

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

CAMPUS DI CESENA
SCUOLA DI SCIENZE

Corso di Laurea in Ingegneria e Scienze Informatiche

API DROPBOX: REALIZZAZIONE DI
UN'APPLICAZIONE PER LA CREAZIONE DI
RAPPORTI DI INTERVENTO

Elaborata nel corso di: Programmazione Orientata agli Oggetti

Tesi di Laurea di:
MICHAEL CAMPORESI

Relatore:
Prof. MIRKO VIROLI

ANNO ACCADEMICO 2017–2018
SESSIONE I

PAROLE CHIAVE

Android
Download
Dropbox
Java
Upload

A Silvia e Massimo, senza i quali questo risultato non
sarebbe stato raggiunto.

Indice

Introduzione	iii
1 Background	1
1.1 Introduzione	1
1.2 Linguaggi utilizzati	2
1.3 Librerie utilizzate e utilities	3
1.4 PDF e XML	4
1.5 Dropbox	4
1.6 Conclusione	8
2 Architettura del sistema	9
2.1 Introduzione	9
2.2 Descrizione del problema	10
2.3 Requisiti del sistema	11
2.4 Soluzione proposta	12
2.5 Motivi della scelta e vantaggi	19
2.6 Soluzioni alternative	20
2.7 Conclusione	21
3 Sviluppo dell'applicazione	22
3.1 Introduzione	22
3.2 Sviluppo file e sviluppi preliminari	23
3.3 Sviluppo sezione desktop	24
3.4 Testing sezione desktop	25
3.5 Sviluppo sezione android	26
3.6 Testing sezione android	27
3.7 Descrizione delle classi desktop	28

3.7.1	Interfaces	28
3.7.2	Utilities	29
3.7.3	Windows	34
3.8	Descrizione manifest e classi android	40
3.8.1	Manifest	40
3.8.2	UtilitiesAndroid	41
3.8.3	Schermate	52
3.9	Risultato dello sviluppo	55
3.10	Conclusione	63
4	Ringraziamenti	65

Introduzione

Quando sono stato assunto nell'azienda Urbitek Srl, uno dei compiti che mi è stato assegnato era quello di realizzare un'applicazione che gestisse tutti gli aspetti degli interventi che i tecnici realizzano in ogni parte d'Italia.

Durante la fase di studio, quando stavo decidendo quale fosse l'approccio migliore da seguire, ho notato che l'azienda fa un pesante uso dell'archiviazione in cloud offerta da Dropbox e mi sono interessato a questa soluzione. Ho studiato nel dettaglio l'API di Dropbox e le funzionalità che questa offre. In pratica le principali funzionalità sono tre: il download di file da un archivio online, l'upload di file sullo stesso e infine la possibilità di esplorare tale archivio, visualizzando i file e le directory presenti ed eventualmente agendo su di essi.

Sono partito da queste interessanti funzionalità ed ho iniziato lo sviluppo dell'applicazione. Ho sviluppato tutte le funzionalità richieste, in modo tale che l'applicazione potesse gestire tutti gli aspetti dell'intervento: dall'assegnamento di un intervento ad un tecnico alla realizzazione di un documento PDF, il rapporto di intervento, da far firmare al cliente e da inviargli via mail. Ho fatto un pesante utilizzo delle funzionalità offerte dall'API di Dropbox, poiché i file di configurazione necessari all'applicazione (un file per la lista dei clienti registrati nell'applicazione, un file per la lista dei materiali che è possibile usare durante un intervento, un file per i dati sulla numerazione dei documenti ed un file per ogni tecnico che può aver assegnato un intervento) vengono scaricati da Dropbox ad ogni apertura dell'applicazione e, ogni volta che vengono modificati, vengono caricati sull'archivio on-line, in modo tale che ogni tecnico, quando apre l'applicazione per generare un nuovo rapporto di intervento, possa avere sempre i file aggiornati.

Anche i rapporti di intervento vengono caricati su Dropbox una volta generati, cosicché tutti li possano consultare appena vengono creati. La consultazione avviene direttamente all'interno dell'applicazione, sfruttando

la già citata capacità di poter esplorare un archivio Dropbox.

Ho realizzato due sezioni distinte, che hanno un funzionamento che si adatta alle esigenze a seconda della piattaforma ma che partono dalla stessa base. Una è la sezione Desktop (Windows, Mac e Linux), mentre l'altra è la sezione Android. Per lo sviluppo della prima ho utilizzato l'ambiente Eclipse, mentre per lo sviluppo della seconda ho utilizzato Android Studio.

In questa tesi verrà descritto tutto il processo di sviluppo dell'applicazione. Nel Capitolo 1 si parlerà del Background, con una descrizione ed un'analisi accurata delle informazioni utili a comprendere il contesto tecnico che verrà poi analizzato all'interno di questa tesi. Nel Capitolo 2, si parlerà dell'architettura del sistema, descrivendo qual'era il problema da risolvere, quali erano le specifiche da realizzare, qual'è stata la soluzione proposta e perchè è stata questa la soluzione alla fine scelta. Nel Capitolo 3 invece si tratterà dello sviluppo dell'applicazione nel suo insieme, descrivendo tutte le fasi di sviluppo, da quelle preliminari a quelle della sezione Android, passando per lo sviluppo di quella Desktop. Infine verranno analizzate tutte le classi che compongono l'applicazione e verrà analizzato e descritto il risultato finale, con un'analisi dettagliata del funzionamento dell'applicazione finita.

Capitolo 1

Background

In questo capitolo si farà un discorso generale che descriverà le informazioni utili relative al contesto tecnico che tratteremo successivamente.

1.1 Introduzione

In questo capitolo verranno trattate e spiegate tutte le informazioni tecniche necessarie alla comprensione dello sviluppo dell'applicazione.

Nella Sezione 1 verranno trattati i linguaggi utilizzati nello sviluppo dell'applicazione, fornendo una breve spiegazione dei linguaggi Java e Android, così come una descrizione degli ambienti di sviluppo utilizzati.

Nella Sezione 2 verranno trattate le librerie utilizzate durante la realizzazione dell'applicazione, con particolare attenzione alle librerie Dropbox e iTextPDF.

Nella Sezione 3 saranno analizzati i file .PDF e .XML. Verrà fatta una breve descrizione di questi file analizzandone gli aspetti principali.

Nella Sezione 4 infine sarà analizzato Dropbox, del quale verrà spiegato il funzionamento e ne verrà fatta una breve descrizione.

1.2 Linguaggi utilizzati

Durante lo sviluppo dell'applicazione, sono stati utilizzati due linguaggi di programmazione: Java e Android. Entrambi i linguaggi sono linguaggi di Programmazione Orientata agli Oggetti. Questi linguaggi infatti si basano sul concetto di Classe. La classe è il costrutto fondamentale, e svolge la maggior parte delle funzioni. Ogni classe al suo interno comprende campi e metodi. Un campo rappresenta un altro oggetto che è necessario all'interno della classe, mentre un metodo rappresenta una funzione che svolge un determinato compito. Oltre alle classi sono molto importanti anche le Interfacce e le Classi Astratte. Le interfacce specificano i comportamenti che una classe che le implementa dovrà avere, mentre le classi astratte sono classi che al loro interno hanno metodi che vanno implementati nelle classi che estendono quella astratta.

Linguaggio Java e Android sono molto simili, con la differenza che Android si basa sui Layout, che sono schermate gestite sempre da classi Java. Altra differenza è la presenza in Android del file Manifest, che specifica le proprietà dell'applicazione Android.

Come ambienti di sviluppo ho utilizzato Eclipse per quanto riguarda Java e Android Studio per Android. Eclipse è uno strumento potente per lo sviluppo del linguaggio Java, che permette di testare l'applicazione, generare un file Jar eseguibile dell'applicazione ed inoltre evidenzia in tempo reale problemi sul codice ed eventuali errori. Per quanto riguarda Android Studio, questo ha un funzionamento simile ad Eclipse, ma offre in più la possibilità di testare l'applicazione sia su un dispositivo esterno collegato al computer, sia di creare un dispositivo virtuale con specifiche hardware e software personalizzabili su cui provare l'applicazione. Anche Android Studio permette di creare dei file, di tipo Apk, che è possibile copiare su un qualsiasi dispositivo Android e, tramite essi, installare l'applicazione su tale dispositivo.

1.3 Librerie utilizzate e utilities

Durante lo sviluppo dell'applicazione, sono state utilizzate due librerie in particolare: la libreria di Dropbox e la libreria iTextPDF.

La libreria di Dropbox comprende tutte le funzionalità che Dropbox offre, e mette a disposizione tutte le classi che permettono di gestire la connessione al cloud, la gestione dei file presenti su un archivio on-line, upload, download e modifiche e visualizzazione dei metadati dei vari file. La libreria è interamente compresa in un file Jar esterno.

La libreria iTextPDF è una libreria che permette di gestire in vari modi i file PDF. Questa è stata utilizzata in fase di realizzazione dell'applicazione per la creazione dei rapporti di intervento in formato PDF. Permette un elevatissimo livello di personalizzazione e permette di definire tutta una serie di parametri del PDF, così come permette di inserire immagini e tabelle all'interno del documento.

Durante la realizzazione del progetto, sono stati utilizzati anche altri software e siti web, come Bitbucket, un sito che offre la possibilità di ospitare on-line un repository di Mercurial. Un repository è una cartella contenente un progetto nella quale, tramite Mercurial, è possibile eseguire varie operazioni. L'operazione di push permette di caricare on-line il repository aggiornato, mentre l'operazione di pull permette di scaricare il repository aggiornato. Per salvare i cambiamenti, invece, si utilizza l'operazione di commit. Per poter agire su questi repository, è stato utilizzato il programma Tortoise HG, che permette di operare su repository Mercurial. Per realizzare questo elaborato di tesi, invece, è stato utilizzato il programma MikTeX, che permette di realizzare un documento in formato LaTeX, il formato in cui è stata realizzata questa tesi. Per realizzare il file eseguibile di Windows compreso di icona dell'azienda è stato utilizzato il programma Launch4j, che permette di creare eseguibili con icona partendo da un Jar eseguibile. Infine, sono stati utilizzati due plugin di Eclipse, un plugin di Mercurial per poter gestire il progetto all'interno di un repository, e un plugin per la realizzazione di diagrammi UML, tramite il quale sono stati realizzati i diagrammi UML contenuti in questo elaborato.

1.4 PDF e XML

Un file con estensione .PDF è un file che al suo interno comprende testo e/o immagini, che possono essere rappresentate in qualsiasi risoluzione. Un aspetto importante di questo tipo di file è che è un file libero, nel senso che chiunque può realizzare un'applicazione che riguarda tale file.

Per quanto riguarda i file .XML, questi sono file di testo scritti in quello che è un metalinguaggio. Questo consente di controllare il significato di ciò che è descritto all'interno del file.

1.5 Dropbox

Una delle tecnologie più importanti utilizzate all'interno dell'applicazione è sicuramente quella di Dropbox.

Dropbox è un servizio che offre uno spazio di archiviazione in cloud dove poter salvare i propri file. Questo spazio è raggiungibile da qualsiasi dispositivo dotato di connessione ad internet. Inoltre, il servizio offre la protezione dei dati da accessi indesiderati, la protezione dalla perdita dei dati e assicura la continua disponibilità dei dati in qualsiasi momento. Nello spazio riservato all'utente, è possibile salvare file e cartelle. E' possibile inoltre scaricare i file in caso di necessità e anche modificarli.

In questa applicazione è stata utilizzata la libreria di Dropbox per sfruttare tutte le funzionalità che offre, ovvero uno spazio di archiviazione online raggiungibile sempre dal quale è possibile scaricare i file desiderati o caricarli nell'archivio.

Dropbox offriva inizialmente quella che è chiamata API V1, che è stata deprecata per lasciare spazio alla nuova API V2. Questa ha portato diversi vantaggi, come la possibilità di poter studiare un'applicazione che consenta di utilizzare Dropbox anche su Smart Watch, la possibilità di integrare Dropbox in blog nei quali viene data possibilità di upload e download di file, la possibilità di utilizzare Dropbox per eseguire dei backup di informazioni importanti da poter poi consultare e ritrovare in futuro.

Questa API si integra con diversi linguaggi, che sono HTTP, .NET, Java, JavaScript, Python, Swift e Objective-C.

L'API si compone di diverse parti:

- Auth

Questa parte gestisce l'autorizzazione di Dropbox ad accedere ad un determinato archivio. Questa si compone di due sottoparti, una relativa all'autorizzazione tramite token e l'altra relativa all'eventuale revoca dell'autorizzazione.

- File Properties

Questa parte gestisce le proprietà dei vari file. Offre diverse funzionalità e possibilità, come la possibilità di aggiungere, sovrascrivere o rimuovere una proprietà da un file, cercare una proprietà o aggiornarla.

Viene anche data la possibilità di gestire dei templates relativi ad un utente, con la facoltà di aggiungere, ricevere, rimuovere e aggiornare un template da un utente e la possibilità di listare i vari template assegnati ad un utente.

- File Requests

Questa parte gestisce le richieste relative ad un file. Dropbox, infatti, permette di gestire i file anche in modo che, perchè questi siano disponibili alla lettura ed al download, bisogna prima effettuare una richiesta che, in base a determinati parametri, può venire accettata o meno. Questa sezione offre le funzionalità di creare una richiesta, fare una lista delle richieste effettuate, aggiornarne una o riceverla per poter agire su di essa.

- Files

Questa parte gestisce tutte le funzionalità che Dropbox mette a disposizione sui vari file. Questa offre la possibilità di eseguire l'upload di un file verso Dropbox, creare una cartella, copiare un file, cancellare un file da Dropbox, eseguire il download di un file da Dropbox, ottenere una lista di file contenuti nell'archivio e ottenere tutti i metadati relativi ad un file.

Le suddette parti sono comuni in tutti i linguaggi.

Per quanto riguarda Java, i package di cui la libreria è composta sono:

- com.dropbox.core

Questo package è il package principale dell'API.

- `com.dropbox.core.android`
Questo package gestisce le funzionalità specifiche di Android.
- `com.dropbox.core.http`
- `com.dropbox.core.json`
- `com.dropbox.core.stone`
- `com.dropbox.core.util`
Questo package gestisce le utilities.
- `com.dropbox.core.v1`
- `com.dropbox.core.v2`
- `com.dropbox.core.v2.async`
- `com.dropbox.core.v2.auth`
Questo package contiene le classi che gestiscono l'autorizzazione alla connessione.
- `com.dropbox.core.v2.files`
Questo package contiene le classi che permettono le operazioni basilari sui file.
- `com.dropbox.core.v2.properties`
Questo package contiene classi per gestire le diverse proprietà.
- `com.dropbox.core.v2.sharing`
Questo package contiene classi che gestiscono file e cartelle condivisi.
- `com.dropbox.core.v2.team`
Questo package contiene classi che gestiscono varie funzionalità legate ai team (che si possono creare su Dropbox per condividere un archivio).
- `com.dropbox.core.v2.teamcommon`

- `com.dropbox.core.v2.teampolicies`
- `com.dropbox.core.v2.users`

Questo package contiene le classi che gestiscono gli utenti.

Sul sito di Dropbox nella sezione dedicata agli sviluppatori, è presente un blog dove è possibile ottenere supporto durante lo sviluppo di un'applicazione che integra l'API. Inoltre sono presenti diversi esempi di utilizzo che permettono di fare esperienza e prendere familiarità con Dropbox. E' anche presente un'esaustiva documentazione relativa a tutte le funzionalità offerte. Questo per tutti i linguaggi nei quali è possibile integrare l'API.

In sintesi, Dropbox offre la possibilità di avere un archivio sempre disponibile, raggiungibile ovunque tramite la rete, sicuro dalla possibile perdita dei dati, e dotato di un metodo di connessione sicuro. Da questo archivio è possibile leggere e scaricare file, e verso di esso è possibile caricarne. Per ognuno di questi file è possibile leggere anche una serie di informazioni utili, come la data di caricamento, la data di ultima modifica, l'autore, la persona che lo ha caricato, etc.

Questi fattori rendono Dropbox uno strumento potentissimo per qualsiasi sviluppatore e un valore aggiunto per ogni applicazione che debba gestire dei file che devono essere raggiunti tramite internet.

1.6 Conclusione

In questo capitolo abbiamo visto prima quali sono stati i linguaggi utilizzati per programmare l'applicazione, con le varie differenze tra i due linguaggi utilizzati e i due ambienti di sviluppo all'interno dei quali è stato scritto il codice dell'applicazione. Poi abbiamo visto le librerie utilizzate e le varie applicazioni esterne utilizzate per facilitare e migliorare lo sviluppo dell'applicazione e la realizzazione di questa tesi. In seguito è stato descritto che cosa sono i file pdf e i file xml, utilizzati entrambi dall'applicazione, i primi che vengono creati come rapporti di intervento, e gli altri che vengono utilizzati come file di configurazione dell'applicazione. Infine abbiamo visto cos'è Dropbox, quali sono le funzionalità che offre e perchè è stato utilizzato all'interno dell'applicazione.

Nel prossimo capitolo verrà analizzata l'architettura del sistema, dove verrà descritto tutto il lavoro preliminare allo sviluppo vero e proprio dell'applicazione.

Capitolo 2

Architettura del sistema

In questo capitolo sarà descritta l'architettura del sistema, dove verrà innanzitutto descritto il problema per poi arrivare alla sua soluzione.

2.1 Introduzione

In questo capitolo verrà descritta l'architettura del sistema, analizzando tutte le fasi preliminari allo sviluppo. Prima di passare allo sviluppo vero e proprio dell'applicazione infatti è stata fatta una fase di analisi che ha compreso varie attività. Tutte queste informazioni teoriche verranno ora descritte all'interno di questo capitolo.

Nella Sezione 2 verrà descritto il problema. La prima attività ad essere svolta è l'analisi del problema che si deve andare a risolvere, cercando di avere ben chiaro quale sia l'applicazione voluta dal cliente.

Nella Sezione 3 verranno descritti tutti i requisiti dettati dal cliente che il sistema deve rispettare. A questo punto il quadro è completo e sono ben chiare le richieste del cliente.

Il prossimo passo è la descrizione della soluzione proposta, che verrà illustrata nella Sezione 4. Questa proposta arriva dopo un attento studio delle richieste del cliente.

Nella Sezione 5 verranno descritti i motivi e i vantaggi della scelta effettuata, cercando di dimostrare il perchè, per il sistema che deve essere implementato, questa è sembrata la scelta migliore tra quelle possibili.

Infine, nella Sezione 6, verranno elencate ed analizzate delle possibili soluzioni alternative che potevano essere implementate per risolvere il pro-

blema in questione, descrivendo per ognuna di queste eventuali vantaggi e/o svantaggi rispetto alla soluzione che alla fine è stata scelta.

2.2 Descrizione del problema

In questa sezione verrà descritto il problema che è stato risolto dall'applicazione.

Sono stato contattato dal committente perchè aveva bisogno di un'applicazione che gestisse tutte le operazioni relative ad uno degli interventi di manutenzione effettuati dall'azienda. L'applicazione avrebbe dovuto innanzitutto gestire l'assegnamento di un intervento ad un tecnico. Poi dovevano essere salvati, in maniera che fossero sempre disponibili ovunque, i clienti dell'azienda e gli oggetti relativi ad un intervento, cioè oggetti consumabili che si possono utilizzare durante un intervento e oggetti che possono essere sostituiti perchè non più funzionanti. L'azienda infatti si occupa di macchine per la sosta, cioè parcometri e impianti di parcheggio automatizzati, con colonnine di entrata e uscita che dispensano e leggono biglietti, sbarre di entrata e uscita, casse manuali ed automatiche che permettono il pagamento del biglietto e quindi l'uscita da parte del veicolo. Queste macchine si compongono di vari pezzi, che durante un intervento possono essere sostituiti, ritirati o consegnati al cliente come pezzi di ricambio. Ci sono però anche prodotti di pulizia che possono essere utilizzati durante un intervento. Inoltre, doveva essere data la possibilità di inserire nel sistema nuovi clienti e nuovi oggetti.

Il compito principale dell'applicazione doveva essere quello di realizzare un rapporto di intervento riguardante un intervento effettuato da un tecnico, che per fare ciò avrebbe dovuto compilare una serie di campi specificando varie informazioni relative all'intervento effettuato. Una volta raccolte tutte le informazioni, l'applicazione avrebbe dovuto generare un file di tipo pdf da salvare in modo che fosse reperibile in maniera semplice e veloce, da qualsiasi luogo e da ogni tecnico. Le informazioni infatti devono sempre essere reperibili, sempre aggiornate, da ogni tecnico in ogni momento. Infine, è stata richiesta la possibilità di avere una schermata all'interno della quale poter visualizzare tutti i rapporti creati e, in caso, poterli cancellare definitivamente.

2.3 Requisiti del sistema

Questi sono i requisiti che il sistema avrebbe dovuto rispettare:

- Divisione dell'applicazione in due sezioni diverse, Desktop e Android.
- Possibilità di assegnare un intervento ad un tecnico. Questa possibilità deve essere data esclusivamente da Desktop a Android, e non viceversa.
- Desktop dovrà svolgere una funzione di gestione e Android invece sarà la sezione utilizzata dai tecnici.
- Deve comunque essere possibile realizzare un rapporto di intervento anche da Desktop.
- Ogni tecnico deve visualizzare su Android gli interventi ad esso assegnati, ma in numero ridotto in modo da non creare confusione.
- Possibilità di avere sempre a disposizione un database contenente interventi assegnati, clienti, oggetti e dati relativi al numero progressivo da utilizzare nella creazione di un rapporto di intervento.
- Possibilità di aggiungere a questo database nuovi clienti e nuovi oggetti all'occorrenza. Questa possibilità deve essere presente solo su Desktop.
- Possibilità di poter resettare in maniera semplice il numero progressivo (portando ad 1 il numero e aumentando di 1 l'anno). Il numero progressivo deve essere indicato in ogni rapporto di intervento nel formato NNN/AAAA dove N è il numero e A è l'anno.
- I file del database devono essere in chiaro e deve essere possibile modificarli in maniera relativamente semplice.
- Tramite la compilazione di vari campi, deve essere possibile la realizzazione di un rapporto di intervento in formato PDF. Questo file deve essere salvato e deve essere disponibile ovunque e ad ogni tecnico.

- Su Android, deve essere possibile realizzare un rapporto di intervento specifico relativo ad un intervento assegnato. In questo caso, alcuni campi specifici devono già essere precompilati, ma deve rimanere la possibilità per il tecnico di modificarli.
- Per ogni intervento assegnato ad un tecnico, questo deve avere la possibilità di scegliere di ricevere una notifica o meno.
- L'applicazione deve poter funzionare anche in assenza di una connessione ad internet.
- Deve essere presente la possibilità di visualizzare in maniera veloce e semplice tutti i rapporti di intervento realizzati dall'applicazione e deve essere data la possibilità sia su Desktop sia su Android di poter eliminare definitivamente un rapporto.

Questa è una lista di tutte le specifiche che l'applicazione avrebbe dovuto rispettare e di tutte le funzionalità che avrebbe dovuto offrire. Queste sono state richieste dal committente e, come risultato finale, tutte queste specifiche sono state rispettate utilizzando soluzioni che soddisfano appieno il cliente. L'applicazione viene correntemente utilizzata tutt'ora per la creazione dei rapporti di intervento dell'azienda.

2.4 Soluzione proposta

Dopo un attento studio del problema e delle funzionalità richieste dal committente, ho deciso di proporre come soluzione al problema un'applicazione divisa in due sezioni, Desktop (computer con sistemi operativi di tipo Windows e Unix, cioè Mac e Linux) realizzata in Java e un'applicazione Android. Per quanto riguarda il database, ho scelto di realizzare una serie di file di configurazione in linguaggio XML, salvati su un archivio Dropbox opportunamente creato e settato.

Per quanto riguarda Desktop, alla prima apertura dell'applicazione, nella cartella home dell'utente viene creata una cartella chiamata "Urbitek". All'interno di questa cartella verranno scaricati i file di configurazione aggiornati ad ogni apertura del programma. Per quanto riguarda Android, invece, nella fase di installazione dell'applicazione, il sistema Android riserva una parte della memoria del dispositivo all'applicazione. Viene infatti

creata una cartella dedicata all'interno della memoria interna del dispositivo in cui, ad ogni apertura, vengono scaricati tutti i file di configurazione dell'applicazione. In questo modo i dati sono sempre aggiornati.

I file di configurazione sono salvati sull'archivio Dropbox. Questi file sono:

- `urbitek_prog.xml`:

Questo file serve all'applicazione per salvare il prossimo numero progressivo da usare e l'anno corrente, poichè questi due dati devono essere utilizzati per creare il rapporto di intervento, che dovrà essere numerato in modo specifico, cioè nel formato NNN/YYYY dove NNN rappresenta il numero progressivo del rapporto da esprimere sempre in tre cifre e YYYY l'anno corrente. Nella finestra principale dell'applicazione su Desktop, un tasto permette ad un operatore di eseguire un reset del progressivo. Questo riporta a 001 il valore salvato nel file per quanto riguarda il numero ed aumenta di 1 l'anno corrente.

- `urbitek_materials.xml`:

In questo file vengono salvati tutti gli oggetti che possono essere venduti, utilizzati, ritirati o sostituiti durante un intervento. Di ogni oggetto viene salvato il Part Number e la Descrizione. Nella schermata principale su Desktop, un tasto permette di aprire una schermata dove è possibile salvare un nuovo Oggetto, salvandone Part Number e Descrizione. Durante la creazione di un rapporto di intervento, è possibile inserire fino a 15 Oggetti. Nel caso vengano inseriti uno o più oggetti, nel PDF finale verrà creata una tabella contenente i dati di ciascuno. I dati di ogni oggetto oltre al Part Number ed alla Descrizione sono anche un Numero Seriale, il tecnico può indicare se l'Oggetto ne è fornito e/o se è necessario fornirlo, e la Tipologia di Utilizzo dello stesso, se cioè è stato Fornito, Ritirato, Sostituito o si tratta di un prodotto che è stato Utilizzato. Per quanto riguarda l'applicazione Android, i campi nei quali bisogna indicare gli oggetti durante la creazione di un PDF sono dei campi di testo. In questi, digitando una lettera si riceverà un suggerimento riferito agli oggetti presenti nel database. Selezionando uno dei suggerimenti, tutti i campi relativi a tale oggetto si completeranno in automatico. E' possibile però anche compilare i campi personalmente, nel caso ad esempio che l'oggetto da inserire non risulti inserito nel database.

- urbitek_customers.xml:

In questo file vengono salvati tutti i Clienti. Di essi viene salvato il Nome, l'Indirizzo e il Luogo Principale di Intervento. Questi dati vengono utilizzati durante la creazione del rapporto di intervento. Il primo campo da compilare infatti è quello del cliente. Una volta scelto il cliente dall'elenco, verrà compilato in automatico il campo relativo al luogo di intervento, per facilitare la compilazione dei vari campi. Il tecnico potrà poi modificarlo nel caso il luogo sia diverso da quello usuale. Come per gli oggetti, nella schermata principale su Desktop è presente un tasto che permette la creazione di un nuovo cliente, e richiede l'immissione dei 3 campi con Nome, Indirizzo e Luogo Predefinito di Intervento. Come avviene per gli oggetti, su Android i campi relativi al cliente sono campi di testo ad inserimento libero da parte dell'utente. Così si può inserire un cliente non presente nel database oppure, sempre grazie ai suggerimenti, scegliere un cliente da quelli suggeriti e così avere compilati in automatico i campi relativi al cliente.

- urbitek_att_int.xml
urbitek_mic_int.xml
urbitek_gab_int.xml
urbitek_luca_int.xml:

Questi file contengono gli interventi che sono assegnati ad ogni tecnico. In essi vengono salvate le informazioni relative all'intervento da effettuare in modo che il tecnico, una volta assegnato l'intervento, lo veda su Android. Una volta terminato l'intervento, il tecnico può creare un rapporto di intervento specifico su misura, invece che un rapporto generico. In questo modo l'applicazione va a recuperare le informazioni relative allo specifico intervento contenute su questi file e compila già alcuni dei campi che dovrebbe compilare il tecnico, come il Cliente, il Luogo di Intervento, etc. E' poi comunque possibile al tecnico modificare qualsiasi di queste informazioni al momento della creazione del rapporto. Una volta terminata la creazione del rapporto specifico ad un intervento assegnato, questo viene cancellato dall'apposito file e il tecnico in questione non lo vedrà più comparire. In questo modo si considera portato a termine l'intervento.

Tutti i file di configurazione sopra citati vengono scaricati automaticamente ad ogni nuova apertura dell'applicazione e, dopo ogni modifica, vengono salvati in locale e subito caricati su Dropbox. L'applicazione è studiata per funzionare anche in assenza di rete: in tal caso i file aggiornati non vengono scaricati ma vengono usati i file locali, gli ultimi file aggiornati che sono stati scaricati in precedenza. Nel caso dell'upload, su Desktop tutti i file vengono salvati in locale ed un messaggio avverte che non è stato possibile caricarli on-line. In tal caso l'utente dovrà provvedere a spostarli sulla cartella Dropbox facendo sì che una volta tornata la connessione si sincronizzino con l'archivio on-line. Su Android, invece, se non c'è rete e l'upload non va a buon fine, viene fatto partire un servizio persistente di Android in background che, ogni cinque minuti, cerca di caricare online i file aggiornati. Una volta che riesce a caricare i file, il servizio termina. Una notifica permanente rimane attiva per tutta la durata del servizio per avvertire l'utente che il servizio è in esecuzione.

Nella schermata principale su Android è presente un tasto che porta ad una schermata dove vengono visualizzate alcune istruzioni per quanto riguarda il funzionamento dell'applicazione. Insieme alle istruzioni, in tale schermata sono presenti due tasti, che attivano o disattivano il servizio notifica. Questo servizio, che rimane sempre attivo in background, provvede a scaricare ogni dieci minuti i file di configurazione aggiornati e, se al tecnico in questione è stato assegnato un nuovo intervento, invia una notifica per informare il tecnico dell'assegnamento dell'intervento. Per controllare ciò l'applicazione confronta il vecchio file con gli interventi con il nuovo, cercando dei cambiamenti. Finché il servizio rimane attivo, una notifica permanente rimane visibile per avvertire il tecnico che tale servizio è attivo.

Nella stessa cartella su Dropbox vengono anche salvati i Rapporti di Intervento che vengono generati dall'applicazione. Nella schermata principale è presente un tasto che apre una schermata in cui vengono visualizzati tutti i rapporti creati. Oltre al nome (nel quale viene indicato anche il numero progressivo e l'anno) viene anche visualizzata su Desktop la data di caricamento del file, in modo che sia facile risalire alla data di intervento. In questa schermata, per ogni rapporto viene anche data la possibilità di aprirlo e di eventualmente cancellarlo dall'archivio. Visto che l'apertura viene fatta in locale, il file viene prima scaricato in una cartella di appoggio e, per risparmiare spazio, alla chiusura dell'applicazione questa viene pulita di ogni elemento al suo interno. Come applicazione per leggere i

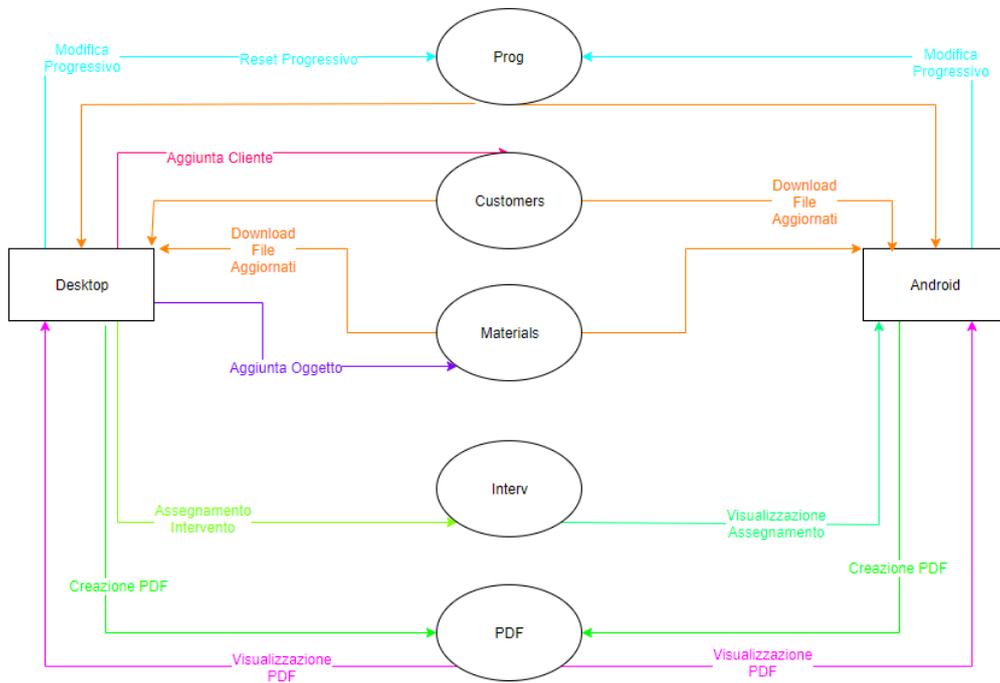
documenti in formato PDF l'applicazione effettua una chiamata di sistema all'applicazione impostata come predefinita.

Per quanto riguarda la creazione del rapporto di intervento, sono presenti una serie di campi che il tecnico deve accuratamente compilare. Questi campi sono:

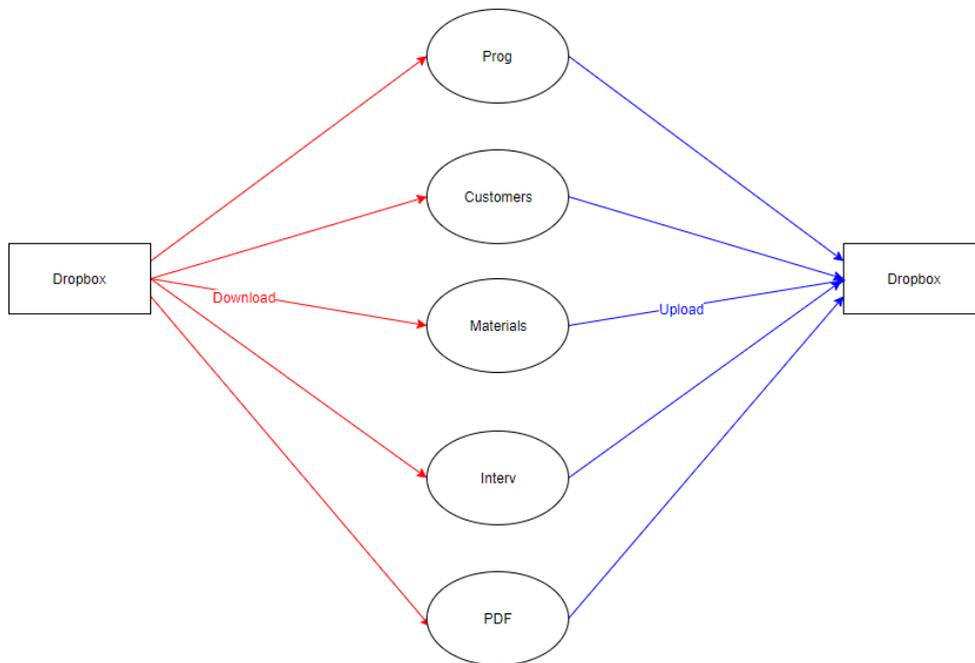
- Nome
Il Nome del Cliente, su Desktop a scelta tra quelli presenti nel sistema.
- Indirizzo
L'indirizzo Legale del Cliente. Questo campo viene compilato automaticamente alla scelta del cliente, ma è possibile modificarlo. Su Android, se il cliente viene scelto selezionando uno dei clienti suggeriti, questo campo viene compilato automaticamente.
- Data
La data di Intervento.
- Motivo
Il Motivo dell'intervento. Questo campo è a scelta tra alcuni motivi preimpostati su Desktop.
- Luogo
Il luogo di intervento. Viene compilato automaticamente alla scelta del cliente ma è sempre possibile modificarlo. Su Android, se viene selezionato un cliente suggerito, viene compilato automaticamente.
- Tipologia
La Tipologia di intervento. Questo campo è a scelta tra alcuni valori preimpostati su Desktop.
- Tempo Impiegato
Il tempo di lavoro impiegato per ciascun tecnico.
- Tempo Trasporto
Il tempo impiegato per il viaggio.

- KM Percorsi
Il numero di KM percorsi per l'intero viaggio.
- Descrizione
La descrizione di tutto quello che è stato fatto durante l'intervento. Questa è la parte più importante e va compilata con molta cura.
- Note Cliente
Eventuali note del cliente.
- Diff. Ore Tecnici
Questa voce permette di differenziare le ore di intervento svolte da più tecnici, specificando per ognuno le ore di lavoro effettuate.
- Tecnici
I tecnici che hanno effettuato l'intervento.
- Materiale
Qui può essere indicato il materiale utilizzato durante l'intervento. Per ogni oggetto si deve specificare Descrizione, Part Number, un eventuale Seriale e l'utilizzo che ne è stato fatto tramite apposite checkbox. Per ogni oggetto, specificato un Part Number o una Descrizione, il corrispettivo campo viene compilato automaticamente. Su Android, se per il Part Number o per la Descrizione viene selezionato un suggerimento, il corrispettivo campo viene compilato automaticamente.

Su Android, per la maggior parte di questi campi vengono visualizzati i suggerimenti all'inserimento della prima lettera. Una volta selezionato un suggerimento, il campo in questione viene compilato automaticamente.



(a) Desktop/Android to Files



(b) Dropbox to Files

Figura 2.1: Architecture Diagram

2.5 Motivi della scelta e vantaggi

Prima di proporre la soluzione al cliente, sono state effettuate molte scelte che ora verranno motivate.

La prima scelta da effettuare è stata quella relativa al linguaggio da utilizzare per lo sviluppo dell'applicazione. La scelta è ricaduta su Java in quanto è il linguaggio che padroneggia maggiormente, oltre al fatto che è un linguaggio potentissimo che permette di sviluppare praticamente ogni tipo di applicazione ed ogni genere di funzionalità. Ovviamente ho dovuto cercare se ci fosse il modo di generare un documento pdf con l'utilizzo di Java e, appena ho trovato la libreria `iTextPdf`, dopo aver appurato che faceva al caso nostro, ho confermato la scelta di Java. Questa scelta è anche motivata dal fatto che la parte di android si sarebbe potuta sviluppare utilizzando molto del codice scritto per la sezione desktop.

La seconda scelta è stata quella di come gestire il database. La scelta è ricaduta su Dropbox per diversi motivi. Il primo è la facilità di sviluppo che Dropbox ha portato. Se non avessi scelto Dropbox, infatti, avrei dovuto sviluppare una sezione server dell'applicazione, che avrebbe dovuto girare su un computer sempre acceso, sempre connesso ad internet, alla quale le sezioni client Desktop e Android si sarebbero dovute connettere per leggere o scrivere i dati sul database gestito dal lato server. L'utilizzo di Dropbox, invece, ha permesso di realizzare un database formato da alcuni file XML, contenuto proprio nell'archivio cloud. L'aggiornamento dei dati non avviene così sul server, ma avviene prima sui client, che poi caricano sul server i file aggiornati. Il server così, rappresentato da Dropbox, si occupa solo di mantenere on-line i file aggiornati.

Si sarebbe potuto gestire il database allo stesso modo con una sezione server creata ad-hoc, ma Dropbox offre il vantaggio di assicurare la non perdita dei dati e la presenza sempre e comunque dei dati sulla rete, garanzie che altrimenti non si avrebbero.

L'ultima scelta di design effettuata è stata quella di come gestire i file di configurazione necessari all'applicazione. Una volta deciso di utilizzare Dropbox per gestire il database, la scelta più sensata era quella di avere sull'archivio online alcuni file che sarebbero poi stati scaricati o caricati a seconda dell'occorrenza. Ora restava da decidere come realizzare questi file.

Visto che in questi file alla fine sarebbero dovuti essere salvati degli oggetti, ho scelto di utilizzare il formato XML, che permette di gestire il

contenuto di ciascun file in base ad alcune proprietà. Così facendo, sarebbe poi stato molto semplice gestire i file ed il loro contenuto dall'applicazione. Inoltre, questo tipo di file è molto leggero e questa leggerezza avrebbe poi facilitato download e upload su Dropbox. Un altro particolare è che i file XML sono in chiaro e questo permette ad un tecnico di modificare uno qualsiasi di questi file per, ad esempio, cancellare un cliente non più da salvare, come richiesto poi nelle specifiche.

2.6 Soluzioni alternative

In questa sezione analizzeremo le soluzioni alternative al problema, che si sarebbero potute implementare per quanto riguarda la realizzazione dell'applicazione.

Nel suo insieme, l'applicazione si sarebbe potuta realizzare in altri modi. Per esempio, si sarebbe potuto realizzare un sito web in HTML che, con l'ausilio di Javascript e Php, avrebbe provveduto a realizzare un rapporto di intervento a seconda dei campi compilati. Per la gestione dell'assegnamento di un intervento, si sarebbero potuti realizzare un meccanismo di login e diverse tipologie di utenti che, a seconda del tipo, avrebbero potuto o assegnare interventi e aggiungere clienti e oggetti al database, o visualizzare gli interventi assegnati.

Per quanto riguarda il database, si sarebbe potuto utilizzare un database SQL, gestito o dalla pagina web, o da una sezione server dedicata. Per quanto riguarda invece la parte Desktop, questa si sarebbe potuta realizzare con un altro linguaggio, ad esempio C#.

Anche per quanto concerne i file di configurazione, anche questi si sarebbero potuti realizzare in un altro formato, a seconda del linguaggio utilizzato e del tipo di applicazione realizzata.

Particolare importanza ha la realizzazione del database. Se non si fosse scelto di utilizzare Dropbox, si sarebbe dovuta creare in ogni caso una sezione server dell'applicazione. Questo avrebbe portato ulteriori difficoltà nello sviluppo, un ulteriore carico di lavoro e possibili altri problemi da risolvere. Altrimenti, si sarebbe comunque dovuto progettare un database SQL, con annesso studio di fattibilità, progettazione concettuale, logica e fisica. La scelta di Dropbox, invece, ha risparmiato tutti questi problemi ed inoltre ha

dato al cliente la sicurezza di non poter perdere i propri dati e la certezza di averli sempre disponibili.

2.7 Conclusione

In questo capitolo abbiamo visto innanzi tutto la descrizione del problema, dove è stata descritta in linea di massima l'applicazione richiesta dal cliente.

In seguito abbiamo visto quali sono nel dettaglio i requisiti del sistema, con un elenco dettagliato delle specifiche richieste dal cliente che l'applicazione avrebbe dovuto rispettare, e le funzionalità che l'applicazione avrebbe dovuto offrire.

Abbiamo visto poi una dettagliata descrizione della soluzione da me proposta. Questa è stata una descrizione molto dettagliata di tutte le parti che compongono il sistema.

Nella sezione successiva abbiamo analizzato tutti i motivi che hanno portato ad effettuare una determinata scelta piuttosto che un'altra. Particolare enfasi è stata posta sul perchè è stata fatta la scelta di utilizzare Dropbox per la gestione del database.

Infine abbiamo visto in quali altri modi si sarebbero potute sviluppare le diverse parti dell'applicazione, e come sarebbe potuta essere l'applicazione utilizzando un linguaggio differente e tecnologie differenti. Molta enfasi è stata posta anche su come sarebbe avvenuto lo sviluppo del database senza Dropbox.

Nel prossimo capitolo analizzeremo le fasi che hanno portato allo sviluppo dell'applicazione, analizzando anche le varie classi di cui è composta e il risultato finale.

Capitolo 3

Sviluppo dell'applicazione

In questo capitolo verrà descritto lo sviluppo dell'applicazione.

3.1 Introduzione

In questo capitolo verranno descritte tutte le fasi che riguardano lo sviluppo dell'applicazione.

Nella Sezione 2 si vedrà lo sviluppo e la creazione dei file di configurazione e si studieranno tutte le operazioni preliminari di sviluppo, come ad esempio la creazione dell'archivio Dropbox.

Nella Sezione 3 si analizzeranno tutte le fasi dello sviluppo della sezione Desktop. In questa sezione verrà fatta una breve analisi generale di tutti i processi di sviluppo, dalle prime schermate fino alle ultime funzionalità.

Nella Sezione 4 verrà raccontata tutta la fase di testing dell'applicazione, relativa alla sezione Desktop.

Nella Sezione 5, come per quanto trattato nella Sezione 3, si vedrà lo sviluppo della sezione Android dell'applicazione. Anche in questo caso saranno seguite tutte le fasi dello sviluppo.

Nella Sezione 6 vedremo tutta la fase di testing relativa alla parte Android dell'applicazione.

Nelle Sezioni 7 e 8, analizzeremo, tramite l'ausilio di diagrammi XML, le varie classi che compongono tutte le parti dell'applicazione. Oltre alle classi verrà anche analizzato il file Manifest relativo alla sezione Android.

Infine, nella Sezione 9, vedremo il risultato finale dello sviluppo, con alcune immagini che mostrano l'applicazione conclusa.

3.2 Sviluppo file e sviluppi preliminari

La prima fase relativa allo sviluppo dell'applicazione è stata quella relativa alla creazione dei file di configurazione e altri sviluppi preliminari. In questa fase ho creato i file XML di configurazione, decidendo nome e struttura di ognuno. Poi, prima di iniziare lo sviluppo dell'applicazione, ho creato il repository di bitbucket al cui interno ho caricato tutto il progetto, dividendo ciascuna parte in un branch separato. Ho inoltre creato un account dropbox, creando anche l'archivio al quale sarebbe stato possibile accedere dall'applicazione. Ho installato tutti i programmi necessari allo sviluppo sulle mie macchine, cosa che è stata resa possibile dall'utilizzo di bitbucket. Con quello, infatti, ho potuto lavorare al progetto da diverse macchine mantenendo sempre sincronizzato il progetto. Infine ho studiato la creazione del pdf relativo ai rapporti di intervento, creando già la struttura e l'impaginazione che i pdf creati dall'applicazione avrebbero dovuto rispettare. A questo punto poteva iniziare la fase di sviluppo vera e propria. Ho deciso di sviluppare le due sezioni in maniera separata: prima Desktop e poi Android.

3.3 Sviluppo sezione desktop

Iniziata la fase di sviluppo, mi sono subito concentrato sulla realizzazione delle finestre. Ho fatto questa scelta perchè volevo, prima di realizzare le funzionalità vere e proprie, presentare all'azienda un prototipo di applicazione per capire se le specifiche e i requisiti erano rimasti invariati o se ci fossero dei cambiamenti da fare.

Come prima cosa ho realizzato la schermata principale, insieme alle schermate di aggiunta di un cliente, di un oggetto e quella relativa alla creazione del rapporto di intervento. Infine, ho provveduto a realizzare le ultime due schermate che non avevo realizzato in precedenza, ossia quella di assegnamento di un intervento e quella dove visualizzare tutti i rapporti.

Quando le schermate sono state approvate, ho iniziato lo sviluppo delle funzionalità. Ho innanzitutto collegato tra loro le schermate. A questo punto ho realizzato il download dei file da Dropbox e l'upload degli stessi. Quindi ho realizzato le funzionalità di assegnamento di un intervento, di aggiunta di un cliente e di aggiunta di un oggetto. In seguito ho realizzato la funzionalità di creazione dei rapporti. Infine ho realizzato la visualizzazione dei rapporti presenti all'interno dell'archivio. Durante la realizzazione di tutte queste funzionalità, per ogni cosa che facevo, prima di proseguire, eseguivo una serie di test per essere sicuro che tutto funzionasse a dovere. Inoltre facevo rapporti continui al titolare per sapere se aveva delle modifiche da apportare e, quello che emergeva, provvedevo a modificarlo subito per evitare di dover fare in seguito delle modifiche a cascata.

3.4 Testing sezione desktop

A questo punto ho iniziato a testare l'applicazione nel suo insieme. Ho risolto i piccoli problemi che si presentavano e successivamente ho presentato l'applicazione al titolare. Sono state realizzate delle piccole modifiche di funzionamento richieste e abbiamo iniziato a testare l'applicazione insieme. Dopo vari test non sono stati riscontrati problemi.

In seguito ho presentato l'applicazione ai tecnici, insegnando loro ad usarla. Dalle loro opinioni è emerso che l'applicazione risulta semplice da utilizzare e ben costruita. Sono emersi solo un paio di problemi risolti subito (la velocità di scorrimento del cursore in alcune schermate e la possibilità di avere la maiuscola automatica all'inizio e dopo i punti nel campo dedicato alla Descrizione nella schermata di creazione dei PDF). Terminata questa fase l'applicazione è stata considerata completamente sviluppata ed è iniziata la fase di utilizzo vero e proprio. Ad oggi, dopo un utilizzo prolungato, non sono ancora emersi problemi nè elementi da modificare. Il titolare ed i tecnici sono molto soddisfatti ed utilizzano l'applicazione senza problemi.

3.5 Sviluppo sezione android

Come per quanto è successo per la sezione Desktop, appena iniziato lo sviluppo ho subito iniziato con la realizzazione delle schermate dell'applicazione. Così facendo mi è stato possibile mostrare passo per passo le varie schermate al titolare in modo da capire immediatamente se ci fosse qualcosa che andava modificato oppure no.

Ho realizzato prima la schermata principale, poi la schermata che contiene le istruzioni e i tasti per attivare e disattivare il servizio notifiche, e infine la schermata di creazione del rapporto di intervento. Infine ho realizzato la schermata all'interno della quale vengono visualizzati i PDF contenuti nell'archivio online di Dropbox.

Una volta terminata la realizzazione delle schermate e ottenuta l'approvazione del titolare a proseguire con lo sviluppo, ho iniziato la realizzazione delle funzionalità vere e proprie. Prima di tutto ho collegato tra loro le varie schermate. In seguito ho realizzato le funzionalità di rete dell'applicazione: prima il download dei file da Dropbox, poi l'upload. In un primo momento queste funzionalità erano realizzate tramite un servizio che, una volta chiamato, provava a caricare/scaricare i file desiderati. Se l'operazione non andava a buon fine, non veniva più ripetuta. In seguito ho realizzato questa funzione migliorandola, rendendola indipendente dalla connessione. Ora quando si vuole caricare/scaricare un file, viene fatto partire un servizio di Android che, in background, cerca di portare a termine l'operazione. Se non ci riesce, invia una notifica all'utente e riprova a eseguire l'operazione dieci minuti dopo. In questo modo si può stare sicuri che, una volta tornata la connessione, l'operazione verrà portata a termine. Terminato lo sviluppo di questa funzione, ho sviluppato il servizio notifiche. In seguito ho costruito la visualizzazione dei vari rapporti presenti su Dropbox, con annessa possibilità di visualizzarli e cancellarli. Diverso tempo è stato richiesto per riuscire a capire come fare a salvare in locale i PDF per poi poterli aprire e in seguito cancellarli sistematicamente per non occupare memoria. Dopo diversi test, quando queste funzionalità sono state approvate dal titolare ed ampiamente provate, ho realizzato l'ultima funzionalità, la creazione dei PDF.

3.6 Testing sezione android

Terminato lo sviluppo, è iniziata la fase di testing. Ho risolto i vari problemi che si presentavano ed ho proposto l'applicazione al titolare. Abbiamo fatto insieme gli aggiustamenti del caso e, dopo prove approfondite, sia basate solo sulla parte Android sia anche sul dialogo tra Android e Desktop, l'applicazione è stata approvata nel suo insieme.

A questo punto l'applicazione è stata presentata ai tecnici. Sono stati risolti gli ultimi problemi e imperfezioni e, dopo che anche i tecnici si sono mostrati soddisfatti dell'applicazione e l'hanno promossa, questa è stata considerata come completata. In seguito, l'applicazione ha iniziato ad essere utilizzata in maniera continua da tutti i tecnici. Dopo molti utilizzi, questi hanno pienamente confermato il loro apprezzamento nei confronti dell'applicazione, lodandone la semplicità e la completezza. Non sono ancora emersi problemi o lamentele, e l'applicazione continua ad essere utilizzata per creare i rapporti di intervento.

3.7 Descrizione delle classi desktop

3.7.1 Interfaces

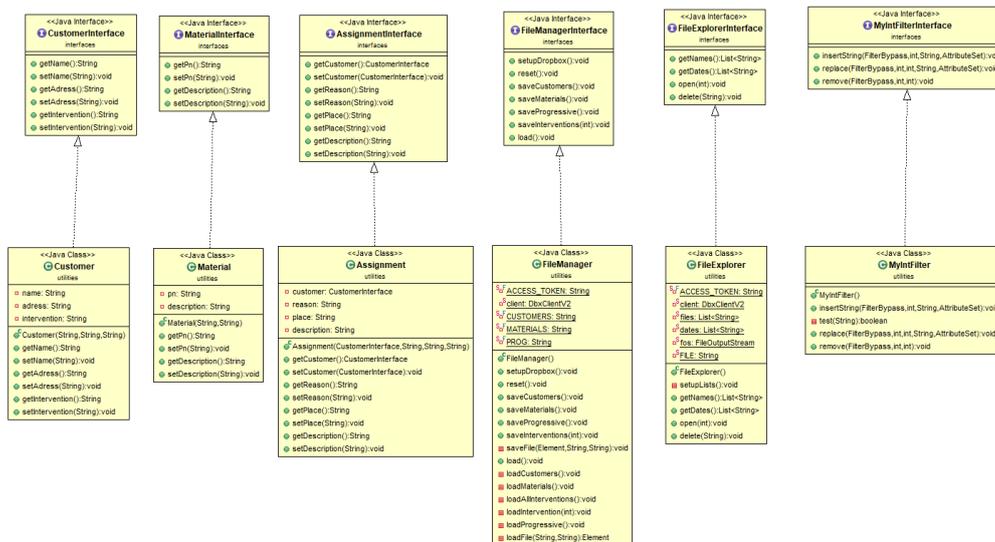


Figura 3.1: Interfaces

In questo package sono contenute le interfacce relative a tutte le classi di Utility dell'applicazione. Non sono state realizzate interfacce relative alle classi di Windows, perchè, essendo delle finestre, sono tutte molto simili tra loro ed hanno un funzionamento basilare. Le funzionalità vere e proprie dell'applicazione sono contenute nelle classi di Utility, per cui è stato scelto, per quelle classi, di realizzare prima le interfacce in modo da presentare anche al titolare prima le interfacce, per fargli avere un'idea del funzionamento dell'applicazione per poi realizzare l'implementazione vera e propria di quelle classi.

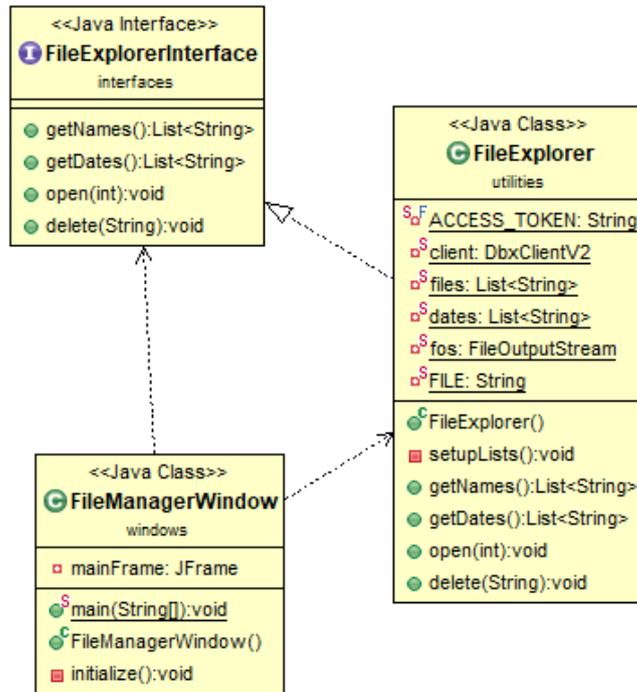
La prima è la classe *Assignment*. Questa classe gestisce l'assegnamento di un intervento ad un tecnico. Come campi questa classe contiene un Cliente (entità di tipo *Customer*) e tre stringhe, una per salvare il Motivo dell'intervento, una per salvare il Luogo di intervento ed una per salvare la Descrizione dell'intervento.

Per quanto riguarda i metodi invece, oltre al costruttore troviamo *Getters* e *Setters* dei vari campi.

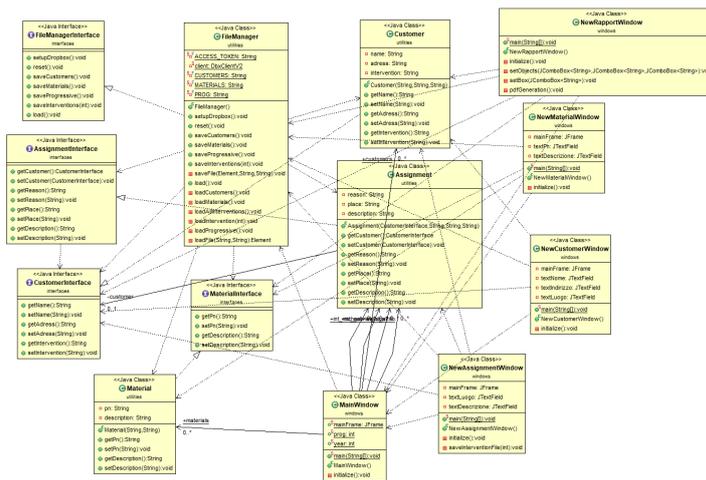
La seconda è la classe *Customer*, che rappresenta un Cliente. Per quanto riguarda i campi, questa classe ha tre campi, tutti e tre di tipo *String*. Questi servono a salvare Nome, Indirizzo e Luogo Predefinito di Intervento relativi al cliente in questione.

Come metodi invece, la classe, oltre al costruttore, contiene i *getters* e i *setters* dei vari campi.

FileExplorer e FileManager



(a) FileExplorer



(b) FileManager

Figura 3.3: FileExplorer e FileManager

La prima classe è la classe `FileExplorer`. Questa si occupa di gestire le varie funzionalità che offre la schermata dove vengono visualizzati i documenti PDF contenuti nell'archivio Dropbox, con la possibilità di visualizzarli e anche di eliminarli.

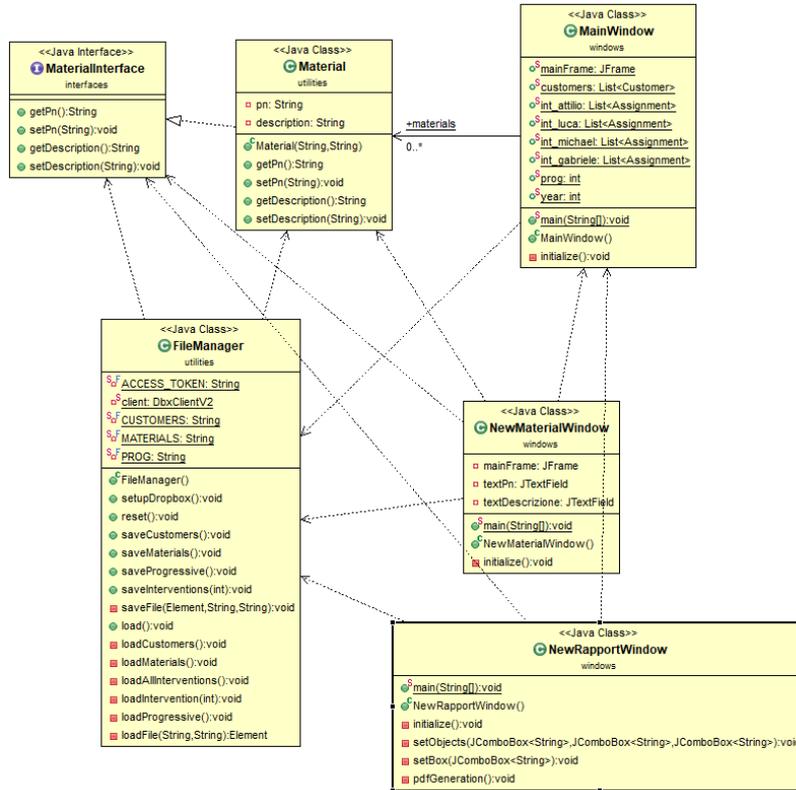
Per quanto riguarda i campi, il primo è una stringa costante che contiene il token di accesso all'archivio Dropbox. In seguito troviamo il client Dropbox che gestisce la connessione all'archivio online e le varie operazioni. Successivamente troviamo due liste di stringhe, una contenente la lista dei nomi dei file contenuti nell'archivio Dropbox e una contenente le date di caricamento dei file corrispondenti. Infine troviamo un `FileOutputStream` usato per il salvataggio locale dei PDF selezionati ed una stringa di appoggio per le operazioni della classe.

Passando ai metodi, oltre al costruttore, il primo metodo presente è `setupLists`, un metodo usato per preparare le due liste di stringhe interrogando l'archivio online. Questo metodo chiama al suo interno i due metodi successivi, che ritornano come output una lista di stringhe ognuno. Gli ultimi due metodi servono per aprire e cancellare un PDF. Il primo di questi metodi prende un intero come argomento, che corrisponde alla posizione nella lista dei nomi del file desiderato da aprire, mentre il secondo prende come argomento il nome del file da cancellare dall'archivio Dropbox.

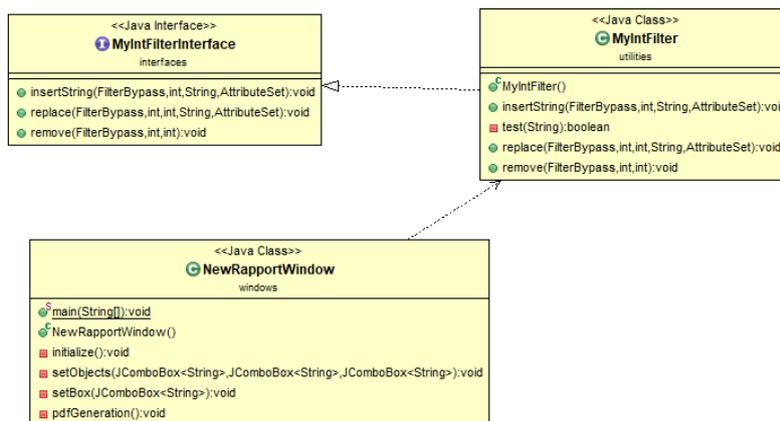
La seconda è la classe `FileManager`. Questa classe si occupa di tutte le operazioni di salvataggio dei file aggiornati su Dropbox e di scaricamento degli stessi sempre da Dropbox. Come campi troviamo in primo luogo il token di accesso a Dropbox. Oltre a tale stringa costante troviamo il client di Dropbox e altre tre stringhe, che indicano i nomi dei tre file di configurazione, ossia quello relativo ai clienti, quello relativo agli oggetti e quello relativo al numero progressivo.

Per quanto riguarda i metodi invece, oltre al costruttore, troviamo un metodo per inizializzare la connessione a Dropbox, un metodo per effettuare il reset del progressivo (riporta a 001 il numero ed aggiunge 1 all'anno), i metodi per salvare clienti, oggetti e progressivo ed infine il metodo per salvare tutti i file insieme. Poi ci sono i metodi che si occupano del caricamento, con un metodo che si occupa di caricare tutti i file necessari, uno di caricare i clienti, uno degli oggetti, uno di caricare tutti gli interventi assegnati, uno che carica un singolo intervento, uno di caricare il progressivo ed infine uno che viene chiamato per caricare un file generico.

Material e MyIntFilter



(a) Material



(b) MyIntFilter

Figura 3.4: Material e MyIntFilter

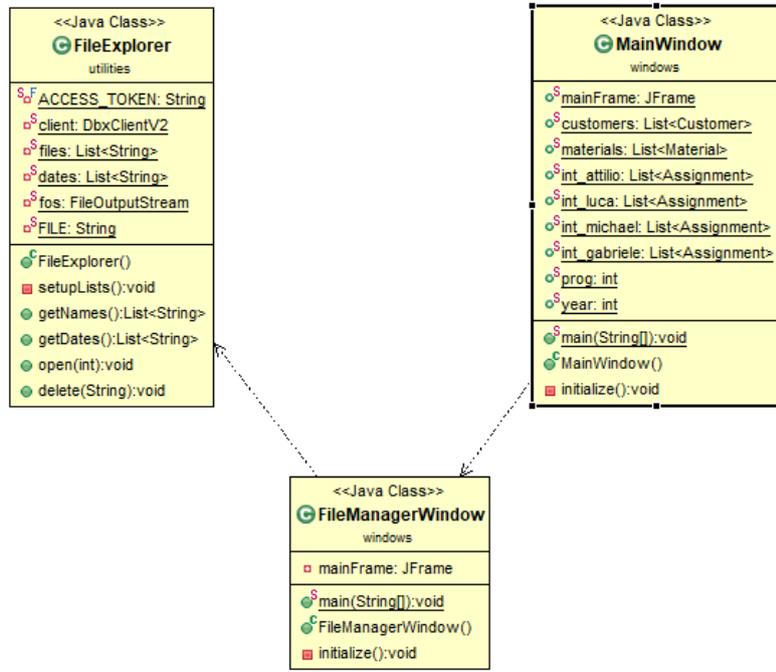
La prima è la classe `Material`, che rappresenta un oggetto utilizzabile durante un intervento. Come campi ha due stringhe che rappresentano `Product Number` e `Descrizione`, mentre come metodi, oltre al costruttore, ci sono i `getters` ed i `setters` dei due campi.

La seconda classe, `MyIntFilter`, è usata per controllare la costruzione di una stringa. Viene utilizzata per controllare che nel campo relativo alla data di intervento venga inserita solo una data in formato `GG/MM/AAAA`. Questa classe non ha campi e, per quanto riguarda i metodi, oltre al costruttore, comprende `insertString`, che permette di inserire una stringa all'interno della stringa selezionata, `test`, che testa che la stringa sia in un formato desiderato, `replace`, che rimpiazza una parte di stringa con un'altra, ed infine `remove`, che rimuove una parte di stringa.

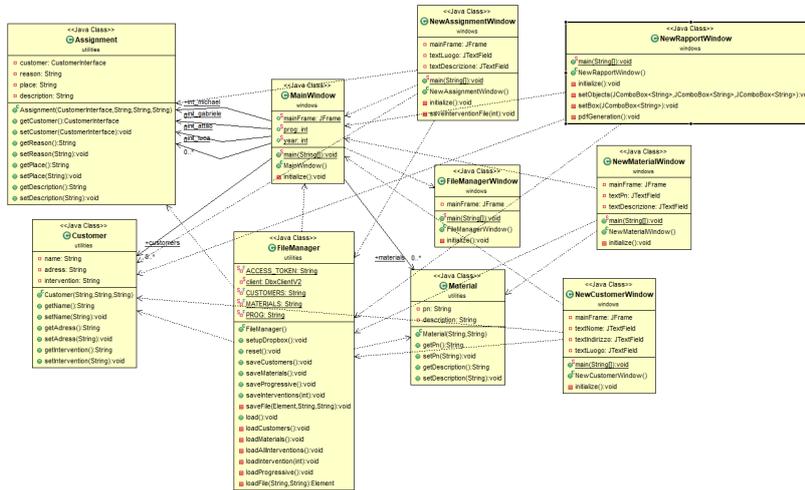
3.7.3 Windows

In questo package sono contenute tutte le finestre dell'applicazione. Ora ogni finestra verrà osservata e spiegata nel dettaglio.

FileManagerWindow e MainWindow



(a) FileManagerWindow



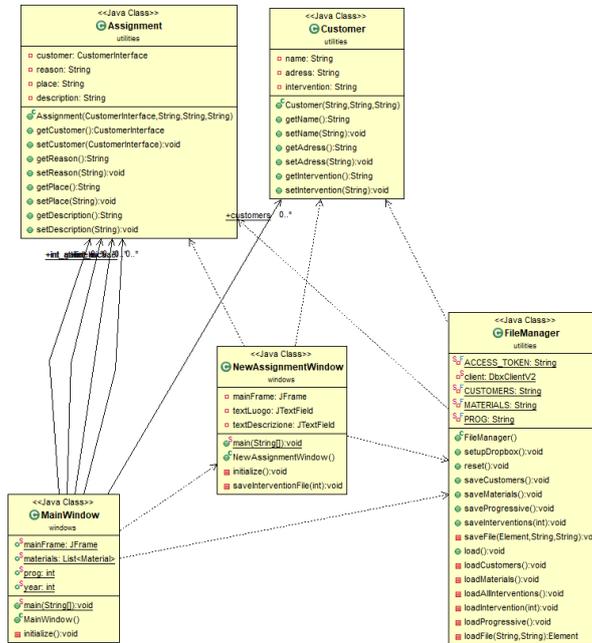
(b) MainWindow

Figura 3.5: FileManagerWindow e MainWindow

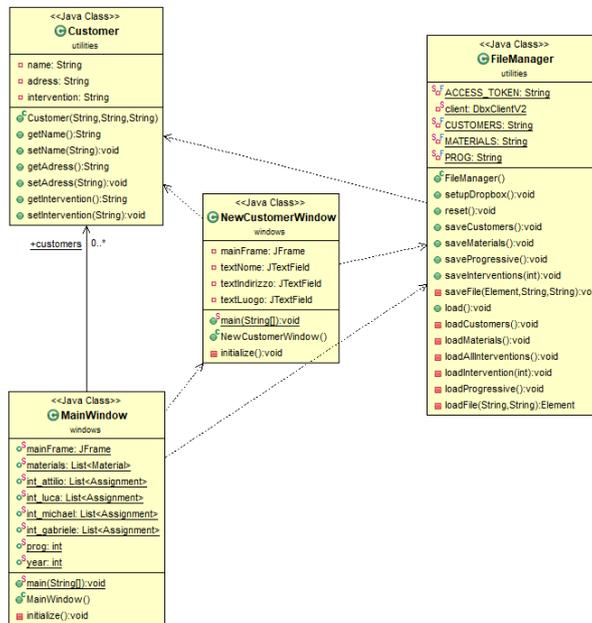
La prima classe rappresenta la finestra dentro alla quale è possibile visualizzare i rapporti di intervento presenti nell'archivio online ed eventualmente visualizzarne uno o cancellarlo. Ha come unico campo il mainFrame e come metodi il main, il costruttore ed un metodo che si occupa dell'inizializzazione della finestra.

La seconda è la finestra principale e la home dell'applicazione. Oltre ai metodi già citati per la classe precedente e che si ripetono praticamente in ogni finestra, come campi troviamo, oltre al frame, anche le liste di clienti, oggetti, una lista di interventi per ogni tecnico e due interi, uno contenente il numero progressivo ed uno l'anno.

NewAssignmentWindow e NewCustomerWindow



(a) NewAssignmentWindow



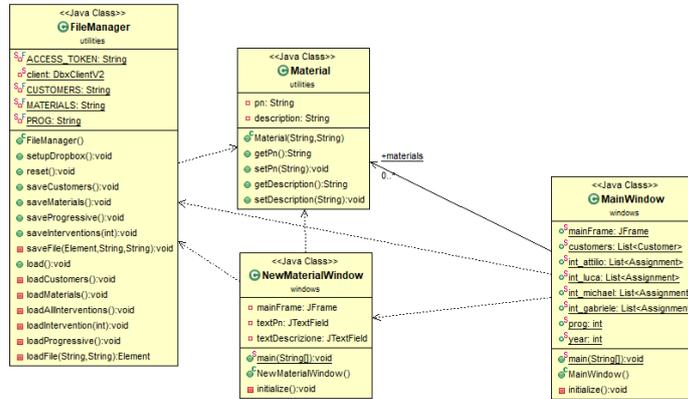
(b) NewCustomerWindow

Figura 3.6: NewAssignmentWindow e NewCustomerWindow

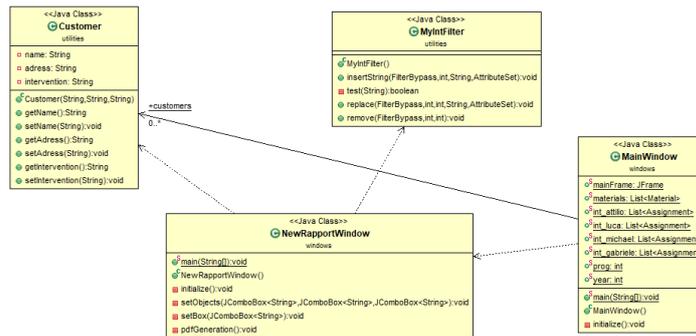
La prima è la finestra relativa all'assegnamento di un nuovo intervento ad un tecnico. Oltre alle caratteristiche viste nelle altre finestre, qui troviamo due campi, dove è possibile indicare luogo di intervento e descrizione. Per quanto riguarda i metodi, oltre ai già noti costruttore, main e initialize, troviamo il metodo `saveInterventionFile`, che ha il compito di aggiornare uno dei file contenente gli interventi relativi ad un tecnico in particolare. Questo metodo prende come parametro un intero che rappresenta il codice del tecnico a cui assegnare l'intervento.

La seconda finestra si occupa dell'inserimento nell'archivio di un nuovo cliente. Oltre alla struttura usuale, troviamo i campi nei quali è possibile indicare nome, indirizzo e luogo predefinito di intervento relativi al cliente che si sta aggiungendo.

NewMaterialWindow e NewRapportWindow



(a) NewMaterialWindow



(b) NewRapportWindow

Figura 3.7: NewMaterialWindow e NewRapportWindow

La prima classe rappresenta la finestra per l’inserimento in archivio di un nuovo oggetto. In particolare si notino i due campi dove si deve indicare Part Number e Descrizione dell’oggetto in questione.

La seconda è una delle schermate più importanti dell’applicazione: quella relativa alla creazione del documento PDF. Nell’immagine non sono stati inseriti tutti i campi per motivi di spazio, ma in questa classe sono presenti tutti i vari campi che è possibile compilare e che poi vanno a formare i dati che compongono il rapporto di intervento.

Come metodi invece, in particolare troviamo i due metodi `setupObjects` e `setBox`, che sono usati per preparare i `JComboBox` contenuti nella schermata con i vari valori che sono predefiniti e non modificabili e pertanto specificati da codice, insieme al metodo `pdfGeneration` che raccoglie tutti i dati dai vari campi, prepara il PDF secondo il formato preimpostato e si occupa di generarlo e salvarlo.

3.8 Descrizione manifest e classi android

3.8.1 Manifest

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3 package="it.urbitek.urbitek_simple">
4 <application
5     android:allowBackup="true"
6     android:icon="@mipmap/ic_urbitek"
7     android:label="Urbitek"
8     android:supportRtl="true"
9     android:theme="@style/AppTheme">
10
11     <service android:name=".UploadOperation" android:exported="false"> </service>
12     <service android:name=".DownloadOperation" android:exported="false"> </service>
13     <activity android:name=".Home">
14         <intent-filter>
15             <action android:name="android.intent.action.MAIN" />
16             <category android:name="android.intent.category.LAUNCHER" />
17         </intent-filter>
18     </activity>
19     <activity android:name=".Rapport" />
20     <activity android:name=".Info" />
21     <activity android:name=".FileManager" />
22
23     <provider
24         android:name="android.support.v4.content.FileProvider"
25         android:grantUriPermissions="true"
26         android:exported="false"
27         android:authorities="${applicationId}"
28         <meta-data
29             android:name="android.support.FILE_PROVIDER_PATHS"
30             android:resource="@xml/file_provider_paths" />
31     </provider>
32 </application>
33
34 <uses-permission android:name="android.permission.INTERNET" />
35 <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
36 </manifest>
```

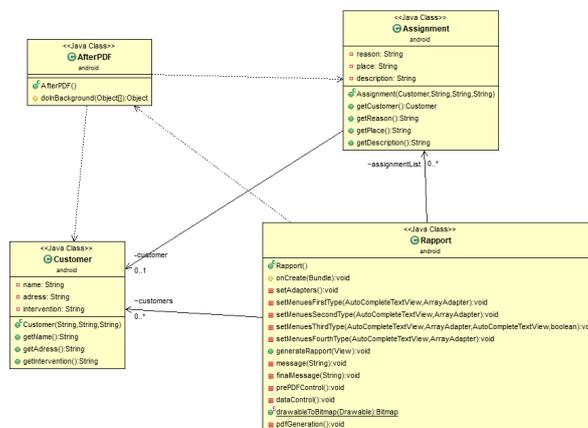
Figura 3.8: Manifest

Questo è il file Manifest dell'applicazione. In questo file ci sono tutte le informazioni di configurazione dell'applicazione. Qui vengono definiti i parametri dell'applicazione, i servizi, le attività e i permessi. E' stato anche definito un Provider per poter agire sui file presenti nella cartella dedicata all'applicazione a creata sul dispositivo al momento dell'installazione.

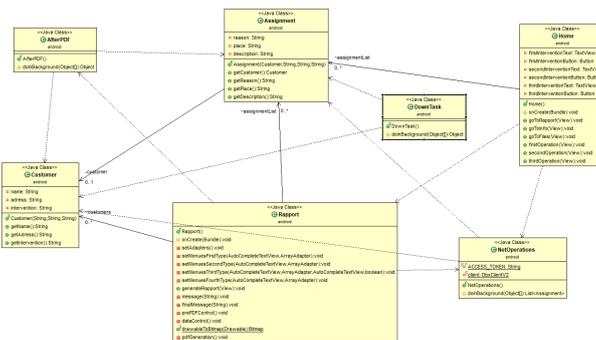
3.8.2 UtilitiesAndroid

Queste sono le classi Java che svolgono le funzionalità vere e proprie dell'applicazione. Le classi che rappresentano un Cliente, un Materiale e l'Assegnamento di un intervento ad un tecnico sono uguali a quelle utilizzate nell'applicazione Java, questo per rendere il più simile possibile la gestione dei file nelle due applicazioni.

AfterPDF e AssignmentAndroid



(a) AfterPDF



(b) AssignmentAndroid

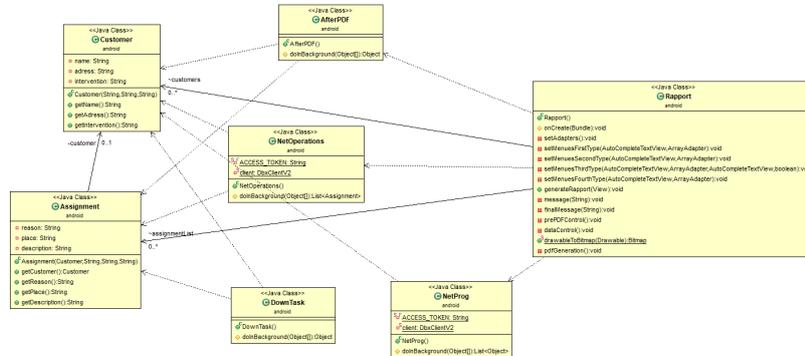
Figura 3.9: AfterPDF e AssignmentAndroid

La prima classe si occupa di eseguire tutte le operazioni che seguono la realizzazione del pdf relativo ad un rapporto di intervento. Questa funzione fa partire un task in background che si occupa di salvare il pdf in locale sul dispositivo e di caricarlo su Dropbox.

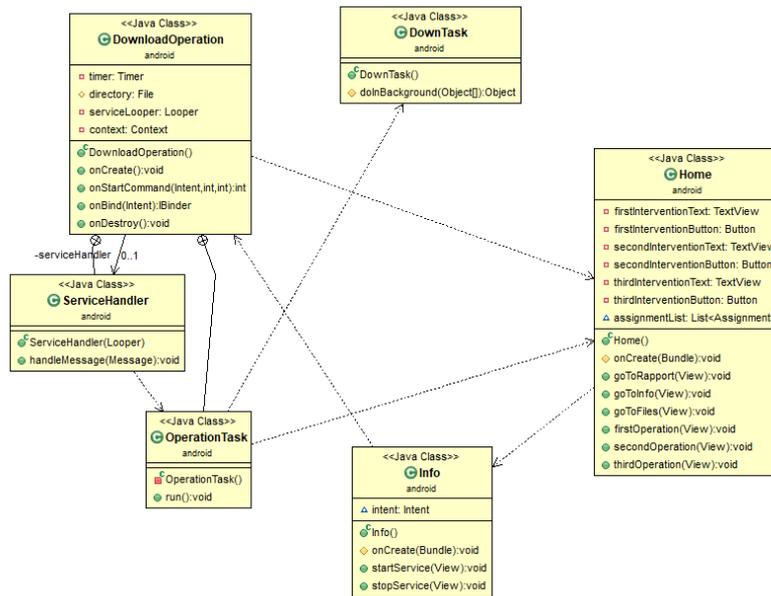
La seconda classe, come per quanto riguarda la sezione Java, rappresenta un assegnamento di un intervento. Come campi, questa classe comprende il cliente e tre stringhe, una per il motivo dell' intervento, una per il luogo di intervento ed una per la descrizione dell'intervento.

Per quanto riguarda i metodi, oltre al costruttore della classe troviamo i getter dei vari campi.

CustomerAndroid e DownloadOperation



(a) CustomerAndroid



(b) DownloadOperation

Figura 3.10: CustomerAndroid e DownloadOperation

Come per la classe `customer` su Desktop, la prima classe rappresenta un cliente. Ha tre stringhe come campi che rappresentano nome, indirizzo e luogo predefinito di intervento del cliente.

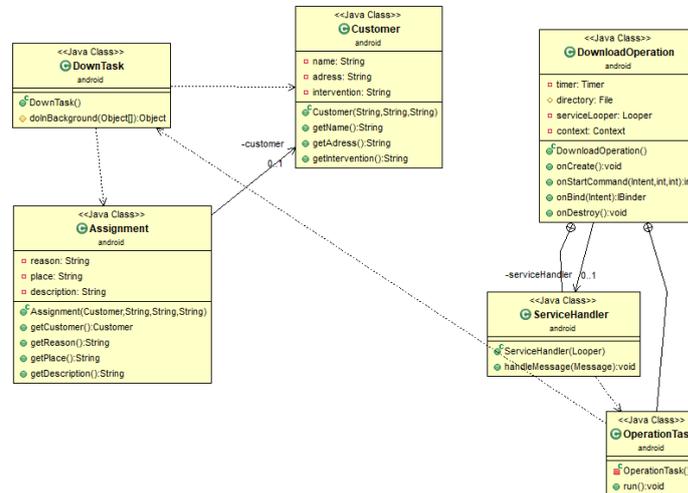
Come metodi, oltre al costruttore troviamo i getter dei campi sopra citati.

La seconda classe gestisce il servizio di notifica. Questa classe si occupa di scaricare periodicamente i file di configurazione aggiornati e, se c'è un nuovo intervento nel file di configurazione corrispondente al tecnico, invia una notifica sul dispositivo per allertare il tecnico dell'avvenuto assegnamento.

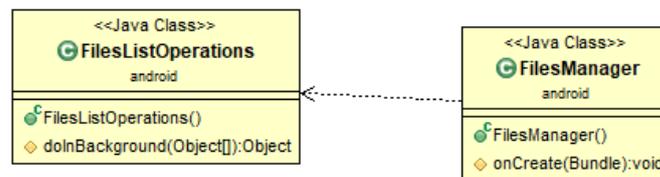
Per quanto riguarda i campi, troviamo il timer che gestisce il numero di secondi ogni quanto scaricare i file aggiornati, un file che specifica la directory nella quale scaricare i file, un `serviceLooper` ed un `serviceHandler` che si occupano di gestire il servizio android che gira in background, ed il contesto dell'applicazione, necessario al sistema per eseguire il task in background e per inviare la notifica al dispositivo.

Per quanto riguarda i metodi, oltre al costruttore abbiamo i metodi che gestiscono il comportamento della classe una volta che una sua entità viene creata, distrutta, attivata o legata ad una schermata.

DownTask e FilesListOperations



(a) DownTask



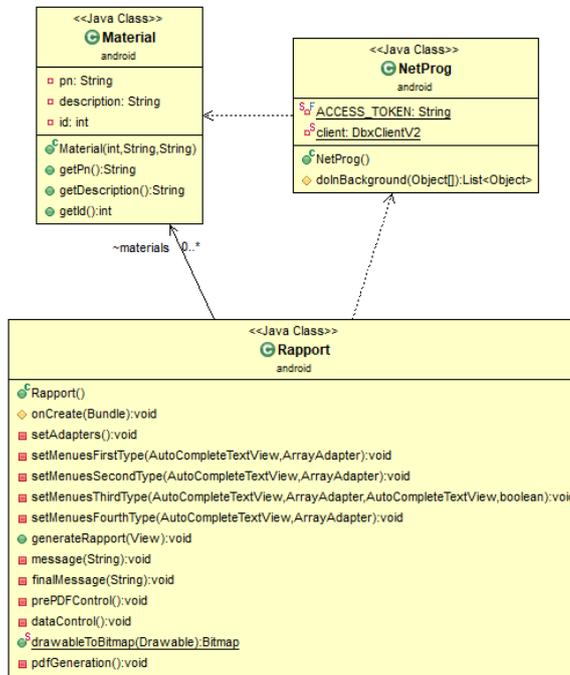
(b) FilesListOperations

Figura 3.11: DownTask e FilesListOperations

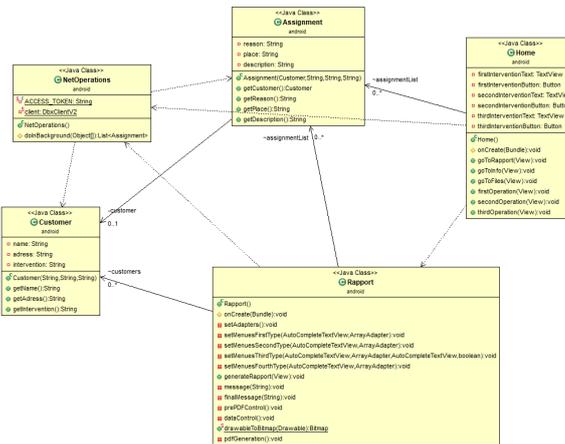
La prima classe si occupa dell'operazione di download dei file di configurazione aggiornati. Questa crea un task di android che gira in background e scarica i file necessari.

La seconda classe si occupa di gestire le operazioni necessarie a popolare la schermata dove vengono visualizzati i pdf creati dall'applicazione. Questa crea un task in background che si occupa di interrogare l'archivio per fare una lista di file da visualizzare.

MaterialAndroid e NetOperations



(a) MaterialAndroid



(b) NetOperations

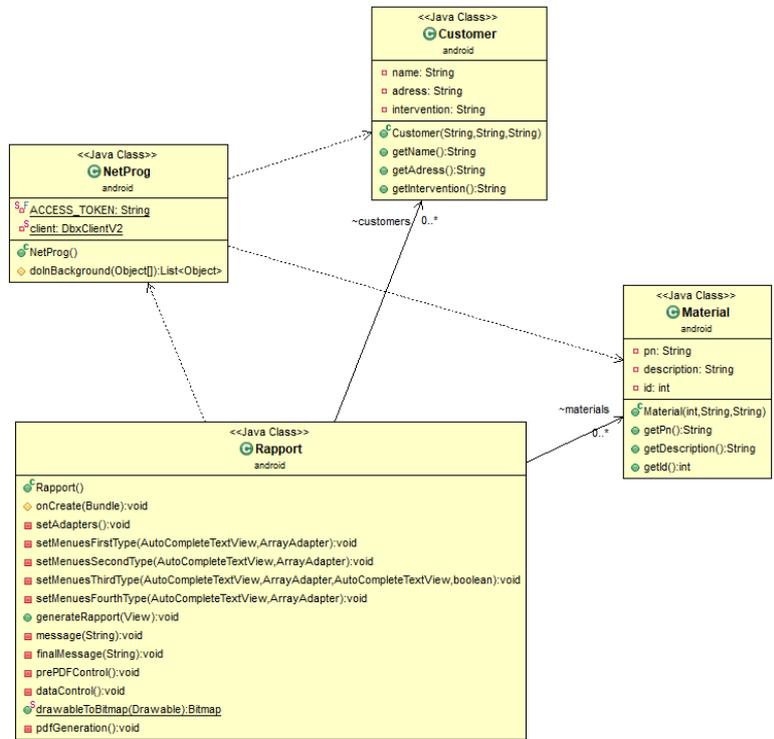
Figura 3.12: MaterialAndroid e NetOperations

La prima classe rappresenta un oggetto che può essere utilizzato durante un intervento. Per quanto riguarda i campi, questa classe comprende due stringhe, una per il product number ed una per la descrizione dell'oggetto, e un intero, che serve a salvare l'id dell'oggetto.

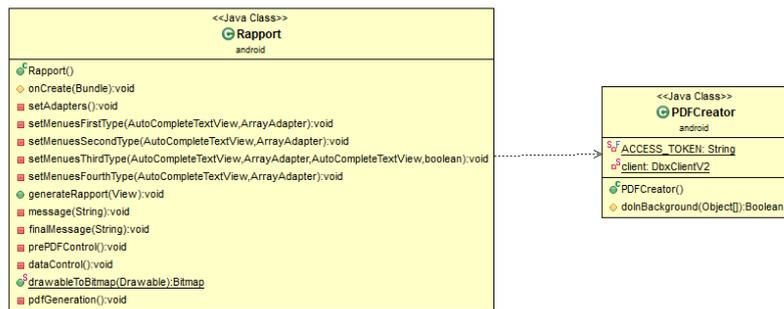
Per quanto concerne i metodi, oltre al costruttore troviamo i getter dei campi.

La seconda classe si occupa di gestire la connessione con Dropbox da parte dell'applicazione. Questa fa partire un task android in background che, in background appunto, si occupa delle operazioni di connessione con il server. Troviamo due campi, una stringa costante che rappresenta il token di accesso a Dropbox ed il client che gestisce la connessione. Oltre alla gestione della connessione, questa classe si occupa del download da Dropbox dei file di configurazione relativi a progressivo, clienti ed oggetti.

NetProg e PDFCreator



(a) NetProg



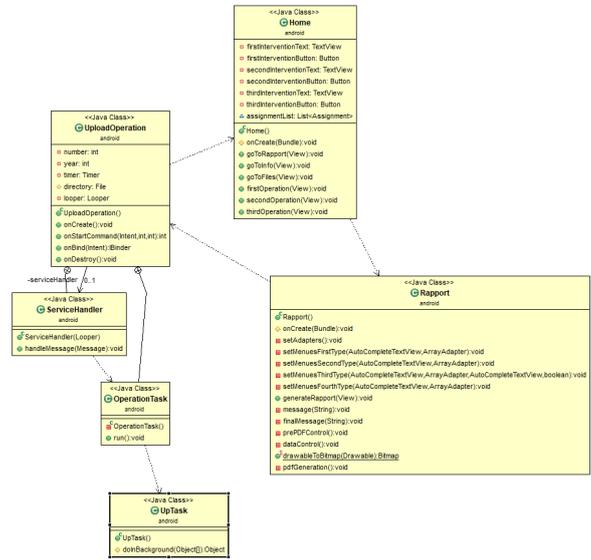
(b) PDFCreator

Figura 3.13: NetProg e PDFCreator

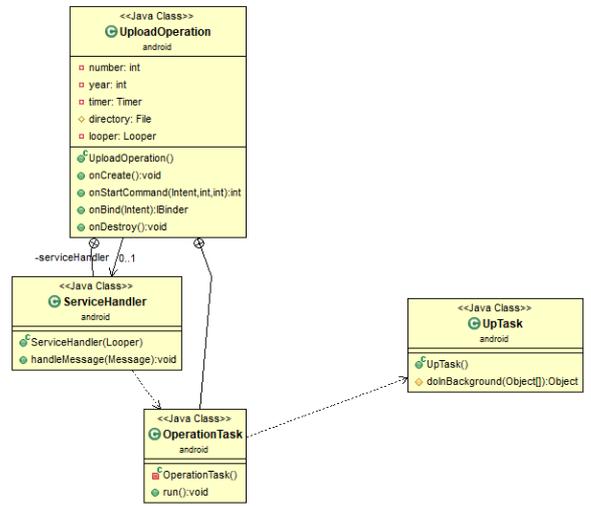
Anche la prima classe, come la precedente, gestisce l'accesso a Dropbox. A differenza della precedente classe, oltre a gestire la connessione a Dropbox, questa si occupa del download del file relativo agli interventi assegnati al tecnico in questione. Ogni applicazione, per quanto riguarda Android, ha un codice univoco che identifica un tecnico in particolare. Tramite questo codice, ogni applicazione Android scarica i dati relativi agli interventi assegnati solo del tecnico con il codice corrispondente.

La seconda classe si occupa della creazione dei file pdf relativi ai rapporti di intervento. Tutto il processo di creazione viene eseguito da un task in background.

UploadOperation e UpTask



(a) UploadOperation



(b) UpTask

Figura 3.14: UploadOperation e UpTask

La prima classe gestisce l'upload di file in caso questo sia fallito la prima volta. Dopo la creazione di un pdf, deve essere caricato anche il file aggiornato relativo al numero progressivo. Se l'operazione di upload fallisce la prima volta, viene creato un servizio in background che, ad un intervallo di tempo fissato, cerca di caricare pdf e file aggiornati su Dropbox.

Per quanto riguarda i campi, sono compresi numero progressivo e anno, il timer che specifica l'intervallo di tempo, la directory dove sono i file da caricare, un Looper ed un serviceHandler che si occupano di gestire il servizio android.

Come metodi, oltre al costruttore troviamo i metodi che gestiscono la creazione, la distruzione, l'attivazione ed il binding del task.

La seconda classe gestisce l'upload dei file su dropbox. L'upload viene eseguito su un task che opera in background sul dispositivo.

La prima è la schermata principale dell'applicazione.

Per quanto riguarda i campi, troviamo un campo testuale ed un bottone per ciascun intervento assegnato. Il massimo di interventi visualizzabili è 3 ed il minimo è 0. A seconda del numero di interventi assegnati, i campi di testo e i bottoni corrispondenti sono visualizzati o meno. Oltre ai campi appena citati, troviamo una lista al cui interno vengono salvati tutti gli interventi assegnati al tecnico.

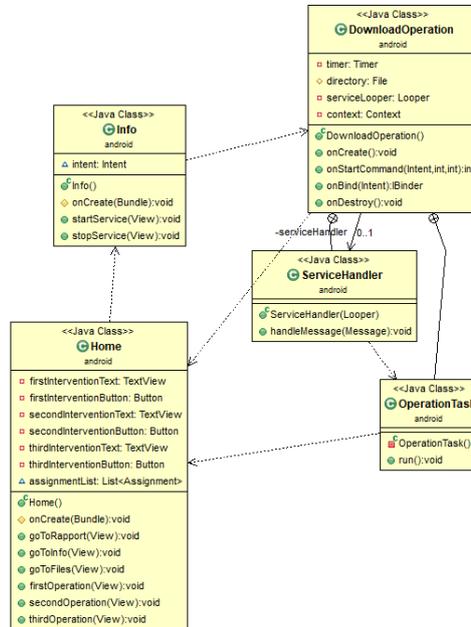
Per quanto riguarda i metodi, oltre al costruttore, troviamo un metodo, comune a tutte le classi relative alle schermate, che viene chiamato alla creazione della schermata e si occupa dell'inizializzazione della stessa. Oltre a questi troviamo i 3 metodi per passare alle altre schermate e i 3 metodi che portano alla schermata di creazione di un rapporto relativo ad uno degli interventi assegnati.

La seconda classe gestisce la schermata relativa alla creazione di un rapporto di intervento.

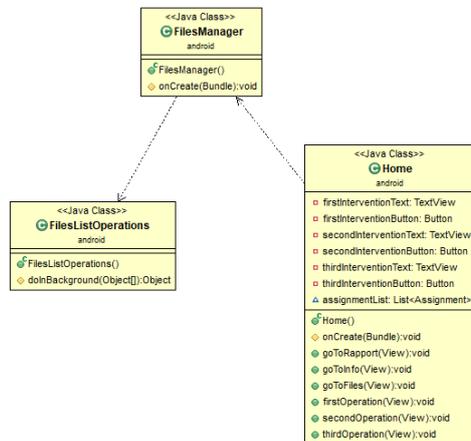
Per motivi di spazio, i campi non sono stati indicati nel diagramma UML. In ogni caso, questa classe come campi ha tutti i vari campi che il tecnico deve compilare nella creazione del rapporto di intervento.

Per quanto riguarda i metodi, oltre al costruttore troviamo cinque metodi che si occupano di preparare le `AutoCompleteTextView`, dei campi che si possono completare compilandoli a mano o scegliendo uno tra i suggerimenti che compaiono dopo l'immissione del primo carattere nel campo. In seguito troviamo il metodo che viene chiamato per creare il rapporto di intervento. Poi troviamo due metodi che si occupano di gestire i box di messaggio che appaiono in caso alcuni campi non siano stati compilati o siano stati compilati con valori non ammessi. In seguito abbiamo due metodi che si occupano di eseguire controlli su alcuni campi (ad esempio che la data sia stata immessa nel formato corretto). Infine troviamo il metodo che genera il pdf nel formato desiderato ed un metodo che si occupa di generare il logo dell'azienda da inserire nel rapporto di intervento in formato bitmap.

Info e FileManager



(a) Info



(b) FileManager

Figura 3.16: Info e FileManager

La prima classe rappresenta la schermata delle istruzioni. In questa schermata sono presenti anche due pulsanti che permettono di attivare o disattivare il servizio notifiche.

La seconda classe gestisce la schermata dove viene visualizzata la lista dei file pdf contenuti nell'archivio Dropbox.

3.9 Risultato dello sviluppo

In questa sezione analizzeremo l'applicazione finita, andando a descrivere le schermate che compongono le due sezioni, quella Desktop e quella Android, le parti di cui si compongono e la loro funzione.



(a) Java Home

The screenshot shows a web browser window titled "URBITEK" displaying a form for creating a new assignment. At the top of the form, there is a red instruction: "COMPILARE ATTENTAMENTE TUTTI I CAMPI". Below this, the form consists of several input fields and dropdown menus, each with a label above it: "CLIENTE" (with a dropdown menu), "MOTIVO DELL'INTERVENTO" (with a dropdown menu), "LUOGO INTERVENTO (MODIFICARE SE NECESSARIO)" (with a text input field), "DESCRIZIONE" (with a text input field), and "TECNICO" (with a dropdown menu). At the bottom of the form is a red button with white text labeled "ASSEGNA".

(b) New Assignment

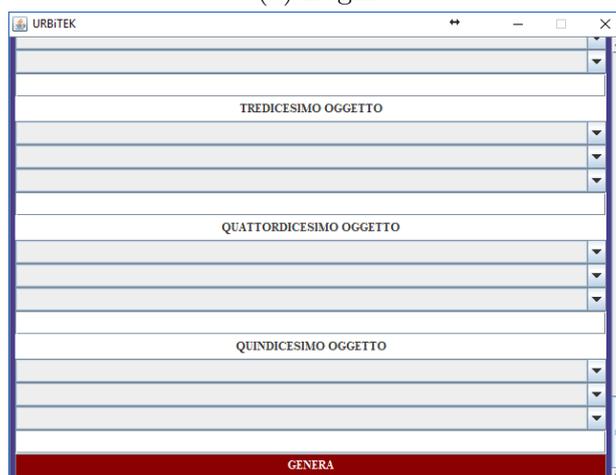
Figura 3.17: Java Home and New Assignment

La prima è la schermata Home della sezione Java. In questa schermata troviamo il logo dell'azienda ed una serie di pulsanti. Questi pulsanti servono ad assegnare un nuovo intervento, creare un nuovo rapporto di intervento, inserire un nuovo cliente, inserire un nuovo oggetto, accedere alla schermata dove vengono visualizzati i rapporti di intervento salvati nell'archivio Dropbox e a resettare il numero progressivo.

La seconda è la schermata che permette di assegnare un nuovo intervento ad un tecnico. Durante l'assegnamento bisogna selezionare un cliente tra quelli presenti nel database, selezionare un motivo di intervento tra i motivi preimpostati, scrivere una breve descrizione dell'intervento e selezionare il tecnico a cui assegnare l'intervento dall'elenco dei tecnici. Per quanto riguarda il luogo, questo viene completato automaticamente alla selezione del cliente ma è sempre possibile modificarlo se necessario.



(a) Begin



(b) End

Figura 3.18: New Rapport Java

Queste due schermate rappresentano l'inizio e la fine della schermata che permette la creazione del rapporto di intervento. In questa schermata troviamo tutta una serie di campi da compilare. I campi sono o campi a scelta tra valori preimpostati o valori del database, oppure campi di testo.

URBITEK

DATI CLIENTE

NOME

INDIRIZZO

LUOGO PREDEFINITO DI INTERVENTO

AGGIUNGI

(a) New Customer

URBITEK

DATI OGGETTO

PART NUMBER

DESCRIZIONE

AGGIUNGI

(b) New Material

Figura 3.19: New Customer and New Material

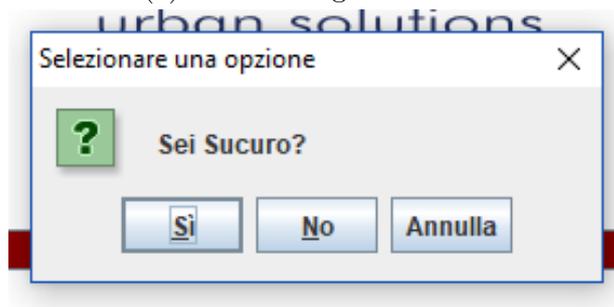
La prima schermata permette l'aggiunta di un nuovo cliente al database. In questa schermata bisogna inserire nome, indirizzo e luogo predefinito di intervento relativi al cliente da inserire nel sistema.

Nella seconda schermata è possibile inserire un nuovo oggetto nel database. Per ogni oggetto bisogna inserire part number e descrizione.



File Name	Upload Date	Actions
/urbitek_rapporto14_2018.pdf	26-3-2018 02:58:25	VISUALIZZA ELIMINA
/urbitek_rapporto15_2018.pdf	17-4-2018 09:12:30	VISUALIZZA ELIMINA
/urbitek_rapporto16_2018.pdf	17-4-2018 09:12:34	VISUALIZZA ELIMINA
/urbitek_rapporto17_2018.pdf	24-4-2018 05:21:28	VISUALIZZA ELIMINA
/urbitek_rapporto18_2018.pdf	11-5-2018 03:10:47	VISUALIZZA ELIMINA
/urbitek_rapporto19_2018.pdf	23-5-2018 04:59:24	VISUALIZZA ELIMINA
/urbitek_rapporto20_2018.pdf	28-5-2018 12:08:27	VISUALIZZA ELIMINA
/urbitek_rapporto21_2018.pdf	29-5-2018 02:46:06	VISUALIZZA ELIMINA
/urbitek_rapporto22_2018.pdf	15-6-2018 10:40:15	VISUALIZZA ELIMINA
/urbitek_rapporto23_2018.pdf	15-6-2018 10:52:51	VISUALIZZA ELIMINA
/urbitek_rapporto24_2018.pdf	15-6-2018 11:42:20	VISUALIZZA ELIMINA

(a) Files Management Java



(b) Progressive Reset

Figura 3.20: Files Management Java and Progressive Reset

La prima è la schermata che permette di visualizzare i rapporti di intervento salvati nell'archivio Dropbox. Per ogni intervento viene visualizzato il nome, la data di caricamento nell'archivio, e viene data la possibilità di visualizzarlo o di eliminarlo definitivamente dall'archivio Dropbox.

La seconda è la finestra di conferma che compare alla pressione del tasto che permette di resettare il numero progressivo. Solo dopo la conferma il progressivo viene resettato, e l'applicazione chiusa automaticamente.

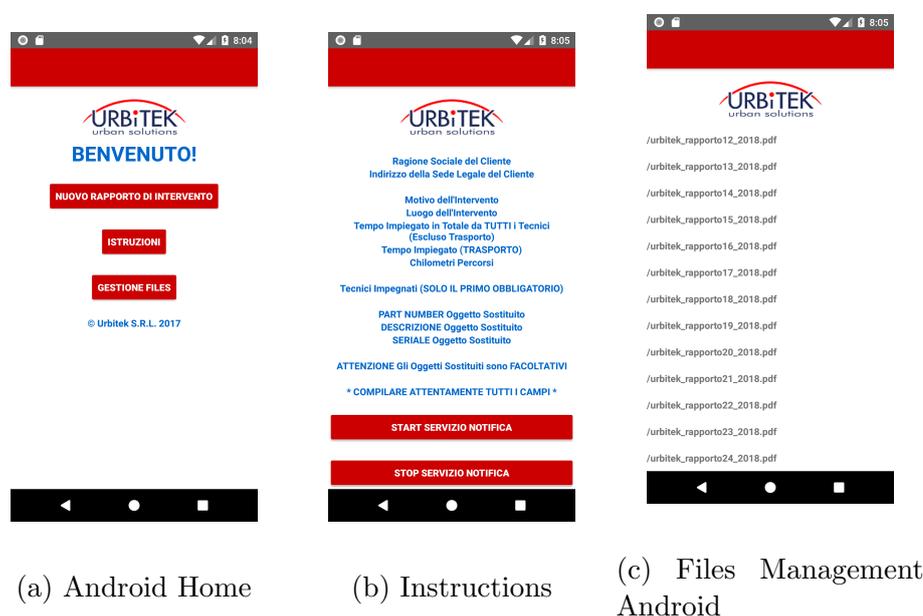


Figura 3.21: Android Home, Instructions and Files Management Android

La prima è la home della sezione Android. In questa schermata, in caso ad un tecnico sia assegnato un intervento, questo viene mostrato dopo il messaggio di benvenuto. Per ogni intervento assegnato nella home vengono visualizzati la descrizione e un pulsante che permette di accedere alla schermata di creazione del rapporto di intervento relativo a tale intervento. Nella home sono sempre presenti il tasto per accedere alla schermata di creazione di un rapporto di intervento, il tasto istruzioni, che porta alla schermata delle istruzioni nella quale è anche possibile attivare e disattivare il servizio notifica, e il tasto gestione file che porta alla schermata nella quale vengono visualizzati i rapporti contenuti nell'archivio Dropbox.

La seconda è la schermata relativa alle istruzioni. In questa schermata, oltre ad alcune istruzioni relative alla compilazione dei campi per la creazione di un rapporto di intervento, sono presenti i due pulsanti che servono ad attivare o a disattivare il servizio notifiche. Una volta attivato, anche se l'applicazione viene chiusa, ogni dieci minuti i file di configurazione aggiornati verranno scaricati automaticamente e, se è stato assegnato un nuovo intervento al tecnico, viene inviata una notifica al dispositivo per avvertire il possessore dell'avvenuto assegnamento. Mentre il servizio è attivo, una notifica permanente avvisa l'utente che il servizio è per l'appunto attivo.

La terza schermata mostra ogni rapporto di intervento contenuto nell'archivio Dropbox. Di ogni rapporto viene mostrato solo il nome. Cliccando su un rapporto, si apre un box di scelta che permette di scegliere quale operazione eseguire sul rapporto, se visualizzarlo, eliminarlo dall'archivio o chiudere il box senza eseguire nessuna operazione.

The figure consists of two side-by-side screenshots of an Android application interface. Both screens feature a red header bar at the top with the 'URBiTEK urban solutions' logo. The left screenshot, labeled '(a) Begin', shows a form with two main sections: 'DATI CLIENTE' and 'DATI INTERVENTO'. Under 'DATI CLIENTE', there are input fields for 'NOME' and 'INDIRIZZO'. Under 'DATI INTERVENTO', there are input fields for 'DATA', 'MOTIVO', 'LUOGO', and 'TIPOLOGIA'. The right screenshot, labeled '(b) End', shows a form with several input fields: '(DESCRIZIONE)', '(SERIALE)', and '(PART NUMBER)'. It also includes a section titled 'QUINDICESIMO OGGETTO' with three checkboxes: 'SOSTITUITO', 'FORNITO', and 'RITIRATO'. Below these fields is a red button labeled 'GENERA RAPPORTO'. Both screenshots show the standard Android navigation bar at the bottom.

(a) Begin

(b) End

Figura 3.22: New Rapport Android

Questa è la schermata di creazione di un nuovo rapporto di intervento relativa alla sezione Android. In questa schermata sono presenti tutta una serie di campi da compilare tramite i quali poi viene generato il rapporto di intervento. Tutti i campi sono campi di testo nei quali viene dato un suggerimento all'utente una volta inserito il primo carattere. I suggerimenti vengono forniti andando a pescare o da alcuni valori preimpostati o dai valori contenuti nel database, a seconda di quale campo si sta compilando.

3.10 Conclusione

In questo capitolo abbiamo visto innanzitutto quali sono stati gli sviluppi preliminari a quello dell'applicazione (sviluppo dei file di configurazione, creazione dell'archivio dropbox e del repository di bitbucket ed installazione dei programmi necessari). In seguito abbiamo visto quali sono state le fasi dello sviluppo dell'applicazione nella sezione Desktop, dalle prime schermate alle ultime funzionalità. Successivamente abbiamo visto quali sono stati i test eseguiti sull'applicazione nella parte Desktop. Abbiamo visto poi sviluppo e testing della sezione Android, in tutte le loro parti. Sono state trattate in seguito le classi di entrambe le sezioni dell'applicazione, con particolare dettaglio sul funzionamento di ognuna. E' stato trattato anche il file Manifest di Android. Infine, abbiamo visto qual'è il risultato dello sviluppo dell'applicazione ed abbiamo analizzato le varie schermate delle due sezioni, descrivendone il funzionamento e le parti che le compongono.

Penso che la soluzione descritta in questa tesi ed adottata dall'azienda per gestire gli interventi sia una soluzione utilizzabile in diversi altri ambiti. Questa soluzione si adatta bene a molti compiti, perchè il vantaggio di utilizzare Java è notevole. Questo è un linguaggio molto potente e flessibile, e permette di realizzare applicazioni di ogni tipo. Il fatto poi di avere l'applicazione divisa in due sezioni, Desktop e Android, permette di poter realizzare un'applicazione che si adatta al tipo di utilizzo che poi ne si andrà a fare. Per un'azienda, infatti, penso che risulti sempre molto comodo effettuare la distinzione tra le funzioni a cui i tecnici e gli operai possono attingere e le funzioni di gestione ed amministrazione. Sviluppando un'applicazione in questo modo, si possono tenere separate le due cose e si può adattare l'applicazione in base all'uso che se ne andrà a fare.

Ad esempio, se la realtà aziendale comprende dei tecnici che devono effettuare determinate operazioni, questi si troveranno sicuramente, a mio avviso, più comodi ad effettuarle via dispositivi mobile, come Smartphone o Tablet, piuttosto che con un Computer, e viceversa un segretario o un dirigente sicuramente avrà più funzioni a sua disposizione e, lavorando spesso da un ufficio e con un computer, si troverà più comodo a lavorare su di esso.

In ogni caso, mi sento di consigliare l'utilizzo delle funzionalità offerte da Dropbox per lo sviluppo di applicazioni che richiedano la gestione di dati che devono essere presenti in rete e sempre disponibili ed accessibili. Questo tipo di soluzione risulta sicuramente più sicura, efficiente, stabile e

semplice di qualsiasi altra. Infatti, questa soluzione, utilizzata nello sviluppo di applicazioni che debbano gestire file che devono essere presenti in rete, facilita e riduce notevolmente lo sviluppo, rende più sicura l'applicazione e fornisce anche un valore aggiunto all'applicazione. Dropbox è infatti pressochè universalmente conosciuto e utilizzato da un vasto numero di aziende per la gestione dei file sensibili che non si vogliono né si possono perdere, quindi il suo inserimento in una qualsiasi applicazione, che va realizzata per un'azienda che ha la necessità di operare con file in rete, ne aumenta sicuramente il valore, soprattutto se l'azienda in questione fa già un largo uso di questa tecnologia.

In generale, quindi, se si deve realizzare un'applicazione che deve gestire dei file, che devono necessariamente sempre essere presenti in rete, e non ha requisiti particolari in termini di tempo di download e upload dei file, a mio avviso è consigliabile l'utilizzo di questa tecnologia, per tutti i vantaggi che offre.

Capitolo 4

Ringraziamenti

Vorrei innanzitutto ringraziare il Professor Mirko Viroli, che mi ha ispirato e aiutato nella preparazione di questa tesi e senza il quale questo lavoro non sarebbe stato possibile.

Ringrazio poi i miei compagni di corso, che mi hanno sempre aiutato e sostenuto durante tutto il lavoro sulla tesi, in particolare Christian e Maicol.

Un grande ringraziamento anche a tutti i membri di Urbitek, l'azienda che mi ha assunto e per la quale ho realizzato l'applicazione oggetto della tesi. Un grazie particolare al mio collega Luca che mi ha sempre spronato a terminare questo lavoro.

Un ringraziamento molto speciale alla mia ragazza Silvia, senza la quale sicuramente non sarei riuscito a portare a termine questo risultato.

Un ringraziamento alla mia famiglia e a quella di Silvia, che considero ormai una seconda famiglia, per avermi sempre aiutato durante il lavoro su questo progetto. Ringrazio infine mio zio Massimo, che ha fatto di tutto per farmi arrivare a questo risultato e che mi ha sempre motivato e sostenuto.

Bibliografia

- [1] Documentazione api dropbox per java. <https://dropbox.github.io/dropbox-sdk-java/api-docs/v2.1.x/>.
- [2] Home page di android studio. <https://developer.android.com/studio/>.
- [3] Home page di bitbucket. <https://bitbucket.org/>.
- [4] Home page di dropbox. <https://www.dropbox.com/it/>.
- [5] Home page di dropbox per sviluppatori. <https://www.dropbox.com/developers>.
- [6] Home page di eclipse. <https://www.eclipse.org/>.
- [7] Home page di java. <https://java.com/it/>.
- [8] Sito internet di launch4j. launch4j.sourceforge.net/.
- [9] Sito internet di tortoise hg. <https://www.mercurial-scm.org/>.
- [10] Sito internet di urbitek s.r.l. <http://www.urbitek.it/>.
- [11] Sito internet relativo a miktex. <https://miktex.org/>.
- [12] Sito internet relativo alla libreria itextpdf. <https://itextpdf.com/>.

Elenco delle figure

2.1	Architecture Diagram	18
3.1	Interfaces	28
3.2	Assignment e Customer	29
3.3	FileExplorer e FileManager	31
3.4	Material e MyIntFilter	33
3.5	FileManagerWindow e MainWindow	35
3.6	NewAssignmentWindow e NewCustomerWindow	37
3.7	NewMaterialWindow e NewRapportWindow	39
3.8	Manifest	40
3.9	AfterPDF e AssignmentAndroid	41
3.10	CustomerAndroid e DownloadOperation	43
3.11	DownTask e FilesListOperations	45
3.12	MaterialAndroid e NetOperations	46
3.13	NetProg e PDFCreator	48
3.14	UploadOperation e UpTask	50
3.15	Home e Rapport	52
3.16	Info e FilesManager	54
3.17	Java Home and New Assignment	56
3.18	New Rapport Java	58
3.19	New Customer and New Material	59
3.20	Files Management Java and Progressive Reset	60
3.21	Android Home, Instructions and Files Management Android	61
3.22	New Rapport Android	62