijuro Nonaka* intitolato “*The New New Development Game*”.

L'articolo prendeva in esame le pratiche di varie aziende manifatturiere, quali Honda, Toyota e 3M, e sottolineava come alla base dell'ottimizzazione dei processi produttivi ci fosse un approccio caratterizzato dalla sovrapposizione di varie fasi, al contrario di quanto avveniva sequenzialmente in altri processi di sviluppo industriali.

La prima applicazione di tale principio nel mondo dello sviluppo del software arriva qualche anno dopo quando *Ken Schwaber* e *Jeff Sutherland* presentano *Scrum* alla conferenza OOPSLA (acronimo di *Object Oriented Programming, Languages and Applications*) in Texas nel 1995.^[16]

Lo stesso *Sutherland* nella sua guida a *Scrum* indica che “*Scrum è un framework di processo utilizzato dai primi anni Novanta per gestire lo sviluppo di prodotti complessi. Scrum non è un processo o una tecnica per costruire prodotti, ma piuttosto è un framework all'interno del quale è possibile utilizzare vari processi e tecniche. Scrum rende chiara l'efficacia relativa del proprio product management e delle proprie pratiche di sviluppo così da poterle migliorare.*”^[17]

Scrum utilizza un approccio di tipo incrementale basato su controlli di tipo empirico. Nella fattispecie, l'empirismo è una corrente di pensiero secondo la quale la conoscenza deriva dall'esperienza pregressa e l'esperienza fa da padrona durante un processo decisionale.

Questo framework ha trovato terreno fertile nel mondo dello sviluppo di software in quanto risulta funzionale in tutti quei contesti laddove risulta difficile pianificare le cose in anticipo. Vengono di seguito analizzati i tre pilastri alla base di tale filosofia di sviluppo:

- Trasparenza di tutti gli aspetti significativi che riguardano la realizzazione del progetto, coadiuvata dalla definizione di standard comuni a tutti gli attori coinvolti;
- Ispezione frequente dei progressi realizzati e delle opere prodotte al fine di individuare tempestivamente eventuali problemi;
- Flessibilità e capacità di adattamento di fronte ad eventuali situazioni in cui si renda necessario cambiare qualcosa durante il processo produttivo.

Lo *Scrum* prevede la suddivisione del processo produttivo in unità indicate come *Sprint*, l'attribuzione di ruoli ben precisi agli attori coinvolti nel processo e l'istituzione di eventi volti ad analizzare e pianificare le varie iterazioni.

Lo *Sprint* è un'unità in cui viene suddiviso il processo di sviluppo ed ha una durata fissa che si attesta nell'ordine di grandezza delle settimane. Prima di ogni *sprint* si identificano gli obiettivi che devono essere raggiunti e si stimano i tempi necessari a perseguire tali obiettivi.

In seguito a tali considerazioni, durante lo *Sprint* non è più permesso cambiare obiettivi e qualsiasi modifica o necessità sarà presa in considerazione soltanto nello *Sprint* successivo.

Al termine di ogni *sprint* viene fornita una versione completa o parzialmente completa del prodotto che implementa gli avanzamenti decisi durante la pianificazione precedentemente avvenuta, indicati con il termine "*Sprint Goals*".

I ruoli principali che possono essere ricoperti all'interno del framework *Scrum* sono il *Team di Sviluppo*, lo *Scrum Master*, il *Product Owner* e lo *Stakeholder* (o gli *Stakeholders*).

Il *Team di sviluppo* è l'artefice delle attività di produzione ed è composto da un numero di persone compreso tra 3 e 9. È stato dimostrato empiricamente come qualsiasi numero di componenti non appartenente a questo intervallo determini un decremento in termini di efficienza del gruppo stesso.

I componenti devono avere le competenze funzionali necessarie a realizzare il lavoro e non è raro che ogni membro del Team possa eccellere in alcune competenze specifiche.

Lo *Scrum Master* è una figura adibita al controllo e alla facilitazione del lavoro eseguito dal *Team di Sviluppo*. Sebbene quello dello *Scrum Master* sia un ruolo di dirigenza, egli non è un vero e proprio leader ma più un supervisore che si assicura che gli sviluppatori possano lavorare senza complicazioni o influenze esterne.

Il *Product Owner* è la figura che rappresenta il cliente o i clienti per cui si sta eseguendo il lavoro. Si occupa di definire i requisiti e le priorità del prodotto. Tale figura non deve corrispondere o essere confusa con quella dello *Scrum Master*.

Infine, lo *Stakeholder* o gli *Stakeholders* corrispondono al cliente o ai clienti finali per cui si sta sviluppando il prodotto. Nel caso particolare della produzione di software, non necessariamente coincidono con l'utenza finale che andrà ad utilizzare il programma sviluppato.

Il framework *Scrum* prevede l'istituzione di determinati eventi che vogliono consentire un'ispezione del processo produttivo in corso.

Ad esempio, lo *Sprint Planning* è la riunione che precede ogni *Sprint* e in cui si pianifica l'attività da svolgere e gli obiettivi che devono essere raggiunti, mentre il *Daily Scrum* è una riunione giornaliera in cui il *Team di Sviluppo* viene aggiornato e si confronta sui vari aspetti inerenti al lavoro finora svolto o ancora da svolgere. O ancora la *Sprint Review* è un incontro retrospettivo che si tiene al termine dello *sprint* e in cui si ispeziona quanto è stato fatto o cosa deve essere aggiustato nel corso delle successive iterazioni.

Ovviamente il framework di *Scrum* fornisce delle linee guida che non sempre è possibile seguire con rigore assoluto.

Ad esempio nel contesto di *YayaMedia* gli *Sprint* hanno una durata temporale variabile che è talvolta inferiore alla settimana, mentre lo *Sprint Planning* è spesso sostituito da una riunione con cadenza giornaliera che assume caratteristiche più simili a quelle di un *Daily Scrum*.

Oppure ancora il *Product Owner* che si interfaccia con i clienti è talvolta una figura interna al *Team di Sviluppo* e spesso corrisponde con lo *Scrum Master*, ma queste scelte sono sempre dettate da condizioni di circostanza, come ad esempio il fatto che il numero di persone che forma il *Team di Sviluppo* è inferiore al numero dei ruoli che devono essere globalmente ricoperti.

Pertanto questa rivisitazione delle linee guida di *Scrum* non può essere sempre considerata un difetto ma, anzi, si allinea in maniera coerente allo spirito adattivo che la metodologia in questione suggerisce di assumere.

3. REQUISITI DEL MODULO

In questo capitolo si intende riportare un'analisi più dettagliata del contesto in cui la realizzazione del modulo oggetto di questa tesi si inserisce e dei requisiti funzionali, di carattere tecnico e non, che devono essere perseguiti al fine di ottenere una porzione funzionante del progetto finale.

Intergroup è un'azienda che dal 1986 opera nel mondo della logistica e in particolare gestisce lo stoccaggio e il trasporto per via terrestre e navale di merci e materie prime.

A partire dal 2019 *Intergroup* ha deciso di collaborare con *YayaMedia* alla realizzazione di una propria *WebApp* al fine di digitalizzare alcune attività, come ad esempio la visualizzazione in tempo reale dello stato dei magazzini o le operazioni relative all'inserimento di ordini e via dicendo.

Il fatto di avere investito nella realizzazione di una *Progressive Web App*, tecnologia di cui sono stati analizzati gli aspetti nel primo capitolo di questa tesi, ha fatto guadagnare a *Intergroup* l'assegnazione (per ben due volte nel 2019 e nel 2021) del premio “*Il logistico dell'anno*”, un premio all'innovazione in ambito tecnologico e multimediale che da 15 anni *Assologistica* (un'associazione confindustriale di operatori della logistica in conto di terzi) assegna a manager e imprese che considera particolarmente innovativi.^[18]

3.1 Analisi del problema

Per le attività che riguardano lo stoccaggio di merci in magazzino e il conseguente trasporto per via terrestre delle stesse, *Intergroup* si affida spesso a dei corrieri che non sono interni all'azienda e talvolta collaborano solo occasionalmente con quest'ultima.

Diventa quindi complicato, vista la mole di spedizioni che vengono effettuate giornalmente, creare per ognuno dei corrieri un'utenza che consenta di accedere alla piattaforma *IGP*, al netto del fatto che questa è stata pensata e realizzata come *PWA* proprio per consentire un

utilizzo immediato dei servizi che sono forniti tramite *browser web*, strumento alla portata di chiunque abbia a disposizione un cellulare fabbricato dopo il 2007 e una connessione a una rete mobile.

Si porta all'attenzione del lettore il fatto che non è la creazione dell'utenza l'unico ostacolo da scavalcare, in quanto ogni persona che va ad adoperare la piattaforma deve essere anche istruita ad un corretto utilizzo della stessa e non sempre questo passaggio risulta facile ed immediato come sembra.

Nella fattispecie il compito dei corrieri è quello di prendere in carico una spedizione, come può essere quella di un lotto di materiale che deve essere spostata da un magazzino a un altro.

Al momento della presa in carico al corriere viene consegnata una bolla di spedizione, ovvero un foglio dove sono riepilogati i dati relativi al lotto e via dicendo, che dovrà essere firmata e consegnata al termine della spedizione in questione.

Al termine di questa operazione il sistema informatico di *Intergroup* deve poter tenere traccia dell'avvenuta spedizione e soprattutto dei dati ad essa relativi, e viste le condizioni sopra citate (che portano i corrieri a non poter usufruire direttamente della piattaforma *IGP*) si è pensato di arginare il problema avvalendosi di uno degli strumenti di messaggistica più diffusi, usato da circa il 30% della popolazione mondiale: *WhatsApp*.

Grazie all'istituzione di un numero *WhatsApp Business* e all'integrazione con *Twilio*, uno strumento che offre interfacce *API* per l'integrazione con i vari servizi di messaggistica presenti sul mercato, ai corrieri che devono consegnare la bolla di spedizione basta semplicemente fotografare in maniera appropriata il documento inviandolo tramite messaggio e l'applicazione sviluppata per *IGP* si occuperà di tutto il resto.

In maniera fedele a quelli che sono i principi dello sviluppo agile e iterativo di *Scrum*, nell'applicazione era già presente una prima versione, operativa e funzionante, di un componente atto a raccogliere e processare le foto delle bolle di spedizione inviate dai corrieri.

È tuttavia necessario sviluppare una versione ampliata di tale componente, in quanto durante un'analisi dei requisiti successiva alla realizzazione e alla consegna di quest'ultimo è emersa la necessità di processare tipi differenti di bolle di spedizione che non sono provvisti di codice QR bidimensionale.

È inoltre necessario inserire controlli sull'integrità dei documenti fotografati.

Si porta all'attenzione del lettore il fatto che questo è esattamente uno degli scenari in cui risulta vantaggiosa la scelta di adottare una metodologia agile per lo sviluppo del software.

3.2 Requisiti funzionali

In questo paragrafo si passa all'analisi delle funzionalità che il componente implementa in una prima versione, per poi continuare con l'esaminazione delle funzionalità che è necessario integrare in una seconda versione.

Allo stato attuale, le bolle di spedizione inviate dai corrieri via *WhatsApp* al numero fornito da *IGP* contengono, oltre a dei campi testuali in cui sono indicate tutte le informazioni relative al lotto della merce e alla spedizione, un codice QR che se decodificato riporta il codice numerico della spedizione presa in esame.

Le bolle di spedizione sono infatti generate da un altro componente dell'applicativo *Intergroup*, le cui caratteristiche e funzionalità non verranno riportate all'interno di questa tesi per necessità di semplificazione e per il fatto che non sono oggetto del lavoro svolto dal candidato che ha redatto questo documento.

Quando un corriere invia la foto di un documento di spedizione, questa viene elaborata grazie al codice QR presente sul documento e i dati relativi alla spedizione e alle giacenze vengono aggiornati.

Il programma, quindi, compila un messaggio di avvenuta ricezione e corretta elaborazione del documento da recapitare via *WhatsApp* al corriere al termine del processo.

L'ampliamento del modulo prevede che il sistema sia dotato di nuove funzionalità in grado di adempiere ai seguenti compiti:

- Controllare che il documento cartaceo, fotografato, sia stato inquadrato per intero
- Controllare che il documento cartaceo sia stato firmato dal corriere
- Deve essere resa possibile l'elaborazione di documenti che non presentano il numero di spedizione sottoforma di codice *QR* ma sottoforma di stringa alfanumerica;

È necessario sviluppare il modulo in maniera conforme a tutte quelle che sono le scelte tecnologiche adottate dall'azienda *YayaMedia*, ovvero l'utilizzo di *ColdFusion* e del linguaggio *CFML* per la scrittura del codice, l'utilizzo di *MySQL* per le interrogazioni sulla banca dati e il mantenimento della compatibilità con tutti gli elementi dell'infrastruttura *IGP/Yaya* come è illustrata al capitolo 1.2 di questa tesi.

4. PROGETTAZIONE DEL MODULO

In questo capitolo viene illustrata la progettazione del componente. In particolare, nella prima parte viene illustrata l'architettura adottata corredata di tutte le interazioni tra i vari servizi esterni, mentre nella seconda parte segue un'illustrazione delle scelte progettuali e dei servizi esterni di cui il programma si avvale.

4.1 Architettura del modulo

Per rappresentare visivamente e in maniera intuitiva l'architettura del modulo segue un diagramma delle interazioni tra i vari componenti.

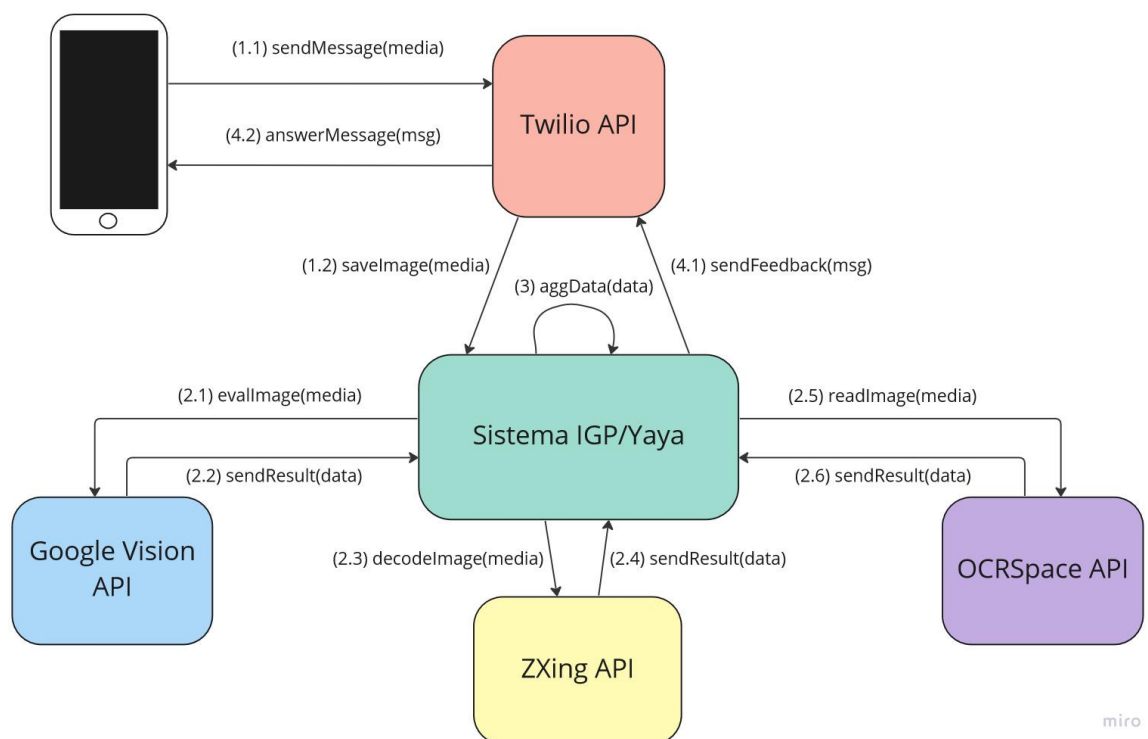


FIGURA 3 – DIAGRAMMA DI INTERAZIONE DEL SISTEMA

La figura fa riferimento all'unico caso d'uso che può interessare il sistema, ovvero l'inoltro della foto di una bolla di spedizione via *WhatsApp*.

Nella prima operazione (1.1) il protagonista è il corriere, che fotografa e invia tramite *WhatsApp* il documento di spedizione che è stato firmato a consegna avvenuta.

Subito dopo (1.2) *Twilio* si occupa di estrapolare il file multimediale dal messaggio e aggiungerlo alla banca dati del sistema *IGP/Yaya*.

Si noti come la comunicazione tra il modulo e i servizi esterni avvenga tramite i protocolli delle chiamate *API*. L'illustrazione delle motivazioni dietro la scelta dei servizi esterni è oggetto del prossimo paragrafo di questo capitolo.

È compito del sistema, nelle operazioni contrassegnate dal prefisso 2, eseguire una serie di controlli sul file multimediale appena ricevuto.

Si possono individuare tre fasi che scandiscono tale gruppo di operazioni: in una prima fase l'immagine raccolta viene inoltrata ai servizi di *Google Vision*, che si occuperà di individuare elementi all'interno della foto e valutare alcuni parametri come vedremo nel corso del capitolo seguente.

In una seconda fase fanno da protagonisti i servizi di *ZXing*, responsabili della decodifica di eventuali codici *QR* bidimensionali presenti all'interno dell'immagine.

Nell'ultima fase di queste operazioni il file multimediale viene eventualmente sottoposto al controllo di *OCRSpace*, che qualora necessario saranno volti ad estrapolare tutto il contenuto testuale, sottoforma di stringa, dalla fotografia della bolla di spedizione.

Nell'operazione numero 3 assistiamo ad una elaborazione delle informazioni raccolte dai servizi esterni, seguita da un aggiornamento della banca dati durante il quale tutte le informazioni relative alla spedizione e alle giacenze vengono aggiornate.

Infine, nel corso dell'operazione numero 4 assistiamo alla compilazione di un messaggio riepilogativo che deve essere recapitato al corriere. Tale messaggio viene dapprima inoltrato a *Twilio* (4.1) che si occuperà di inoltrarlo via *WhatsApp* al mittente (4.2).

4.2 Scelta delle API

Una parte importante del modulo consiste nell'interazione con servizi esterni che offrono una serie di strumenti atti a eseguire la ricezione e il controllo delle foto dei documenti.

Le soluzioni adottate a tale scopo sono le seguenti:

- *Twilio API*
- *ZXing API*
- *Google Cloud Vision API*
- *OCRSpace API*

Di queste, le prime due sono state scelte al momento della realizzazione della prima versione del modulo. Vista la buona resa delle stesse si è deciso di mantenere l'integrazione con tali strumenti, che rimangono operativi in maniera alterata nella seconda versione del modulo.

Twilio è una piattaforma semplice e potente che offre servizi estremamente flessibili e pienamente programmabili che consentono l'interazione con una molteplicità di canali di comunicazione quali chiamate, SMS, messaggistica *WhatsApp*, posta elettronica e altro.

I servizi di *Twilio* seguono una tariffa determinata in base al consumo. Anche questa è una delle caratteristiche che rendono la piattaforma una scelta ottima per la realizzazione del componente preso in esame in questa tesi.^[19]

ZXing è una piattaforma open-source implementata in *Java*, ma aperta all'integrazione con numerosi altri linguaggi di programmazione, che consente il processamento di immagini contenenti codici a barre unidimensionali (come sono i normali codici a barre che possono essere trovati sulle etichette dei prodotti in vendita presso un negozio) e bidimensionali (come sono i codici QR presenti sulle bolle di spedizione).^[20]

Tali caratteristiche rendono *ZXing* uno strumento fondamentale per il funzionamento del programma, che ha dimostrato forte stabilità e un alto livello di affidabilità già nella prima implementazione del modulo.

Si è quindi scelto di mantenere inalterata l'integrazione con tale strumento anche nella successiva versione del programma.

Google Cloud Vision API è un particolare strumento che permette agli sviluppatori di analizzare il contenuto delle immagini sfruttando modelli di machine learning in continua evoluzione.^[21]

Gli strumenti offerti da questa piattaforma consentono di rilevare oggetti ed elementi all'interno di un'immagine, compresi quindi testi o particolari sezioni di testo stampato o scritto a mano, corredati da coordinate per ogni elemento e vari metadati derivanti dall'analisi del contenuto multimediale.

Queste caratteristiche rendono *Google Vision* uno strumento sicuramente interessante di cui avvalersi al fine di ottenere un corretto funzionamento del programma.

Nella pagina seguente sono riportate delle parti della risposta che lo strumento fornisce in seguito all'analisi di un'immagine, in questo caso proprio una delle fotografie dei documenti di spedizione.



FIGURA 4 – ANALISI DI GOOGLE VISION

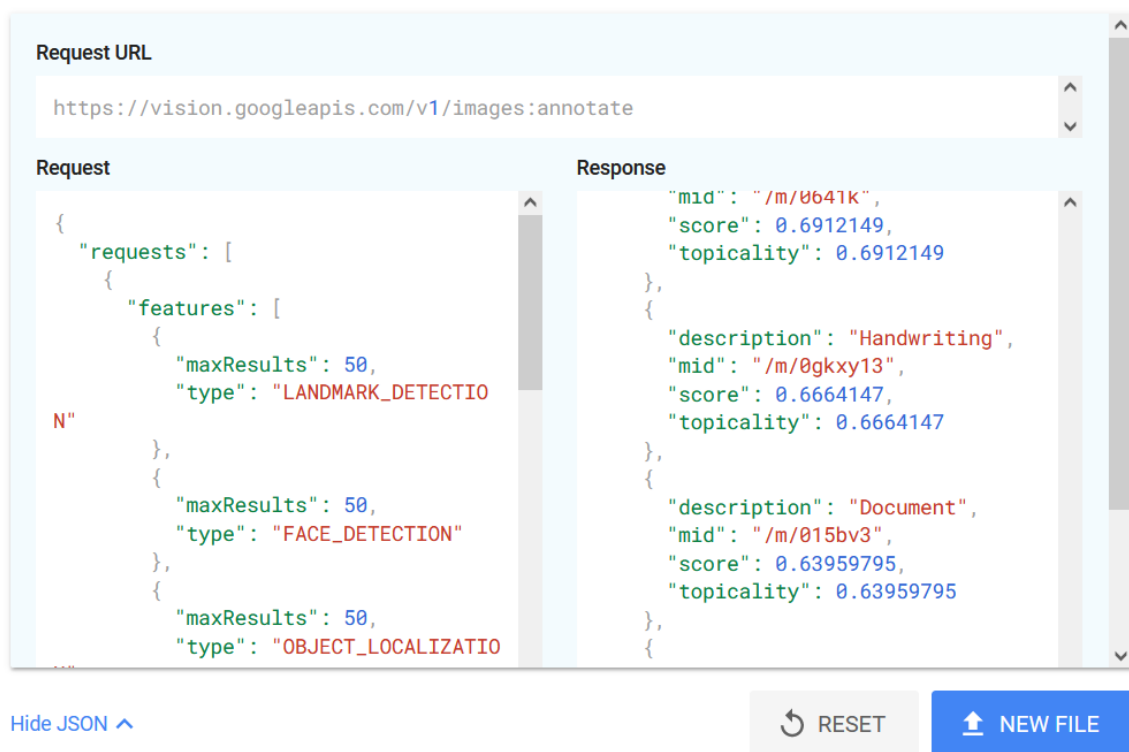


FIGURA 5 – RISPOSTA IN FORMATO JSON

L'ultimo strumento che si è deciso di adottare è *OCRSpace*, un *API* che consente di estrarre il contenuto testuale di documenti e foto e restituirlo in formato *JSON*. Nel piano di utilizzo gratuito, *OCRSpace* consente l'invio di 500 richieste giornaliere, che salgono a 10 000 nell'offerta a pagamento.^[22]


Verrà resa nota ed esplicata nei capitoli seguenti la necessità di avere questo strumento come risorsa ausiliaria a *Google Vision*, che al pari di *OCRSpace* offre già la possibilità di estrapolare contenuti testuali da una foto.

Qui di seguito viene riportata la risposta che lo strumento fornisce quando riceve come input un'immagine, in questo caso proprio una delle fotografie dei documenti di spedizione.

Parsed Successfully! All images / pages were parsed successfully. (Processing time: 2.843 seconds) ✕

Download Searchable PDF: Searchable PDF not generated as it was not requested. ✕

Image Preview



File loaded successfully. Note: TIFF files are not displayed, but OCR works

Download

Show Overlay

OCR'ed Result

Text	Json
<pre> "Width": 65.5 } }, "MaxHeight": 17.5, "MinTop": 1214.5 }, { "LineText": "INTERLOG SRL Lun Caboto, 110 04024 GAETA (LT) IT", "Words": [{ "WordText": "INTERLOG", "Left": 471. </pre>	

FIGURA 6 – ANALISI DI OCRSPACE

5. REALIZZAZIONE DEL MODULO

In questo capitolo vengono illustrati nel dettaglio i processi che implementano le funzionalità richieste nei requisiti funzionali del modulo (capitolo 3.2).

Il primo passaggio che viene eseguito da questa porzione di programma consiste nell'andare ad individuare il file multimediale che è stato raccolto dai servizi di *Twilio* e riposto in una specifica tabella all'interno della banca dati del sistema.

Tale operazione è temporizzata e va ad eseguire la scansione dei media in arrivo ogni due minuti circa, si può quindi parlare di una vera e propria coda, con priorità di tipo *FIFO* (acronimo di *first in first out*) assegnata ai media in arrivo.

Si pone all'attenzione del lettore il fatto che le operazioni di ricezione dei media, inserimento in coda e analisi temporizzata della coda non verranno affrontate nel dettaglio durante il corso di questa tesi in quanto la loro realizzazione è antecedente al lavoro svolto dal candidato nell'ambito del tirocinio.

Viene dunque affrontato, per ogni immagine raccolta, il controllo e il processamento vero e proprio del file multimediale.

Il primo passaggio consiste nel far processare ai vari servizi l'immagine e per ogni servizio viene raccolta in una variabile la risposta contenente tutti i dati necessari ad elaborare l'ordine.

Il primo servizio al quale viene sottoposta l'immagine è *Google vision*. La risposta fornita dai servizi di *Google Vision* è estremamente vasta, basti pensare che per ogni immagine vengono forniti metadati di ogni tipo, come la palette dei colori dominanti all'interno della stessa, e per ognuno degli elementi individuati vengono indicati dati come le coordinate dei vertici e così via.

Dal momento in cui il documento in formato *JSON* fornito in risposta all'analisi presenta tipicamente diverse migliaia di nodi e ha una lunghezza complessiva appartenente all'ordine di grandezza delle decine di migliaia di righe, si è reso necessario l'implementazione di una funzione ausiliaria che estrapola solo e soltanto i nodi di interesse ai fini del controllo del

documento e istituisce una seconda struttura più snella che viene poi utilizzata dalla funzione principale. Tali operazioni sono facilitate grazie all'utilizzo di funzioni come *StructFindValue* e *StructInsert* offerte dal framework *ColdFusion* e riportate nella seguente figura.

```
3580
3581      <!--- /// QRCODE /// --->
3582      <cfset qrcode = StructFindValue(sResponse,"2D barcode","one")>
3583      <cfscript>
3584      |     StructInsert(sReturn.DATA.QRCODE,"2D barcode",qrcode);
3585      </cfscript>
3586
```

FIGURA 7 –RICERCA DI NODI IN UNA STRUTTURA

5.1 Controllo di integrità delle acquisizioni

Si entra quindi nel vivo delle operazioni partendo dal controllo di integrità delle acquisizioni. In particolar modo si intende controllare che il documento, durante il processo di acquisizione fotografica, sia stato inquadrato per intero.

Al fine di eseguire tale controllo è stato elaborato un sistema che si avvale dell'utilizzo di marcatori da disporre ai quattro angoli del foglio.

Durante il processo di sviluppo dell'applicativo si è scelto di usare come marcatori quattro caratteri speciali, nella fattispecie \$1, \$2, \$3 e \$4.

Per quello che riguarda invece l'utilizzo reale di questa pratica, sarà compito del personale di *Intergroup* decidere se mantenere questi caratteri speciali come marcatori, e quindi inserirli nel modello ufficiale delle bolle di spedizione, o sostituirli con altri elementi quali possono essere caratteri differenti, simboli o piccoli codici a barre.

Allo stato attuale il modulo è tarato per riconoscere i caratteri speciali sopra citati, ma può essere facilmente rifattorizzato per consentire la ricerca di qualsiasi alternativa sfruttando il medesimo principio.

Nella figura di seguito è riportata, a titolo illustrativo, una parte della struttura fornita da una funzione ausiliaria che esegue una scrematura dei risultati forniti dall'analisi operata da *Google Vision API*.

struct																															
DATA	struct																														
CCHECK	0																														
CCORNERS	<table> <tr> <th colspan="2">struct</th></tr> <tr> <td>C1</td><td> <table> <tr> <th colspan="2">array</th></tr> <tr> <td>1</td><td> <table> <tr> <th colspan="2">struct</th></tr> <tr> <td>x</td><td>0.3219996</td></tr> <tr> <td>y</td><td>0.10340404</td></tr> </table> </td></tr> </table> </td></tr> <tr> <td>C2</td><td> <table> <tr> <th colspan="2">array</th></tr> <tr> <td>1</td><td> <table> <tr> <th colspan="2">struct</th></tr> <tr> <td>x</td><td>0.2589006</td></tr> <tr> <td>y</td><td>0.20212424</td></tr> </table> </td></tr> </table> </td></tr> <tr> <td>C3</td><td>array [empty]</td></tr> <tr> <td>C4</td><td>array [empty]</td></tr> </table>	struct		C1	<table> <tr> <th colspan="2">array</th></tr> <tr> <td>1</td><td> <table> <tr> <th colspan="2">struct</th></tr> <tr> <td>x</td><td>0.3219996</td></tr> <tr> <td>y</td><td>0.10340404</td></tr> </table> </td></tr> </table>	array		1	<table> <tr> <th colspan="2">struct</th></tr> <tr> <td>x</td><td>0.3219996</td></tr> <tr> <td>y</td><td>0.10340404</td></tr> </table>	struct		x	0.3219996	y	0.10340404	C2	<table> <tr> <th colspan="2">array</th></tr> <tr> <td>1</td><td> <table> <tr> <th colspan="2">struct</th></tr> <tr> <td>x</td><td>0.2589006</td></tr> <tr> <td>y</td><td>0.20212424</td></tr> </table> </td></tr> </table>	array		1	<table> <tr> <th colspan="2">struct</th></tr> <tr> <td>x</td><td>0.2589006</td></tr> <tr> <td>y</td><td>0.20212424</td></tr> </table>	struct		x	0.2589006	y	0.20212424	C3	array [empty]	C4	array [empty]
struct																															
C1	<table> <tr> <th colspan="2">array</th></tr> <tr> <td>1</td><td> <table> <tr> <th colspan="2">struct</th></tr> <tr> <td>x</td><td>0.3219996</td></tr> <tr> <td>y</td><td>0.10340404</td></tr> </table> </td></tr> </table>	array		1	<table> <tr> <th colspan="2">struct</th></tr> <tr> <td>x</td><td>0.3219996</td></tr> <tr> <td>y</td><td>0.10340404</td></tr> </table>	struct		x	0.3219996	y	0.10340404																				
array																															
1	<table> <tr> <th colspan="2">struct</th></tr> <tr> <td>x</td><td>0.3219996</td></tr> <tr> <td>y</td><td>0.10340404</td></tr> </table>	struct		x	0.3219996	y	0.10340404																								
struct																															
x	0.3219996																														
y	0.10340404																														
C2	<table> <tr> <th colspan="2">array</th></tr> <tr> <td>1</td><td> <table> <tr> <th colspan="2">struct</th></tr> <tr> <td>x</td><td>0.2589006</td></tr> <tr> <td>y</td><td>0.20212424</td></tr> </table> </td></tr> </table>	array		1	<table> <tr> <th colspan="2">struct</th></tr> <tr> <td>x</td><td>0.2589006</td></tr> <tr> <td>y</td><td>0.20212424</td></tr> </table>	struct		x	0.2589006	y	0.20212424																				
array																															
1	<table> <tr> <th colspan="2">struct</th></tr> <tr> <td>x</td><td>0.2589006</td></tr> <tr> <td>y</td><td>0.20212424</td></tr> </table>	struct		x	0.2589006	y	0.20212424																								
struct																															
x	0.2589006																														
y	0.20212424																														
C3	array [empty]																														
C4	array [empty]																														
CMISSING	<table> <tr> <th colspan="2">array</th></tr> <tr> <td>1</td><td>c3</td></tr> <tr> <td>2</td><td>c4</td></tr> </table>	array		1	c3	2	c4																								
array																															
1	c3																														
2	c4																														
NFOUND	1																														
QRCHECK	1																														
QRCODE	struct																														

FIGURA 8 – INDIVIDUAZIONE DI MARCATORI

Si noti come la struttura dati è organizzata in modo tale da fornire in un primo nodo *CCHECK* il risultato della valutazione, dove il campo assume valore pari a 1 se ho trovato almeno 3 marcatori di angolo e assume valore pari a 0 se non ho trovato abbastanza marcatori.

Nel nodo *CCORNERS* viene invece inserito il dettaglio di ognuno dei marcatori individuati, mentre nel nodo *CMISSING* è fornito un array contenente i marcatori che invece non sono stati individuati.

5.2 Controllo semantico delle firme

Sfruttando la stessa funzione ausiliaria di prima diventa facile ottenere l'accesso alle informazioni necessarie a controllare che il documento di spedizione sia stato firmato.

Infatti, tra le tante informazioni fornite dalla diagnostica di *Google Vision* (tra qui la presenza di loghi, il font utilizzato e così via dicendo) ce n'è una in particolare utile a tale scopo: si tratta del nodo denominato "*Handwriting*".

Tale nodo non risulta essere presente quando l'algoritmo di *Google Vision* non individua alcuna porzione di testo che ritiene essere stata scritta a mano, mentre è presente ed è corredato di un campo "*Score*" quando invece porzioni di testo identificate come scritte a mano sono individuate.

Il campo *Score* può assumere un valore decimale compreso tra 0 e 1, dove il punteggio pieno pari a 1 viene assegnato quando l'algoritmo di *Google* è sicuro al 100% che la porzione di testo individuata è stata scritta manualmente da un essere umano.

Si è deciso implementare il meccanismo di controllo nel seguente modo: quando il nodo *Handwriting* è presente ed ha uno *score* maggiore o uguale a 0.5, allora il documento è da considerarsi come firmato. Se tale nodo non è presente o lo *Score* assume un valore inferiore al valore scelto come soglia, il documento non può considerarsi firmato e bisogna notificarlo al mittente.

A titolo di completezza è riportata nell'immagine sottostante la porzione di struttura dati che accoglie i valori relativi all'apposizione di firme, dove il campo *SCHECK* è il risultato della valutazione delle stesse e può assumere un valore pari a 0 o 1.

						<table><tr><td colspan="2">struct</td></tr><tr><td>x</td><td>0.45057395</td></tr><tr><td>y</td><td>0.19911206</td></tr></table>	struct		x	0.45057395	y	0.19911206
struct												
x	0.45057395											
y	0.19911206											
					4	<table><tr><td colspan="2">struct</td></tr><tr><td>x</td><td>0.3219996</td></tr><tr><td>y</td><td>0.19911206</td></tr></table>	struct		x	0.3219996	y	0.19911206
struct												
x	0.3219996											
y	0.19911206											
			mid	/j/863wbw								
			name	2D barcode								
			score	0.9755927								
		path	.DATA.result.responses[1].localizedObjectAnnotations[1].name									

SCHECK	1																										
SIGNATURE	<table><tr><td colspan="2">struct</td></tr><tr><td>Handwriting</td><td><table><tr><td colspan="2">array</td></tr><tr><td>1</td><td><table><tr><td colspan="2">struct</td></tr><tr><td>key</td><td>description</td></tr><tr><td>owner</td><td><table><tr><td colspan="2">struct</td></tr><tr><td>description</td><td>Handwriting</td></tr><tr><td>mid</td><td>/m/0gkxy13</td></tr><tr><td>score</td><td>0.61064434</td></tr><tr><td>topicality</td><td>0.61064434</td></tr></table></td></tr><tr><td>path</td><td>.DATA.result.responses[1].labelAnnotations[7].description</td></tr></table></td></tr></table></td></tr></table>	struct		Handwriting	<table><tr><td colspan="2">array</td></tr><tr><td>1</td><td><table><tr><td colspan="2">struct</td></tr><tr><td>key</td><td>description</td></tr><tr><td>owner</td><td><table><tr><td colspan="2">struct</td></tr><tr><td>description</td><td>Handwriting</td></tr><tr><td>mid</td><td>/m/0gkxy13</td></tr><tr><td>score</td><td>0.61064434</td></tr><tr><td>topicality</td><td>0.61064434</td></tr></table></td></tr><tr><td>path</td><td>.DATA.result.responses[1].labelAnnotations[7].description</td></tr></table></td></tr></table>	array		1	<table><tr><td colspan="2">struct</td></tr><tr><td>key</td><td>description</td></tr><tr><td>owner</td><td><table><tr><td colspan="2">struct</td></tr><tr><td>description</td><td>Handwriting</td></tr><tr><td>mid</td><td>/m/0gkxy13</td></tr><tr><td>score</td><td>0.61064434</td></tr><tr><td>topicality</td><td>0.61064434</td></tr></table></td></tr><tr><td>path</td><td>.DATA.result.responses[1].labelAnnotations[7].description</td></tr></table>	struct		key	description	owner	<table><tr><td colspan="2">struct</td></tr><tr><td>description</td><td>Handwriting</td></tr><tr><td>mid</td><td>/m/0gkxy13</td></tr><tr><td>score</td><td>0.61064434</td></tr><tr><td>topicality</td><td>0.61064434</td></tr></table>	struct		description	Handwriting	mid	/m/0gkxy13	score	0.61064434	topicality	0.61064434	path	.DATA.result.responses[1].labelAnnotations[7].description
struct																											
Handwriting	<table><tr><td colspan="2">array</td></tr><tr><td>1</td><td><table><tr><td colspan="2">struct</td></tr><tr><td>key</td><td>description</td></tr><tr><td>owner</td><td><table><tr><td colspan="2">struct</td></tr><tr><td>description</td><td>Handwriting</td></tr><tr><td>mid</td><td>/m/0gkxy13</td></tr><tr><td>score</td><td>0.61064434</td></tr><tr><td>topicality</td><td>0.61064434</td></tr></table></td></tr><tr><td>path</td><td>.DATA.result.responses[1].labelAnnotations[7].description</td></tr></table></td></tr></table>	array		1	<table><tr><td colspan="2">struct</td></tr><tr><td>key</td><td>description</td></tr><tr><td>owner</td><td><table><tr><td colspan="2">struct</td></tr><tr><td>description</td><td>Handwriting</td></tr><tr><td>mid</td><td>/m/0gkxy13</td></tr><tr><td>score</td><td>0.61064434</td></tr><tr><td>topicality</td><td>0.61064434</td></tr></table></td></tr><tr><td>path</td><td>.DATA.result.responses[1].labelAnnotations[7].description</td></tr></table>	struct		key	description	owner	<table><tr><td colspan="2">struct</td></tr><tr><td>description</td><td>Handwriting</td></tr><tr><td>mid</td><td>/m/0gkxy13</td></tr><tr><td>score</td><td>0.61064434</td></tr><tr><td>topicality</td><td>0.61064434</td></tr></table>	struct		description	Handwriting	mid	/m/0gkxy13	score	0.61064434	topicality	0.61064434	path	.DATA.result.responses[1].labelAnnotations[7].description				
array																											
1	<table><tr><td colspan="2">struct</td></tr><tr><td>key</td><td>description</td></tr><tr><td>owner</td><td><table><tr><td colspan="2">struct</td></tr><tr><td>description</td><td>Handwriting</td></tr><tr><td>mid</td><td>/m/0gkxy13</td></tr><tr><td>score</td><td>0.61064434</td></tr><tr><td>topicality</td><td>0.61064434</td></tr></table></td></tr><tr><td>path</td><td>.DATA.result.responses[1].labelAnnotations[7].description</td></tr></table>	struct		key	description	owner	<table><tr><td colspan="2">struct</td></tr><tr><td>description</td><td>Handwriting</td></tr><tr><td>mid</td><td>/m/0gkxy13</td></tr><tr><td>score</td><td>0.61064434</td></tr><tr><td>topicality</td><td>0.61064434</td></tr></table>	struct		description	Handwriting	mid	/m/0gkxy13	score	0.61064434	topicality	0.61064434	path	.DATA.result.responses[1].labelAnnotations[7].description								
struct																											
key	description																										
owner	<table><tr><td colspan="2">struct</td></tr><tr><td>description</td><td>Handwriting</td></tr><tr><td>mid</td><td>/m/0gkxy13</td></tr><tr><td>score</td><td>0.61064434</td></tr><tr><td>topicality</td><td>0.61064434</td></tr></table>	struct		description	Handwriting	mid	/m/0gkxy13	score	0.61064434	topicality	0.61064434																
struct																											
description	Handwriting																										
mid	/m/0gkxy13																										
score	0.61064434																										
topicality	0.61064434																										
path	.DATA.result.responses[1].labelAnnotations[7].description																										

FIGURA 9 – RILEVAMENTO DI HANDWRITING

5.3 Estrapolazione dei codici di spedizione

In quest'ultima parte del capitolo dedicato alla realizzazione del modulo vengono riportati i dettagli relativi all'implementazione del processo che si occupa di estrapolare dal documento ricevuto il codice di spedizione.

Si porta all'attenzione del lettore il fatto che tale codice è l'unica informazione che deve essere ottenuta con un margine di errore nullo.

Benché sul documento di spedizione siano riportate molte informazioni relative alla stessa, come ad esempio la data, il tipo e la quantità di materiale spedito, la destinazione e via dicendo, tali informazioni sono già note al sistema in quanto sono state fornite nel momento in cui è stata registrata la spedizione e generato il relativo documento.

In questo contesto, una corretta acquisizione del codice di spedizione è necessaria affinché il sistema possa risalire correttamente ai dati relativi alla stessa, aggiornare le giacenze nei magazzini interessati e contrassegnare la spedizione come avvenuta in una determinata data.

Le operazioni che vengono eseguite su banca dati in questa parte del programma non saranno oggetto di discussione nella seguente tesi in quanto non riguardano direttamente il funzionamento del modulo per il controllo di documenti e aggiungerebbero informazioni tecniche di carattere documentale poco rilevanti in questo contesto.

Il codice riportato sui documenti di spedizione è un codice numerico che può assumere due forme:

- Può presentarsi sottoforma di codice *QR*
- Può essere scritto esplicitamente preceduto dal prefisso “*XAB*”

Il sistema è stato progettato e costruito in maniera tale da potersi adattare a entrambi i casi. Un utente può fotografare indifferentemente una bolla di spedizione munita di codice *QR* o di codice *XAB*.

5.3.1 Codici QR

Per i due scenari, inizialmente i controlli eseguiti sono gli stessi (integrità dell'acquisizione e presenza di firme). Arrivati a un certo punto del flusso di esecuzione, se all'interno del documento viene rilevato un codice QR si entra in un ramo in cui tale codice viene decodificato e vengono eseguite determinate operazioni in archivio dati.

La presenza di codici QR all'interno del documento viene segnalata dallo stesso componente che si occupa di scremare tutti i risultati ottenuti da *Google Vision*. Quest'ultimo infatti può individuare all'interno di documenti codici a barre unidimensionali e bidimensionali, fornendo le coordinate dei vertici degli stessi e uno *Score* analogo a quello dell'*Handwriting*.

Si è scelto di mantenere, per la decodifica dei codici QR, l'integrazione con *ZXing* già presente dalla prima versione del modulo. Complice di tale scelta è stato il fatto che la funzionalità di *Google Vision* per la scansione di codici a barre è indicata come deprecata.

Per la lettura del codice QR viene invocata un'apposita funzione esterna al modulo, secondo il principio che suggerisce di mantenere alta coesione e basso accoppiamento all'interno del codice. Stesso principio viene applicato nei riguardi della funzione che si occupa di eseguire le operazioni in archivio.

Nella seguente figura viene illustrata la semplicità con la quale si possono effettuare chiamate API tramite il tag `<cfhttp>` di *ColdFusion*.

```
3723
3724
3725     <cfhttp
3726         url="https://zxing.org/w/decode?u=#arguments.cImgPath#"
3727         method="get"
3728         result="sResponse"
3729     >
3730 </cfhttp>
```

FIGURA 10 – CHIAMATA API IN COLDFUSION

5.3.2 Codici XAB

Qualora all'interno del documento non venisse rilevato alcun codice QR, allora è possibile che nella bolla di spedizione il codice sia scritto esplicitamente all'interno del testo, preceduto dal prefisso "XAB".

Nella figura sottostante è riportato un esempio di bolla di spedizione con queste caratteristiche.

interport s.r.l.
Lungotevere della Vittoria, 9
Roma 00195
C.F./P.IVA 01410870594

Deposito Intermedio Doganale IT00CEY00952T
Deposito Strategico Nazionale IT11287
81037 SESSA AURUNCA (CE)
Via Appia Km 158.4 - Tel +39 0771.938170

intergroup
logistics experience

BOLLA ACCOMPAGNATORIA BENI VIAGGIANTI
(D.P.R. 627/78)
XAB 02160 /2021
MODUL PROJECT s.a.s. - Via Migliara 42 n° 9 - 04100 BORGIO FAITI (LT)
Cod. Fisc. / P. IVA 01828970598 - Aut. Min. n. 88723 del 31.07.1998

COMMITTENTE / CEDENTE / DEPOSITANTE
COLACEM SPA
VIA DELLA VITTORINA 60
06024 GUBBIO (PG)
C.F./P.IVA 01157050541

Numero 2/21/1534 Data 27/04/2021
Codice cliente 10000020

LUOGO DI PARTENZA WH SESSA VIA APPIA, KM 158.4 - 81037 SESSA AURUNCA (CE) IT
DESTINATARIO COLACEM SPA VIA DELLA VITTORINA 60 06024 GUBBIO (PG)
COD. DESTINAZIONE COLACEM SPA - C DA CARRERA DEL CONTE COD DITTA IT00ISC00001C 86078 SESTO CAMPANO (IS)

CAUSALE TRASPORTO TRASFERIMENTO PARTITA
RIF. M/SOFIAR

COD.		DESCRIZIONE	PESO LORDO	PESO NETTO
092		Ns. Ordine 21-OS001875 del 10/03/21 PETCOKE PETCOKE - 092 RINFUSA	31,16TON	31,16TON

TOTALI

TRASPORTO A CARICO DEL ☐ MITTENTE / DEPOSITARIO ☐ COMMITTENTE / CEDENTE / DEPOSITANTE / DESTINATARIO ☒ VETTORE

TARGA ASPETTO ESTERIORE DEI BENI PORTO X

DATA 27/04/2021 ORA 11:22

FIRMA DEL VETTORE / 27/04/2021

FIRMA DEL DESTINATARIO

EBB26LC#AD30319RE RINFUSA
AUTOTRASPORTI F.LLI DI CRISTINZI SRL VIA PIER DELLA FRANCESCA 10 MONTAQUILA (IS)

FIGURA 11 – DOCUMENTO DI SPEDIZIONE CON CODICE XAB

All'interno di questo ramo del flusso di esecuzione la faccenda si complica in quanto non ci si può avvalere di uno strumento dedicato all'extrapolazione del codice, quale poteva essere un decodificatore di codici a barre.

Si rende quindi necessario costruire uno strumento che vada a cercare all'interno del documento la porzione di testo in cui è contenuto il codice di spedizione.

Sempre in base ai principi di alta coesione e basso accoppiamento, viene istituita una funzione dedicata all'extrapolazione del codice che viene invocata durante il flusso di esecuzione principale del programma.

Tale funzione si occupa di effettuare una chiamata *API* al servizio di *OCRSpace* che, data un'immagine in input, ritorna come output una stringa corrispondente al testo contenuto in tale immagine.

Il componente ausiliario si occupa quindi di cercare all'interno di tale stringa la sequenza di numeri corrispondente al codice di spedizione.

Per adempiere a questo compito la funzione ricorre all'espedito del *pattern matching*, ovvero va alla ricerca di una sequenza di caratteri avente determinate caratteristiche.

Tali caratteristiche possono essere facilmente individuate guardando l'immagine sovrastante: il codice è composto da una sequenza di 5 numeri, preceduti dal prefisso "XAB" e seguiti dal suffisso "/2021" che fa evidentemente riferimento all'anno in cui la spedizione avviene.

A questo punto basta individuare il punto della stringa in cui occorre la sequenza di caratteri "XAB" e il punto della stringa in cui occorre l'anno della spedizione preceduto da uno "/" e basterà andare a considerare il codice di spedizione come composizione di tutti i numeri che occorrono nel mezzo.

Diventa però doveroso notare come l'algoritmo che implementa il *pattern matching* debba adattarsi a diverse situazioni, come ad esempio possono essere la ricezione di bolle che fanno riferimento a spedizioni in anni successivi al 2021 oppure eventuali errori di decodifica del testo contenuto nell'immagine.

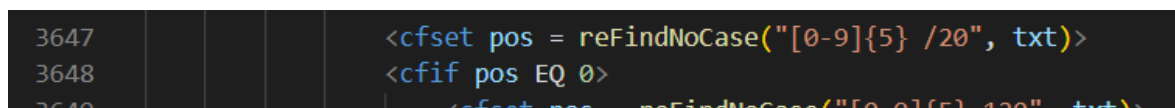
È stato infatti empiricamente notato come le funzionalità di *OCRSpace* possano in determinati casi fornire un'interpretazione errata del carattere "/", che viene talvolta confuso con un "|", un "1" o, a volte, direttamente omesso.

Tale errore di interpretazione introduce difficoltà relative al fatto che, in determinati casi, quando i caratteri “/” vengono omessi o sostituiti con un “1” molte delle date che sono presenti nel documento diventano potenzialmente pericolose.

Per fare un esempio, basti pensare arbitrariamente alla stringa “19/10/2021”, che in seguito a un errata interpretazione dovuta a una foto scattata in condizioni di scarsa luce ha la possibilità di essere interpretata come “19110/2021” e quindi diventare terribilmente simile a un codice di spedizione.

L’algoritmo implementato dalla funzione deve essere un ottimo compromesso tra flessibilità e affidabilità. L’implementazione di tale algoritmo è stata semplificata dall’utilizzo di funzioni di *ColdFusion* adibite alla ricerca di pattern specifici.

Viene illustrato nell’immagine sottostante l’utilizzo della funzione *reFindNoCase* che prende come argomento una delle sequenze da trovare, in questo caso una sequenza di {5} caratteri compresi nell’intervallo [0-9] e seguiti dalla stringa “/20” all’interno della variabile *txt*, dove viene iniettato il testo estrapolato dal documento o una porzione di esso.



```
3647 <cfset pos = reFindNoCase("[0-9]{5} /20", txt)>  
3648 <cfif pos EQ 0>  
3649 <cfset pos = reFindNoCase("[0-9]{5} 120", txt)>
```

FIGURA 12 – FUNZIONE PER ESEGUIRE PATTERN MATCHING

Si noti come per necessità di semplificazione non vengono elencate tutte quante le casistiche a cui l’algoritmo di *pattern matching* deve far fronte.

Viene invece portato all’attenzione del lettore ed esposto nell’immagine sottostante uno dei passaggi chiave nell’estrpolazione del codice di spedizione, volto a verificare che la sequenza di caratteri individuata abbia il prefisso ed il suffisso giusti e contenga solo caratteri numerici.

```

3662      </cfif>
3663
3664      <cfif sReturn.DATA.XABCHECK EQ 0>
3665          <cfset pos = FindNoCase("XAB", txt)>
3666          <cfif pos GT 0>
3667              <cfset xab = Trim(mid(txt, (pos+4), 5))>
3668              <cfif IsNumeric("#xab#")>
3669                  <cfset sReturn.DATA.XABCHECK = 1>
3670              </cfif>
3671          </cfif>
3672      </cfif>
3673
3674      <cfif (sReturn.DATA.XABCHECK EQ 1) AND (Len(xab) EQ 5)>

```

FIGURA 13 – ALCUNI DEI CONTROLLI DI VALIDAZIONE EFFETTUATI

Concluso il controllo, questo secondo ramo di esecuzione termina restituendo il codice individuato al processo principale, che in maniera analoga a quanto succede per i documenti provvisti di codice QR, invoca una funzione ausiliaria che si occupa di tutte le operazioni che devono essere eseguite sulla banca dati, la cui esistenza è ancora una volta giustificata dai principi di alta coesione e basso accoppiamento sopra citati.

A questo punto il programma si occupa di raccogliere, concatenandoli tra loro, tutti i messaggi diagnostici che è necessario inoltrare all'operatore nei casi in cui, ad esempio, un documento non sia stato firmato, o siano occorsi dei problemi durante la rilevazione del codice, oppure in caso di disservizio da parte di una delle API esterne al sistema e così via dicendo.

Tale messaggio diagnostico grazie alle funzionalità integrate di *Twilio* viene recapitato al mittente del documento, sempre tramite *WhatsApp*, nel giro di qualche minuto dalla ricezione.

Ovviamente la durata dell'attesa varia in funzione del numero di messaggi che si trovano nella coda. Questo avviene in quanto, per politiche di sicurezza volte a evitare l'accesso concorrente a determinate risorse, il sistema processa un solo messaggio alla volta.

6. VERIFICHE FUNZIONALI

Segue a questo punto un capitolo dedicato alla verifica del corretto funzionamento del modulo.

Tale verifica è stata effettuata principalmente in due modalità. La prima prevede il controllo funzionale dei singoli componenti, comprese tutte le funzioni ausiliarie, ed è stata svolta sull'infrastruttura di sviluppo per non creare situazioni ambigue in ambiente di produzione.

La seconda modalità prevede invece la verifica del modulo in ambiente di produzione previa accordo con il personale di Intergroup, al fine di testare il modulo nella sua interezza, dall'invio delle bolle di spedizione tramite messaggio *WhatsApp* al recapito della diagnostica che regolarmente avviene nel giro di qualche minuto.

È stato scelto deliberatamente di non fare uso di alcun tipo di automazione per il processo di testing in quanto in questo contesto risulta necessaria la presenza di un essere umano in grado di verificare che quanto elaborato dal programma corrisponda all'effettivo contenuto del documento.

Si pone l'attenzione sul fatto che pur volendo istituire dei test che automaticamente verifichino che il risultato dell'elaborazione del documento corrisponde ad un risultato atteso, è necessario che tale risultato atteso sia frutto di un'elaborazione umana per ognuno dei documenti oggetto di testing.

Benché attuare il secondo degli scenari descritti sia indiscutibilmente una buona pratica, alla fine si è optato per la verifica “a mano” delle funzionalità, complice anche l'immediatezza con cui le acquisizioni dei documenti possono essere sottoposte al sistema.

6.1 Il dataset oggetto di testing

Sono stati sottoposti al sistema documenti contenenti codici di entrambi i tipi, sia QR che XAB.

Per ottenere degli scenari quanto più realistici possibili, i documenti sono stati acquisiti da diverse angolazioni e con diverso grado di rotazione.

La scelta di adottare tale pratica si è rivelata fondamentale al fine di optare per l'adozione di determinate soluzioni implementative nel codice. Tali considerazioni sono affrontate in maniera più approfondita nel seguito di questo capitolo.

Nell'immagine seguente è riportato il risultato dell'analisi di un documento ad opera della funzione ausiliaria atta ad individuare ed estrapolare il codice XAB.

struct		
DATA	struct	
	NFOUND	0
	OCRAPI	***** Result for Image/Page 1 ***** interport s.r.l. Lungotevere della Vittoria, 9 Roma 00195 C.F.I P. IVA 0141 experience BOLLA ACCOMPAGNATORIA BENI VIAGGIANTI SPADENTE / DEPOSITANTE Numero 2/21/1530 Data 27 P. IVA01828970598 - Aut. Min. n. 88723 del 31.07.1998 LUOGO DI PARTENZA WH SESSA VIA APPIA, KM 158,4 810 00034 COLLEFERRO (NM) Codice cliente 10000346 CAUSALE TRASPORTO TRASFERIMENTO PARTITA RIF. MN RESO 27/04/2021 E 1 CONDUCENTE PESO NETTO 29,72TON 29,72 ORA INIZIO TRASPORTO 10:01 COMMITTENTE /CEDI ESTERIORE DEI BENI RINFUSA PORTO INTERLOG SRL Lun Caboto, 110 04024 GAETA (LT) IT FIRMA DEL DESTINAT
	XAB	02156
	XABCHECK	1

FIGURA 14 -ESTRAPOLAZIONE DI UN CODICE XAB

Il testing di questa funzione in particolare è stato eseguito in ambiente di sviluppo chiamando il componente via *url* e fornendo come parametro il percorso locale del file che si intendeva sottoporre al controllo.

È stato rilevato come, nonostante tutte le situazioni ambigue che deve fronteggiare, l'algoritmo implementato da tale funzione risulti abbastanza robusto, riportando un risultato corretto in un intervallo che si attesta attorno all'80% dei casi presi in esame.

Viene infine riportata, a titolo di completezza, una schermata acquisita da telefono che intende mostrare la maniera in cui le fotografie dei documenti sono inviate e valutate via *WhatsApp* nell'ambito dei test eseguiti in produzione.

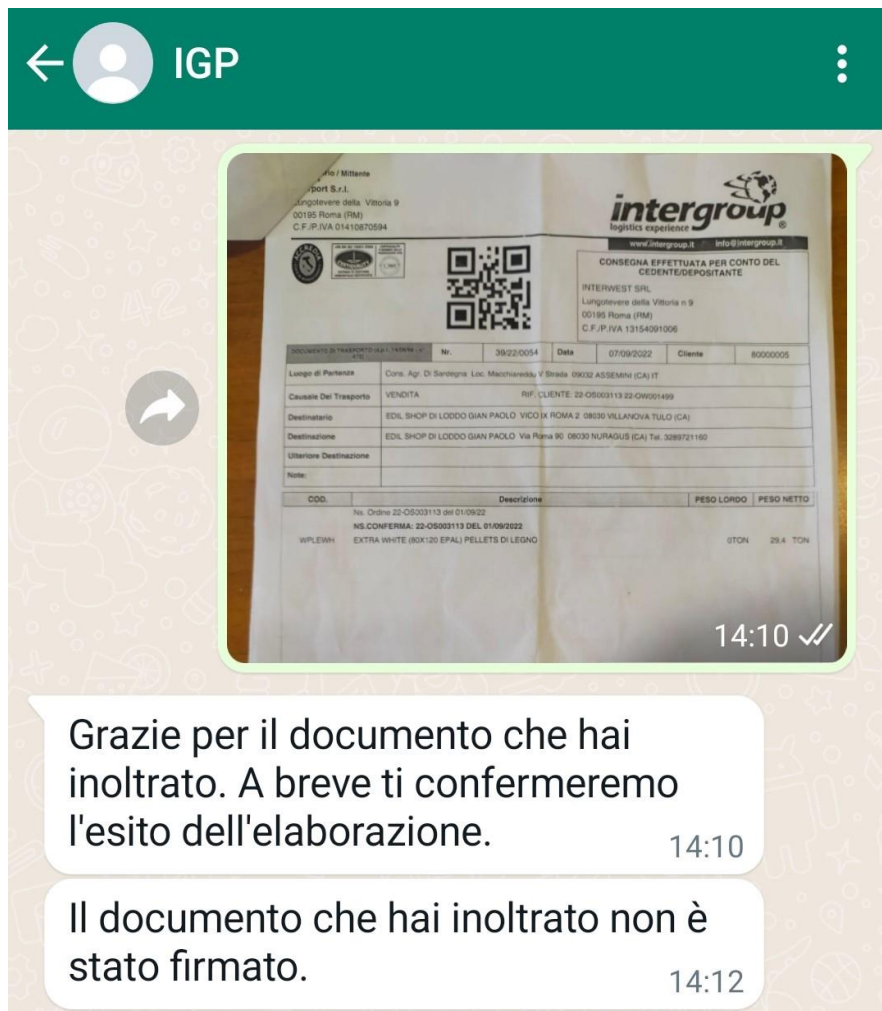


FIGURA 15 – TEST ESEGUITI IN PRODUZIONE

6.2 Imprevisti e difficoltà incontrate

Le maggiori criticità incontrate durante lo sviluppo del modulo per la ricezione e il controllo di documenti fotografati via *WhatsApp* sono state relative ai servizi offerti da *Google Vision*.

Nella fattispecie, un primo inconveniente riguarda la ricezione e la mancata decodifica di codici QR presenti all'interno dei documenti.

È infatti emerso dai test come talvolta, quando vengono sottoposti documenti fotografati con un'inclinazione molto pronunciata ($>45^\circ$) *Google Vision* inizi a fare fatica ad intercettare correttamente tutti gli elementi contenuti nel foglio, porzioni di testo e codici a barre compresi.

In questo ambito, i servizi offerti da *ZXing* e *OCRSpace* sono risultati molto più robusti e affidabili.

A tal proposito, si è deciso di sottoporre a *ZXing* le acquisizioni per un secondo controllo quando *Google Vision* non riesce ad individuare un codice QR nel documento. Facendo in questo modo, se una prima analisi ad opera di *Vision* individua un codice a barre all'interno del documento allora il flusso di esecuzione entrerà nel ramo relativo all'analisi di codici QR, qualora invece la prima analisi dovesse dare un esito negativo tale esito potrà essere confermato solamente da una seconda analisi ad opera di *ZXing*.

A questo punto, se entrambe le analisi risulteranno avere un esito negativo, il programma dirotterà il flusso di esecuzione nel ramo in cui si va alla ricerca di un codice XAB.

Ed è in questo contesto che emerge la difficoltà di *Google Vision* relativa all'estrapolazione di testi da immagini e documenti.

Infatti, nonostante il corredo di *Vision* comprenda tale funzionalità, in presenza di documenti acquisiti ancora una volta con una rotazione della fotocamera sufficientemente molesta la lettura del testo verrà compromessa e alcune porzioni delle righe verranno scambiate con quelle sovrastanti o sottostanti (in base ai punti di vista).

Infatti, per tutto il ramo di esecuzione relativo alle bolle di spedizione contenenti codici XAB si è deciso di avvalersi dei servizi di OCRSpace, che risulta più robusto nella lettura di documenti acquisiti con angolazioni strane.

Riguardo quest'ultima, l'unica criticità riscontrata è relativa al numero di richieste che si possono inoltrare. In particolare non sono le richieste giornaliere a porre dei limiti, ma il numero di richieste che si possono sottoporre in un intervallo di tempo sufficientemente piccolo.

Quando l'API viene messo sotto stress e riceve decine di richieste a distanza di pochi secondi l'una dall'altra cade spesso in down e interrompe il servizio per circa 5 minuti.

Si è quindi reso necessario implementare un controllo (realizzato utilizzando un semplice meccanismo di *request timeout*) che avvisa l'utenza del fatto che uno dei componenti esterni al sistema è fuori servizio per qualche minuto.

Probabilmente tale fragilità è prerogativa della versione gratuita del servizio offerto da OCRSpace, versione che si è scelto di adoperare in fase di testing dal momento in cui le 500 richieste giornaliere eccedevano largamente il numero di chiamate che era necessario effettuare.

L'ultimo imprevisto riguarda infine il concetto di "*Handwriting*" posto in essere dalla piattaforma di Google Vision.

Si è infatti notato, sulla base di osservazioni empiriche in pieno stile *Scrum*, come l'intelligenza del colosso californiano considerasse come "scritte a mano" soltanto le frasi o le parole che poteva, in una certa misura, interpretare.

Questa considerazione ha senso se si pensa al fatto che uno scarabocchio fatto a penna sul foglio che viene fotografato non può e non deve essere considerato al pari dell'apposizione di una firma da parte di un operatore affiliato a *Intergroup*.

Questa stessa considerazione fa cadere in una zona grigia di indecisione e ambiguità tutte le firme eseguite con una calligrafia poco o per nulla leggibile.

Tuttavia non può non essere tenuto conto del fatto che i sistemi di *machine learning* di Google sono in continua evoluzione e potrebbe essere solo questione di tempo prima che questa situazione di ambiguità venga risolta.

7. CONCLUSIONI

Una volta illustrate tutte le caratteristiche e i dettagli del progetto realizzato viene lasciato spazio ad alcune considerazioni finali.

L'obiettivo del lavoro svolto nell'ambito del tirocinio e oggetto della presente tesi è stato lo sviluppo di un modulo per la ricezione e il controllo di documenti fotografati da dispositivo mobile come parte di un applicazione web basata su tecnologie cloud.

Il sistema sviluppato risulta essere abbastanza stabile e adempie correttamente ai compiti per il quale è stato designato, al netto di sottili differenze funzionali rispetto a quelle previste nella prima fase di analisi dei requisiti, che ad ogni modo non compromettono assolutamente il corretto funzionamento dell'applicativo.

Il modulo è stato progettato e realizzato in maniera tale da poter essere facilmente rifattorizzato, grazie all'applicazione di espedienti che perseguono i principi di alta coesione e basso accoppiamento, qualora occorresse apportare modifiche o effettuare integrazioni.

La piattaforma *IGP/Yaya* è infatti oggetto di continue rivisitazioni e miglioramenti nel corso del tempo, volti a far fronte alle necessità degli utenti che la adoperano quotidianamente in ambito lavorativo.

A tal proposito si rivela azzeccata la scelta di adottare un approccio agile allo sviluppo del software, visti i vantaggi che comporta in termini di flessibilità sia durante la progettazione che nella fase di realizzazione dello stesso.

Si vuole infine sottolineare come la decisione di implementare il programma basandosi su tecnologie appartenenti al mondo delle applicazioni *web based* e delle *progressive web app* in particolare sia stata una scelta ottima viste le numerose proprietà di carattere innovativo che è possibile attribuire a quelle che, ad un occhio inesperto, potrebbero apparire come delle semplici pagine di un sito internet.

INDICE DELLE FIGURE

FIGURA 1 - IL SISTEMA YAYA	13
FIGURA 2 - IL SISTEMA IGP	15
FIGURA 3 - DIAGRAMMA DI INTERAZIONE DEL SISTEMA.....	28
FIGURA 4 -ANALISI DI GOOGLE VISION	32
FIGURA 5 -RISPOSTA IN FORMATO JSON.....	32
FIGURA 6 - ANALISI DI OCRSPACE.....	33
FIGURA 7 - RICERCA DI NODI IN UNA STRUTTURA.....	35
FIGURA 8 - INDIVIDUAZIONE DI MARCATORI.....	36
FIGURA 9 - RILEVAMENTO DI HANDWRITING	38
FIGURA 10 - CHIAMATA API IN COLDFUSION.....	40
FIGURA 11 - DOCUMENTO DI SPEDIZIONE CON CODICE XAB	41
FIGURA 12 - FUNZIONE PER ESEGUIRE PATTERN MATCHING	43
FIGURA 13 - ALCUNI DEI CONTROLLI DI VALIDAZIONE EFFETTUATI.....	44
FIGURA 14 - ESTRAPOLAZIONE DI UN CODICE XAB	46
FIGURA 15 - TEST ESEGUITI IN PRODUZIONE	47

BIBLIOGRAFIA

- [1] Ansa, “Italia, più smartphone che abitanti: il panorama digitale 2020”
- [2] Statista, “Global smartphone penetration rate from 2016 to 2020”
- [3] YayaMedia, <https://www.yayamedia.it/>
- [4] Google Developers, <https://web.dev/progressive-web-apps/>
- [5] IBM, <https://www.ibm.com/it-it/cloud/learn/three-tier-architecture>
- [6] Mozilla, https://developer.mozilla.org/en-US/docs/Web/API/Service_Worker_API
- [7] W3C, <https://www.w3.org/TR/appmanifest/>
- [8] Wikipedia, https://it.wikipedia.org/wiki/Progressive_Web_App
- [9] Adobe, <https://helpx.adobe.com/coldfusion/cfml-reference/user-guide.html>
- [10] YayaMedia, “IGP Architecture”
- [11] YayaMedia, “Documentazione IGP”
- [12] Wikipedia, https://it.wikipedia.org/wiki/Ward_Cunningham
- [13] Manifesto Agile, <https://agilemanifesto.org/iso/en/manifesto.html>
- [14] Wikipedia, https://it.wikipedia.org/wiki/Modello_a_cascata
- [15] Ken Schwaber e Mike Beedle, “Agile Software Development with Scrum”
- [16] OOPSLA, <https://dblp.org/db/conf/oopsla/oopsla95.html>
- [17] Ken Schwaber e Jeff Sutherland, “La Guida a Scrum”
- [18] Intergroup, <https://www.intergroup.it/articoli/>
- [19] Twilio, <https://www.twilio.com/>
- [20] ZXing, <https://zxing.org/w/decode.jspx>

[21] Google Vision, <https://cloud.google.com/vision>

[22] OCRSpace, <https://ocr.space/>