

Images to PDF

ODD

Object Design Document

|  |  |
| --- | --- |
| Rif. | G12\_ODD\_ver.1.00 |
| Versione | 1.00 |
| Data | 28/01/22 |
| Destinatario | Carmine Gravino |
| Team | G12 |
|  |  |

**Storico Revisioni**

|  |  |  |  |
| --- | --- | --- | --- |
| **Cronologia modifiche** | | | |
| *Data* | *Versione* | *Descrizione* | *Autore* |
| *19/01/22* | *0.01* | *Prima stesura* | *EC, GD* |
| *22/01/22* | *0.02* | *Stesura Capitolo 1* | *MG* |
| *23/01/22* | *0.03* | *Packages* | *MG, FP* |
| *24/01/22* | *0.04* | *Revisione packages* | *Tutto il team* |
| *25/01/22* | *0.05* | *Class interfaces* | *MG* |
| *26/01/22* | *0.06* | *Revisione class interfaces* | *Tutto il team* |
| *27/01/22* | *0.07* | *Design patterns e Glossario* | *FP, MG* |
| *28/01/22* | *1.00* | *ODD completato* | *Tutto il team* |

**1 Introduzione**

Image to PDF ha come scopo quello di semplificare le operazioni più comuni normalmente effettuate su un PDF. Che si tratti di esigenze lavorative o no, tutti gli utenti, anche con scarse conoscenze degli smartphone potranno in tutta facilità effettuare tutte le operazioni disponibili all’interno dell’app.

In questa prima sezione del documento, saranno descritti le linee guida per la fase di implementazione, la nomenclatura, la documentazione e le convenzioni sui formati.

**1.1 Object design goals (Trade-offs)**

**Prestazioni vs Costi:**

Il sistema, un prodotto open source, è realizzato con basso budget, minimizzando i costi, riuscendo però a soddisfare requisiti prestazionali soddisfacenti.

**Spazio vs Tempo di risposta:**

Il sistema presenta ottimi tempi di risposta, che però peggiorano in casi di modifica o creazione di PDF di dimensioni considerevoli.

**Interfaccia vs Facilità d’utilizzo:**

Le interfacce risultano molto semplici e non particolarmente dettagliate, questo per permettere l’utilizzo del software anche ad utenti poco esperti nel campo informatico.

**1.2 Linee guida per la documentazione dell’interfaccia**

Le linee guida includono una lista di regole che gli sviluppatori devono rispettare durante la progettazione delle interfacce. Si è fatto dunque riferimento alla convenzione java nota come Java Coding Conventions.

**1.3 Definizioni, acronimi e abbreviazioni**

**MG =** Marco Gallo

**GD =** Gionata D’Amore

**EC =** Emanuele Castellano

**FP =** Federico Pomposelli

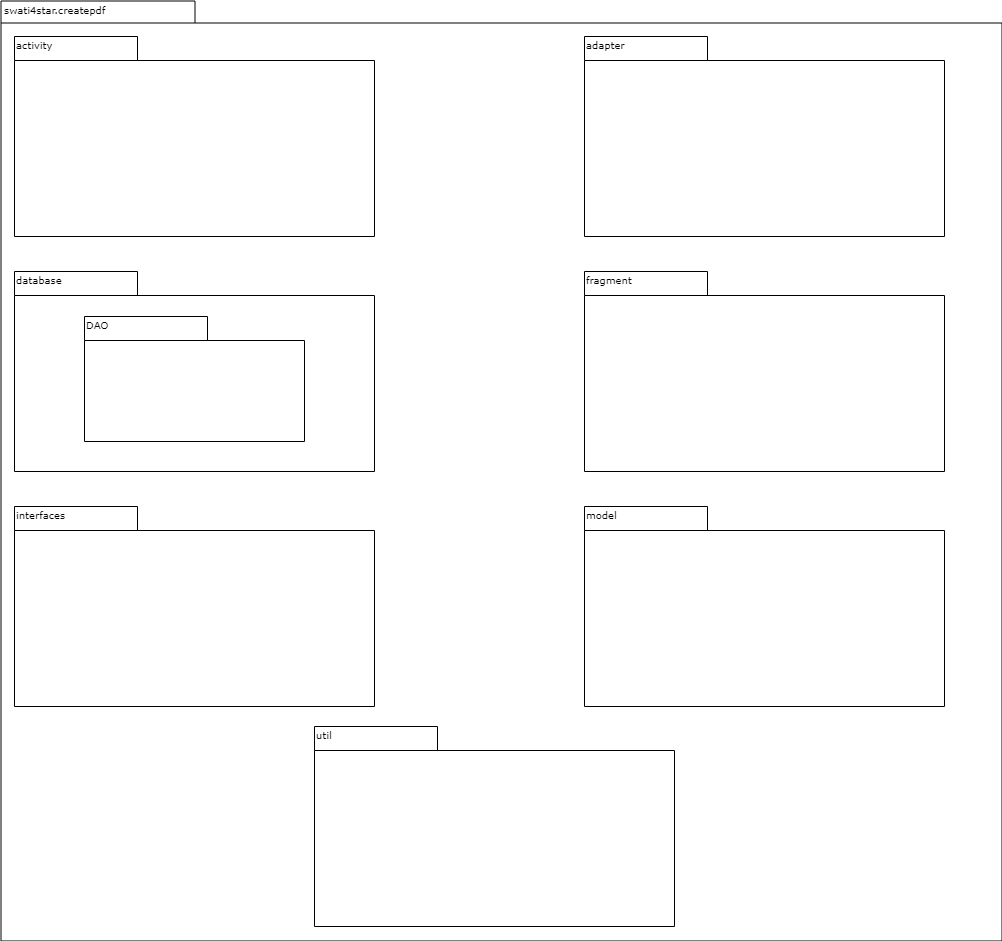
**1.4 Riferimenti**

- <https://github.com/Swati4star/Images-to-PDF> (link al progetto Open-Source)

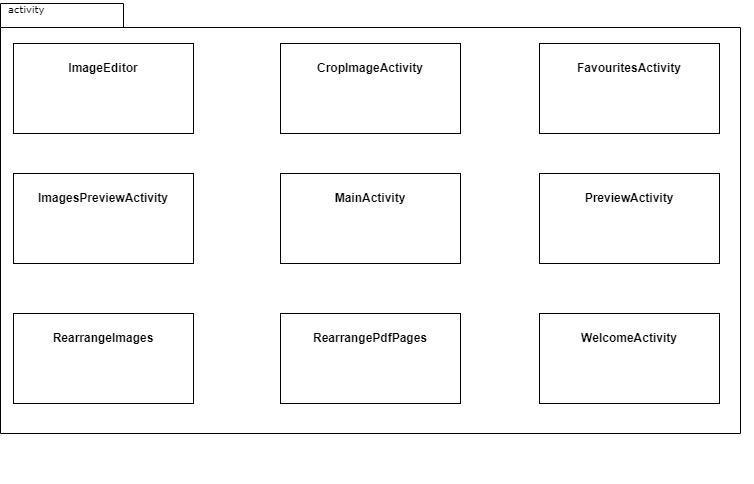
**2 Packages**

Come specificato nell’ SDD, in questa sezione viene mostrata la suddivisione del sistema in package:

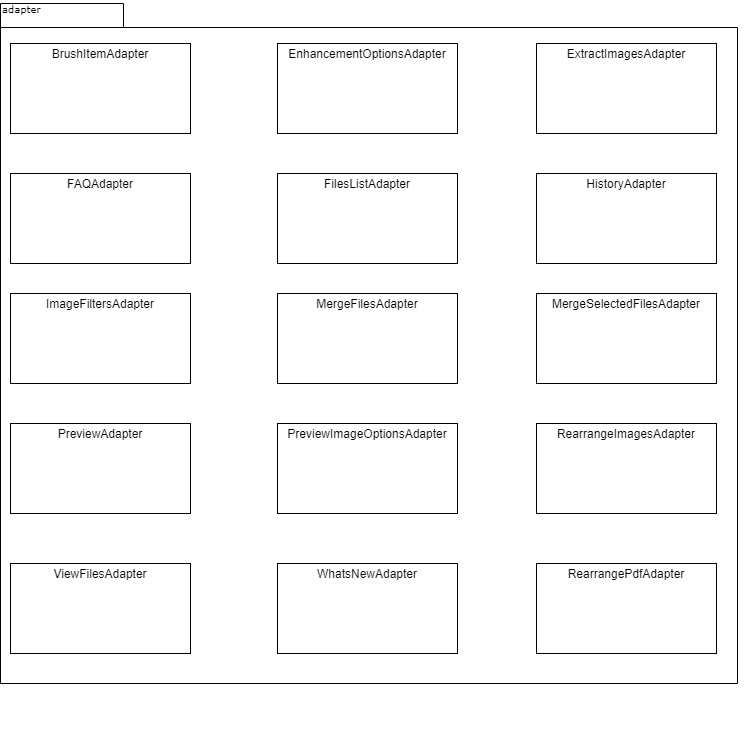
* **activity**: contiene le classi che forniscono un’interfaccia per l’interazione dell’utente con il sistema
* **adapter**: contiene le classi che fanno da ponte tra le view e il model, fornendo l’accesso a quest’ultimo
* **database**: contiene le classi che accedono e operano sul database, contiene inoltre i bean dei rispettivi DAO
* **fragment**: contiene le classi che rappresentano una porzione riutilizzabile dell’UI, definibili come sub-activity
* **interfaces**: contiene le interfacce Java e i listener
* **model**: contiene i bean del sistema
* **util**: contiene le classi che si occupano della logica applicativa, rendono operativa l’applicazione



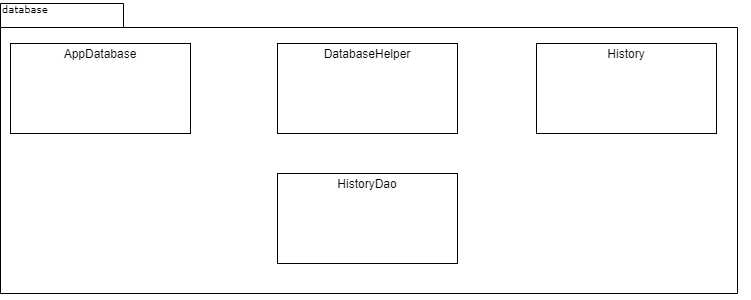
**Package activity**



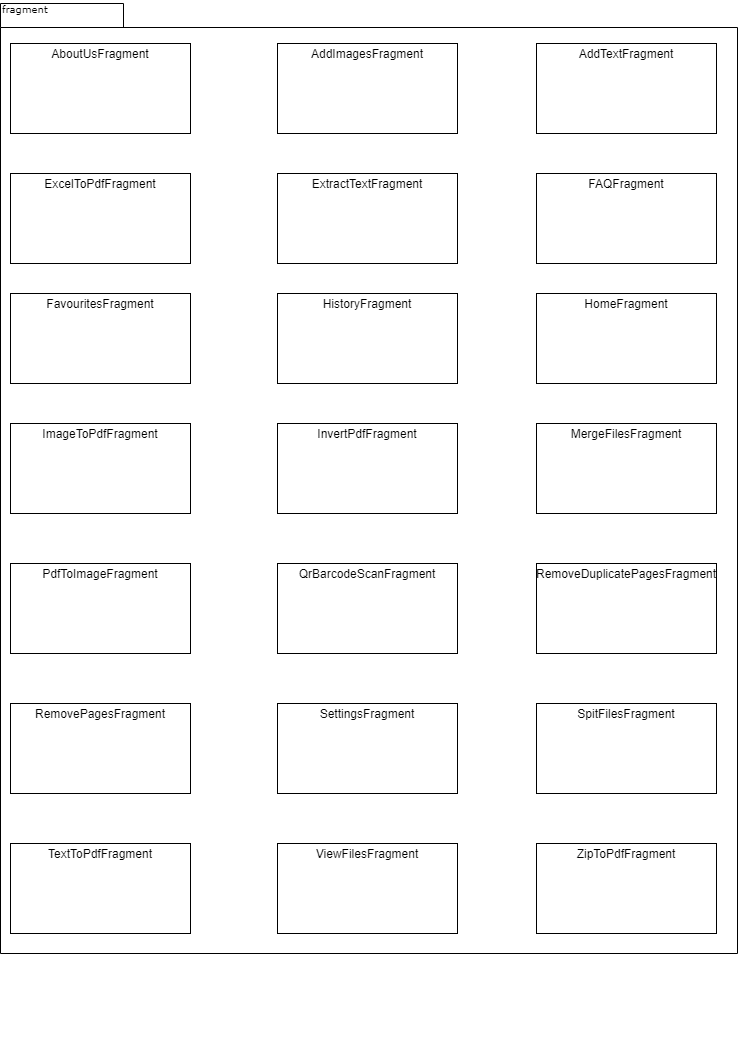
**Package adapter**



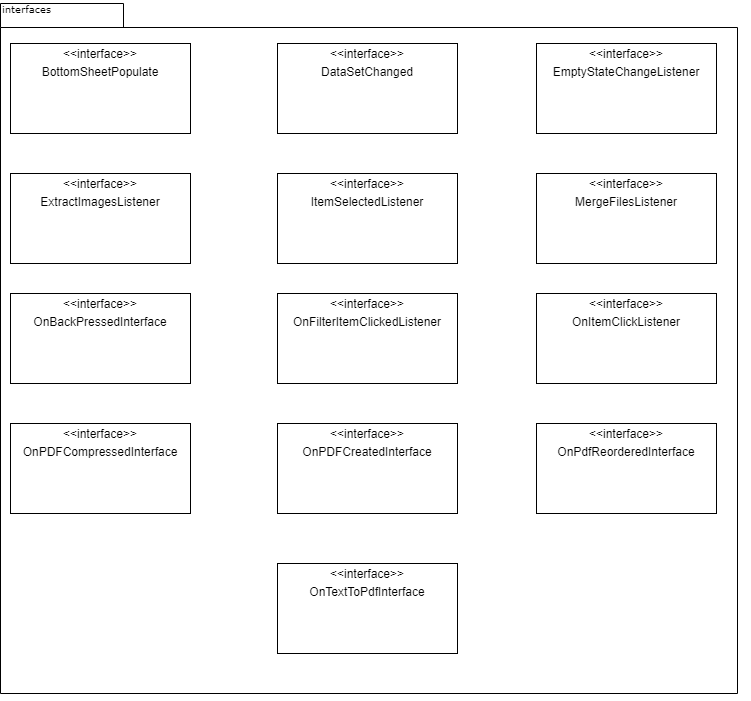
**Package database**



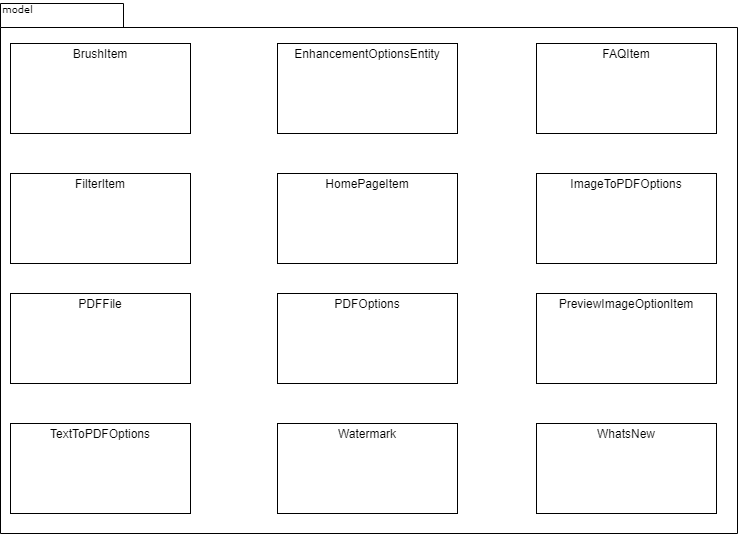
**Package fragment**



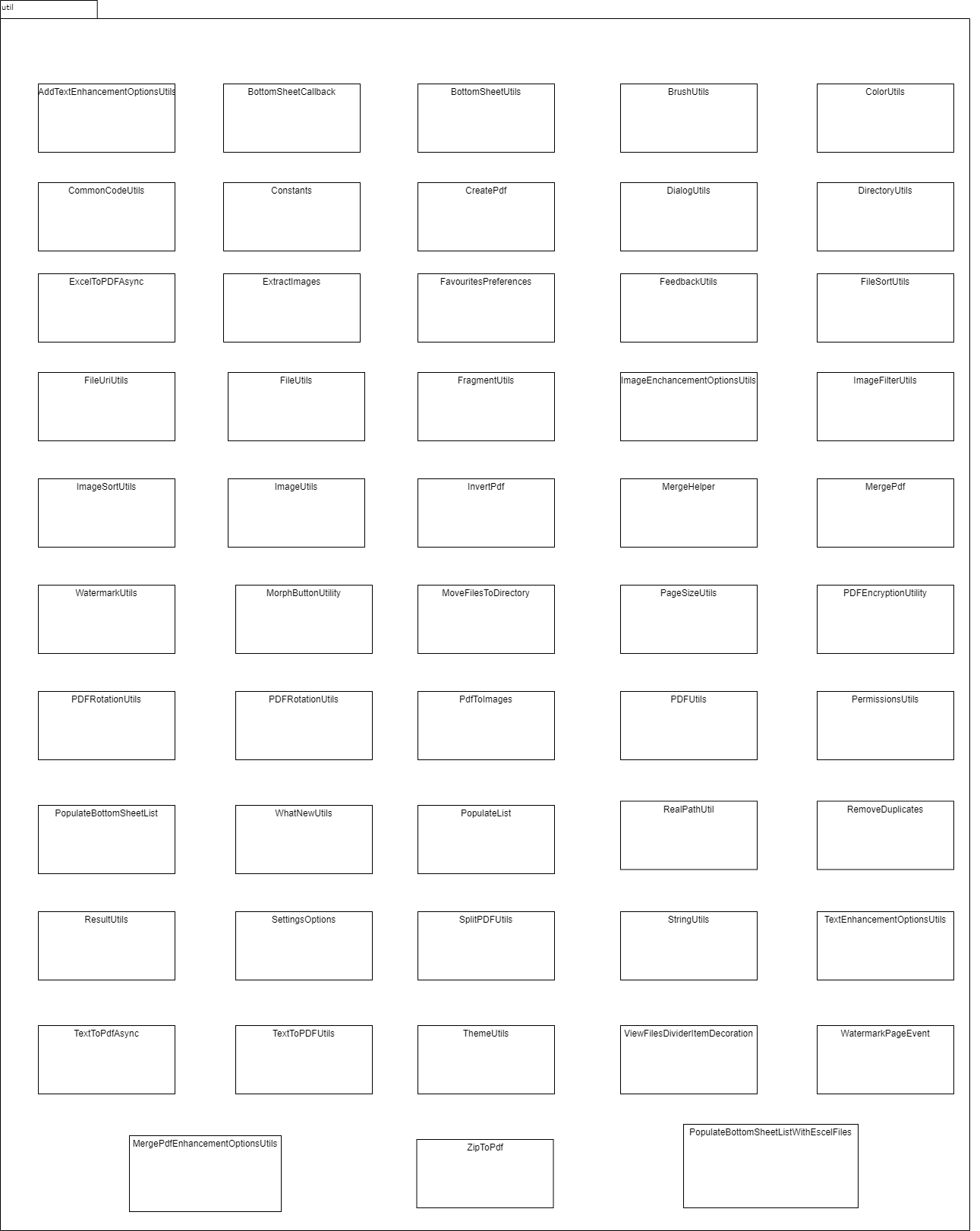
**Package interfaces**



**Package model**



**Package util**



**3 Class Interfaces**

|  |  |
| --- | --- |
| **Nome classe** | BottomSheetPopulate |
| **Descrizione** | La classe permette di gestire le operazioni che popolano le liste degli elementi in background |
| **Metodi** | +onPopulate(ArrayList<String>paths): void |
| **Invariante di classe** | \ |

|  |  |
| --- | --- |
| **Nome metodo** | +onPopulate(ArrayList<String>paths): void |
| **Descrizione** | Il metodo popola una lista di elementi in background |
| **Pre-condizione** |  |
| **Post-condizione** |  |

|  |  |
| --- | --- |
| **Nome classe** | DataSetChanged |
| **Descrizione** | La classe permette l’aggiornamento del dataset |
| **Metodi** | +updateDataset(): void |
| **Invariante di classe** | \ |

|  |  |
| --- | --- |
| **Nome metodo** | +updateDataset(): void |
| **Descrizione** | Il metodo aggiorna il dataset |
| **Pre-condizione** | If(position>=0 && position < mFileList.size()) |
| **Post-condizione** |  |

|  |  |
| --- | --- |
| **Nome classe** | EmptyStateChangeListener |
| **Descrizione** | Questa classe fa da listener per le classi che gestiscono la visibilità e i permessi dei file |
| **Metodi** | +setEmptyStateVisible(): void  +setEmptyStateInvisible(): void  +showNoPermissionsView(): void  +hideNoPermissionsView(): void  +filesPopulated(): void |
| **Invariante di classe** | \ |

|  |  |
| --- | --- |
| **Nome metodo** | +setEmptyStateVisible(): void |
| **Descrizione** | Il metodo imposta lo stato ‘’visibile’’ al file |
| **Pre-condizione** | If(mFileList.size()==0) |
| **Post-condizione** |  |

|  |  |
| --- | --- |
| **Nome metodo** | +setEmptyStateInvisible(): void |
| **Descrizione** | Il metodo imposta lo stato ‘’invisibile’’ al file |
| **Pre-condizione** | If(mFileList.size()==0) |
| **Post-condizione** |  |

|  |  |
| --- | --- |
| **Nome metodo** | +showNoPermissionsView(): void |
| **Descrizione** | Il metodo mostra la view ‘’NoPermissions’’ |
| **Pre-condizione** | If(files==null) |
| **Post-condizione** |  |

|  |  |
| --- | --- |
| **Nome metodo** | +hideNoPermissionsView(): void |
| **Descrizione** | Il metodo nasconde la view ‘’NoPermissions’’ |
| **Pre-condizione** | If(files!=null && files.size > 0) |
| **Post-condizione** |  |

|  |  |
| --- | --- |
| **Nome metodo** | +filesPopulated(): void |
| **Descrizione** | Il metodo notifica che l’array di file PDF è popolato |
| **Pre-condizione** | If(files!=null && files.size > 0) |
| **Post-condizione** |  |

|  |  |
| --- | --- |
| **Nome classe** | ExtractImagesListener |
| **Descrizione** | Questa classe listener gestisce le funzioni per le estrazioni delle immagini |
| **Metodi** | +resetView(): void  +extractionStarted(): void  +updateView(int imageCount, ArrayList<String> outputFilePaths): void |
| **Invariante di classe** | \ |

|  |  |
| --- | --- |
| **Nome metodo** | +resetView(): void |
| **Descrizione** | Il metodo resetta la view |
| **Pre-condizione** |  |
| **Post-condizione** |  |

|  |  |
| --- | --- |
| **Nome metodo** | +extractionStarted(): void |
| **Descrizione** | Il motodo notifica l’inizio dell’operazione di estrazione delle immagini |
| **Pre-condizione** |  |
| **Post-condizione** |  |

|  |  |
| --- | --- |
| **Nome metodo** | +updateView(int imageCount, ArrayList<String> outputFilePaths): void |
| **Descrizione** | Il metodo aggiorna la view dopo l’estrazione delle immagini |
| **Pre-condizione** |  |
| **Post-condizione** |  |

|  |  |
| --- | --- |
| **Nome classe** | ItemSelectedListener |
| **Descrizione** | Questa classe listener controlla se il file è selezionato |
| **Metodi** | +isSelected(Boolean isSelected, int countFiles): void |
| **Invariante di classe** | \ |

|  |  |
| --- | --- |
| **Nome metodo** | +isSelected(Boolean isSelected, int countFiles): void |
| **Descrizione** | Il metodo controlla se il file è selezionato |
| **Pre-condizione** |  |
| **Post-condizione** |  |

|  |  |
| --- | --- |
| **Nome classe** | MergeFilesListener |
| **Descrizione** | Questa classe listener gestisce le funzioni per il merge dei file PDF |
| **Metodi** | +resetValues(boolean isPDFMerged, String path): void  +mergeStarted(): void |
| **Invariante di classe** | \ |

|  |  |
| --- | --- |
| **Nome metodo** | +resetValues(boolean isPDFMerged, String path): void |
| **Descrizione** | Il metodo resetta i parametri del file dopo l’operazione di merge |
| **Pre-condizione** |  |
| **Post-condizione** |  |

|  |  |
| --- | --- |
| **Nome metodo** | +mergeStarted(): void |
| **Descrizione** | Il metodo notifica che l’operazione di merge è iniziata |
| **Pre-condizione** |  |
| **Post-condizione** |  |

|  |  |
| --- | --- |
| **Nome classe** | OnBackPressedInterface |
| **Descrizione** | La classe gestisce il comportamento dell’app alla pressione del tasto ‘’back’’ |
| **Metodi** | +closeBottomSheet(): void  +ckeckSheetBehaviour(): boolean |
| **Invariante di classe** | \ |

|  |  |
| --- | --- |
| **Nome metodo** | +closeBottomSheet(): void |
| **Descrizione** | Chiude l’elemento bottom sheet |
| **Pre-condizione** | If(bottomSheetBehaviour==true) |
| **Post-condizione** |  |

|  |  |
| --- | --- |
| **Nome metodo** | +ckeckSheetBehaviour(): boolean |
| **Descrizione** | Il metodo controlla il contenuto del fragment |
| **Pre-condizione** |  |
| **Post-condizione** |  |

|  |  |
| --- | --- |
| **Nome classe** | OnFilterItemClickedListener |
| **Descrizione** | La classe listener gestisce le funzioni ‘’on click’’ per la gestione della modifica delle immagini |
| **Metodi** | +onItemClick(View v, int position): void |
| **Invariante di classe** | \ |

|  |  |
| --- | --- |
| **Nome metodo** | +onItemClick(View v, int position): void |
| **Descrizione** | Il metodo imposta la view e la posizione dell’adapter al click |
| **Pre-condizione** |  |
| **Post-condizione** |  |

|  |  |
| --- | --- |
| **Nome classe** | OnItemClickListener |
| **Descrizione** | La classe listener gestisce le funzioni ‘’on click’’ per la modifica dei colori delle immagini |
| **Metodi** | +onItemClick(int position): void |
| **Invariante di classe** | \ |

|  |  |
| --- | --- |
| **Nome metodo** | +onItemClick(int position): void |
| **Descrizione** | Il metodo imposta la posizione dell’adapter al click |
| **Pre-condizione** |  |
| **Post-condizione** |  |

|  |  |
| --- | --- |
| **Nome classe** | OnPDFCompressedInterface |
| **Descrizione** | La classe gestisce le funzionalità di notifica all’utente durante la compressione di un file PDF |
| **Metodi** | +pdfCompressionStarted(): void  +pdfCompressionEnded(String path, Boolean success): void |
| **Invariante di classe** | \ |

|  |  |
| --- | --- |
| **Nome metodo** | +pdfCompressionStarted(): void |
| **Descrizione** | Il metodo notifica all’utente l’inizio dell’operazione di compressione |
| **Pre-condizione** |  |
| **Post-condizione** |  |

|  |  |
| --- | --- |
| **Nome metodo** | +pdfCompressionEnded(String path, Boolean success): void |
| **Descrizione** | Il metodo notifica all’utente la fine dell’operazione di compressione |
| **Pre-condizione** |  |
| **Post-condizione** |  |

|  |  |
| --- | --- |
| **Nome classe** | OnPDFCreatedInterface |
| **Descrizione** | La classe gestisce le funzionalità di notifica all’utente durante la creazione di un PDF |
| **Metodi** | +onPDFCreationStarted(): void  +onPDFCreated(boolean success, String path): void |
| **Invariante di classe** | \ |

|  |  |
| --- | --- |
| **Nome metodo** | +onPDFCreationStarted(): void |
| **Descrizione** | Il metodo notifica all’utente l’inizio dell’operazione di creazione di un file PDF |
| **Pre-condizione** |  |
| **Post-condizione** |  |

|  |  |
| --- | --- |
| **Nome metodo** | +onPDFCreated(boolean success, String path): void |
| **Descrizione** | Il metodo notifica all’utente il termine dell’operazione di creazione di un file PDF |
| **Pre-condizione** |  |
| **Post-condizione** |  |

|  |  |
| --- | --- |
| **Nome classe** | OnPdfReorderedInterface |
| **Descrizione** | La classe gestisce le funzionalità di notifica all’utente durante il riordinamento delle pagine di un PDF |
| **Metodi** | +onPdfReorderStarted(): void  +onPdfReorderCompleted(List<Bitmap> bitmaps): void  +onPdfReorderFailed(): void |
| **Invariante di classe** | \ |

|  |  |
| --- | --- |
| **Nome metodo** | +onPdfReorderStarted(): void |
| **Descrizione** | Il metodo notifica all’utente l’inizio dell’operazione di riordinamento pagine di un PDF |
| **Pre-condizione** |  |
| **Post-condizione** |  |

|  |  |
| --- | --- |
| **Nome metodo** | +onPdfReorderCompleted(List<Bitmap> bitmaps): void |
| **Descrizione** | Il metodo notifica all’utente il termine dell’operazione di riordinamento pagine di un PDF |
| **Pre-condizione** |  |
| **Post-condizione** |  |

|  |  |
| --- | --- |
| **Nome metodo** | +onPdfReorderFailed(): void |
| **Descrizione** | Il metodo notifica all’utente il fallimento dell’operazione di riordinamento pagine di un PDF |
| **Pre-condizione** |  |
| **Post-condizione** |  |

|  |  |
| --- | --- |
| **Nome classe** | OnTextToPdfInterface |
| **Descrizione** | La classe gestisce le funzionalità di notifica all’utente dell’operazione di conversione da file di testo a PDF |
| **Metodi** | +onPDFCreationStarted(): void  +onPDFCreated(boolean success): void |
| **Invariante di classe** | \ |

|  |  |
| --- | --- |
| **Nome metodo** | +onPDFCreationStarted(): void |
| **Descrizione** | Il metodo notifica all’utente l’inizio dell’operazione di conversione file di testo in file PDF |
| **Pre-condizione** |  |
| **Post-condizione** |  |

|  |  |
| --- | --- |
| **Nome metodo** | +onPDFCreated(boolean success): void |
| **Descrizione** | Il metodo notifica all’utente la fine dell’operazione di conversione file di testo in file PDF |
| **Pre-condizione** |  |
| **Post-condizione** |  |

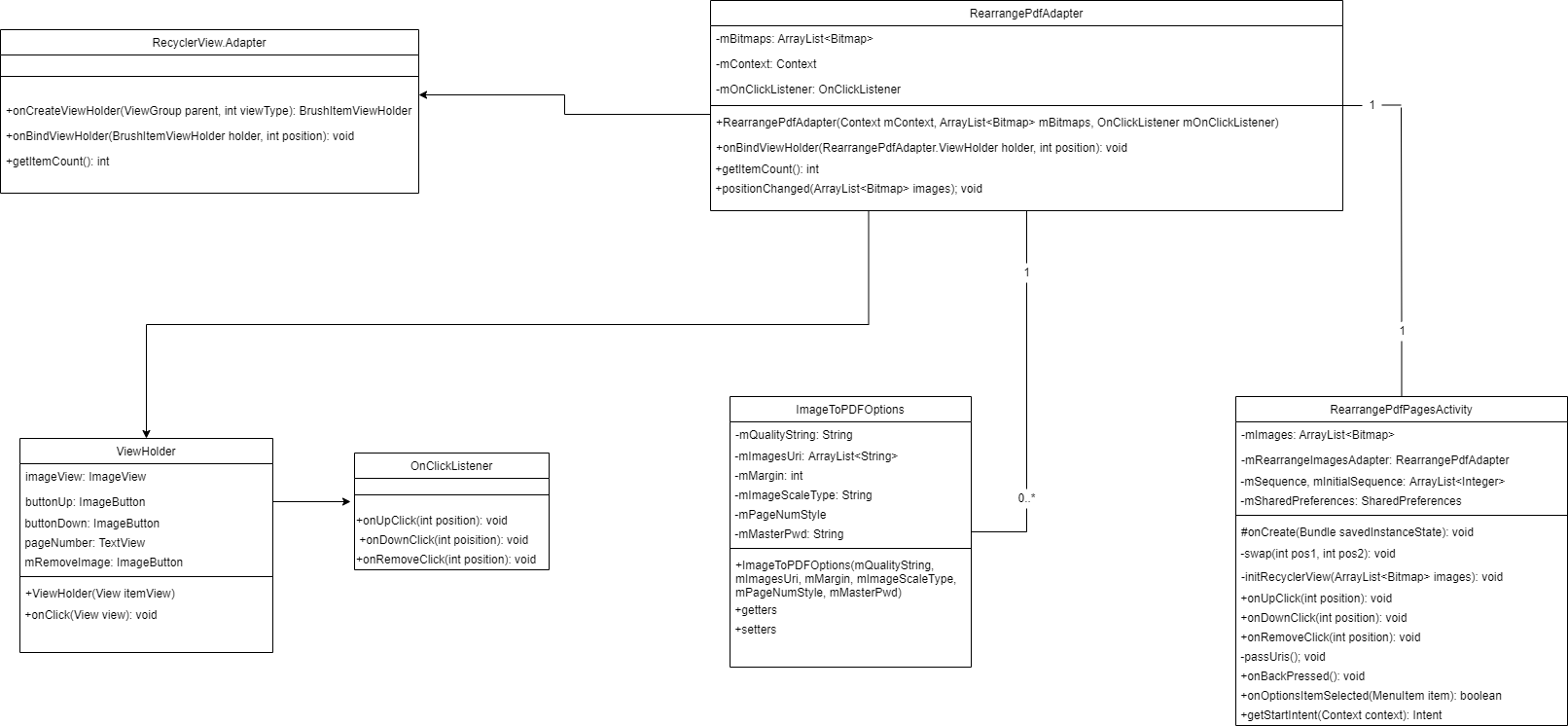
**4 Design Patterns**

Nella presente sezione si andranno a descrivere e illustrare i design pattern utilizzati nello sviluppo dell’applicazione Images-to-PDF. Per ogni pattern si darà una breve introduzione teorica, il problema da esso risolto, e verrà illustrato un grafico della struttura delle classi che implementano il pattern.

**Adapter Pattern**

L’adapter è un design pattern strutturale che permette di far lavorare due classi incompatibili convertendo la classe della interfaccia in quella che il cliente si aspetta. Si implementa attraverso delle classi denominate “adapter” che convertono i dati in oggetti leggibili dal sistema.

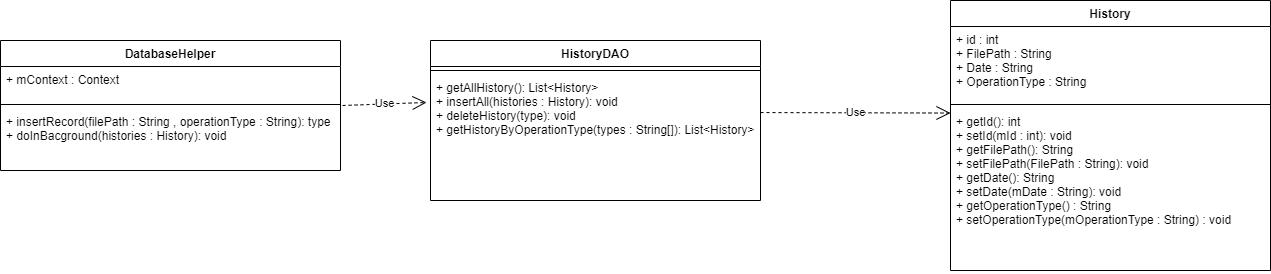
Nel caso di un applicativo android, la vista dell’interfaccia è implementata con la viewgroup (classe che si occupa del layout della view ) chiamata “ RecyclerView ”. Ogni classe adapter quindi connette i relativi dati alla nostra RecyclerView e li rappresenta tramite ViewHolder, componente responsabile di collezionare e mostrare i dati. L’ adapter di per sé si occupa solo di interfacciare la view con i dati, il viewholder è il metadato vero e proprio da inserire nella RecyclerView.



**DAO**

Un Dao pattern consiste nell’avere un’interfaccia astratta per collegarci ed effettuare alcune operazioni sui database in maniera diretta e senza esporre dettagli ulteriori su di esso.

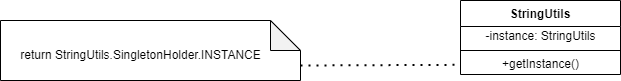
Images-To-PDF gestisce una entità del database che rappresenta la cronologia delle operazioni effettuate dall’utente su una base di dati persistente. È stato quindi implementato un DAO per gestire le operazioni su questo dato.



**Singleton**

I design patterns creazionali consentono di fornire un livello di astrazione relativamente alla creazione di oggetti (processo di istanziazione) aiutando lo sviluppatore a rendere il sistema indipendente dalle modalità di creazione, composizione e rappresentazione degli oggetti utilizzati. Garantisce che venga strutturata una sola istanza di una classe e che venga fornito un punto di accesso globale dalla classe stessa.

Images-To-PDF utilizza principalmente questo pattern per gli oggetti che modificano e agiscono sui documenti selezionati nell’applicazione, evitando che possano crearsi più istanze con diversi parametri che potrebbero creare dei conflitti e deviare il comportamento delle componenti confondendo i risultati.



**5 Glossario**

**Package:** Raggruppamento di classi e interfacce

**Images-to-PDF:** Nome del progetto software

**Merge:** Operazione di unione di molteplici file PDF