



## UNIVERSITA' DEGLI STUDI DELLA BASILICATA

DIPARTIMENTO DI MATEMATICA, INFORMATICA ED ECONOMIA

Corso di Laurea in Scienze e Tecnologie Informatiche

# Progettazione e Sviluppo di una app mobile per la document security

Candidato:

**Mario Rosa**

Matricola 54104

Relatore:

**Ing. Domenico Daniele Bloisi**



*Alla mia famiglia*



# Ringraziamenti



# Indice

<b>1</b>	<b>Introduzione</b>	<b>3</b>
1.1	Motivazioni . . . . .	4
1.2	Obiettivi della tesi . . . . .	5
1.3	Contributi personali . . . . .	5
1.4	Struttura della tesi . . . . .	6
<b>2</b>	<b>Document security</b>	<b>7</b>
2.1	Document Management System . . . . .	7
2.2	Requisiti per la sicurezza . . . . .	8
2.3	Il formato PDF . . . . .	10
<b>3</b>	<b>Stato d'arte</b>	<b>11</b>
3.1	Strumenti per la sicurezza . . . . .	11
3.2	Software per la gestione . . . . .	13
<b>4</b>	<b>Sviluppo e progettazione</b>	<b>15</b>
4.1	L'idea . . . . .	15
4.2	Analisi dei requisiti . . . . .	15
<b>5</b>	<b>Implementazione</b>	<b>21</b>
5.1	Applicazione lato server . . . . .	22
5.2	Persistenza dei dati . . . . .	24
5.3	Applicazione lato client . . . . .	27
5.4	Riservatezza . . . . .	35
5.5	Controllo e permessi . . . . .	41
5.6	Protezione e sicurezza dei documenti . . . . .	42
<b>6</b>	<b>Risultati sperimentali</b>	<b>45</b>
6.1	Set-up sperimentale . . . . .	45
6.2	Test della protezione dei documenti . . . . .	46

<b>7 Conclusioni</b>	<b>51</b>
7.1 Sviluppi futuri . . . . .	52
<b>Bibliografia</b>	<b>53</b>



# Capitolo 1

## Introduzione

La *document security* ovvero la Sicurezza dei documenti è un tema di fondamentale importanza nell'era digitale in cui viviamo. Quando si parla di sicurezza dei documenti si fa riferimento a tutte le procedure e protocolli che si adottano per garantire la protezione di un documento digitale per tutto le fasi del suo ciclo di vita

Il Codice dell'Amministrazione Digitale (CAD-DLgs 82/2005) definisce il documento informatico come "*rappresentazione informatica di atti, fatti o dati giuridicamente rilevanti*".

Il Regolamento eIDAS n. 910/2014 definisce il documento elettronico come "*qualsiasi contenuto conservato in forma elettronica, in particolare testo o registrazione sonora, visiva o audiovisiva*".

Con sempre più documenti che vengono digitalizzati, la sicurezza dei documenti sta diventando estremamente importante perché la crescita esponenziale di dati digitali ha portato anche un aumento della vulnerabilità delle informazioni sensibili e delle violazioni della privacy. Le violazioni della sicurezza possono causare la fuga di informazioni sensibili, l'accesso a documenti da parte di persone non autorizzate e perdita o sottrazioni di dati. Sempre più organizzazioni, piccole e grandi, vanno verso la digitalizzazione i processi aziendali, spesso automatizzati e basati principalmente sulla gestione di documenti digitali che contengono informazioni critiche, personali e sensibili ed è essenziale proteggere la riservatezza e la privacy delle informazioni utilizzate durante questi processi. Per un'azienda è di fondamentale importanza dotarsi di un sistema che assicuri massima protezione ai documenti, ma soprattutto che risulti essere conforme e in linea con la normativa europea GDPR. La sicurezza dei documenti non è un facoltativo. Ogni azienda è tenuta a dimostrare la sicurezza dei propri processi documentali per ottenere la conformità GDPR.

Il **Regolamento generale sulla protezione dei dati** (GDPR) è un regolamento

europeo entrato in vigore 25 maggio 2018, che disciplina il modo in cui le aziende e le altre organizzazioni trattano i dati personali. È il provvedimento più significativo degli ultimi 20 anni in materia di protezione dei dati e ha implicazioni importanti per qualsiasi organizzazione al mondo che si rivolga ai cittadini dell’Unione Europea. Il GDPR ha introdotto nuove regole per la raccolta, l’elaborazione e la conservazione dei dati personali dei cittadini dell’Unione Europea. Le organizzazioni che non rispettano queste regole possono essere soggette a sanzioni significative. Il regolamento si applica a tutte le organizzazioni che trattano i dati personali dei cittadini dell’Unione Europea, indipendentemente dal fatto che l’organizzazione sia situata all’interno o all’esterno dell’UE. Le sanzioni previste dal GDPR possono essere molto elevate. In caso di violazione del regolamento, le organizzazioni possono essere soggette a multe fino al 4% del loro fatturato annuo globale o 20 milioni di euro (a seconda di quale cifra sia maggiore). Tuttavia, le sanzioni dipendono dalla gravità della violazione e dal fatto che l’organizzazione abbia preso misure per prevenire la violazione. I sistemi di gestione per la sicurezza dei documenti devono rispettare i requisiti normativi per la privacy e la sicurezza.

## 1.1 Motivazioni

Negli ultimi anni la pandemia di COVID-19 ha accelerato significativamente il processo di digitalizzazione in molti settori. A causa delle restrizioni imposte dal distanziamento sociale e delle limitazioni agli spostamenti, molte organizzazioni hanno adottato soluzioni digitali per mantenere la continuità delle attività. Numerose organizzazioni, come aziende, imprese, istituzioni governative, strutture sanitarie, finanziarie, bancarie, amministrazioni pubbliche e hanno registrato un aumento dell’adozione di processi documentali digitali per ridurre la dipendenza dai documenti cartacei e facilitare il lavoro da remoto. Questo ha portato anche ad un forte aumento delle minacce e attacchi informatici, ma anche alla perdita accidentale dei documenti digitali causate da errori umani. Per queste ragioni, è essenziale adottare misure adeguate per la sicurezza dei documenti e garantire la protezione dei dati sensibili e confidenziali che devono essere protetti da accessi non autorizzati, furti di informazioni, manipolazioni o divulgazioni indebite. La sicurezza dei documenti è un tema che non deve essere sottovalutato, dovrebbe essere trattata come una priorità sia a livello individuale che organizzativo per evitare potenziali rischi. È importante educare le persone sull’importanza di questo tema e fornire formazione sulle buone pratiche, e portare alla consapevolezza che la mancanza di adeguate misure di sicurezza può portare a gravi conseguenze.

## 1.2 Obiettivi della tesi

In questa tesi, si esplorerà il tema della sicurezza dei documenti e si analizzeranno le principali tecniche e tecnologie utilizzate per garantire la protezione dei dati sensibili. Si tratterà di un'analisi delle principali minacce alla sicurezza dei documenti, nonché delle strategie di sicurezza che possono essere implementate per prevenire tali minacce. In particolare, si esamineranno le tecnologie di cifratura dei dati, le politiche di gestione delle chiavi di crittografia, i protocolli di autenticazione e di autorizzazione degli utenti, nonché le tecnologie di controllo degli accessi e di gestione dei diritti di accesso. Il lavoro di tesi si concentrerà anche sull'applicazione pratica di queste tecnologie, attraverso la progettazione e lo sviluppo di un sistema di sicurezza che nel complesso sia capace garantire la protezione e la sicurezza dei documenti digitali.

## 1.3 Contributi personali

Il mio contributo personale è stato quello della progettazione e lo sviluppo di un'app mobile client-server per la gestione di documenti PDF. All'inizio è stato importante approfondire il tema della sicurezza dei documenti e per acquisire conoscenze e competenze tecniche fondamentali per la progettazione e lo sviluppo di soluzioni di sicurezza dei documenti altamente efficaci e affidabili, per poi passare alla progettazione dell'applicazione. La progettazione è iniziata con l'analisi dei requisiti per capire quali funzionalità doveva fornire l'applicazione e quali dati dovrà gestire. Una volta completata la fase di progettazione, sono passato alla fase di sviluppo. In questa fase è stato scritto il codice dell'applicazione e viene definita l'architettura dell'applicazione, del database e la logica di business e la scelta delle tecnologie da utilizzare.

## 1.4 Struttura della tesi

Concluso il Capitolo 1, relativo all'introduzione, a seguire La tesi è suddivisa in sei capitoli:

- Nel Capitolo 2 vengono descritti i requisiti implementati nei sistemi di gestione dei documenti per garantire la Document security e i vantaggi che offrono tali sistemi. In fine una descrizione del formato PDF dei documenti.
- Capitolo 3: Stato d'arte
- Nel Capitolo 4 si passa alla sviluppo e la progettazione di un' applicazione per la gestione e la sicurezza di documenti PDF, in particolare si affronta la fase di raccolta e analisi dei requisiti e realizzazione dei diagrammi UML.
- Il Capitolo 5 è dedicato alla descrizione delle tecnologie utilizzate nella fase di implementazione del sistema.
- Nel Capitolo 6 vengono riportare quelle che sono le principali criticità dell'applicazione e del sistema che potrebbero mette a rischio la sicurezza dei documenti.
- Il Capitolo 7 è il capitolo conclusivo, vengono considerati sia quelli che sono i principali punti di forza di questo progetto sia i possibili sviluppi futuri per migliorare il sistema.

# Capitolo 2

## Document security

In questo capitolo saranno descritte le misure di sicurezza da implementare nei sistemi di gestione dei documenti digitali, che sono necessarie per garantire la sicurezza e protezione dei documenti.

### 2.1 Document Management System

Un buon *sistema di gestione dei documenti* è un software che fornisce un modo automatizzato per archiviare, acquisire, distribuire, gestire e tenere traccia di documenti elettronici, facilitandone l'accesso, la modifica e la condivisione e allo stesso tempo fornire un elevato grado di protezione e controllo, al fine di evitare furti di informazioni proprietarie. In generale, un *sistemi di gestione dei documenti* comprende diversi componenti capaci di rispettare i *requisiti di sicurezza* che vederemo in seguito nel dettaglio, per garantire una completa protezione dei documenti. Quindi alcuni di questi aiutano a limitare l'accesso solo agli utenti autorizzati, mentre altri componenti controllano le autorizzazioni o implementano sistemi utili per la crittografia dei documenti. Non tutti i sistemi di gestione dei documenti sono uguali ma variano in base al tipo di attività e delle esigenze, in generale si posso distinguere due macro categorie:

- *Sistema On-Premise*: aziende o organizzazioni acquistano il software di gestione che è ospitato sui server del fornitore e accedono al sistema tramite internet. Il fornitore è responsabile di tutta la manutenzione, gli aggiornamenti e la sicurezza.
- *Sistema Off-Premise*: aziende o organizzazioni acquistano il software di gestione lo installano dietro i propri server locali. L'azienda è responsabile della sicurezza e della gestione del sistema.

Altri vantaggi che dovremmo aspettarci dall'implementazione di un sistema di gestione dei documenti adeguatamente configurato:

- Efficienza migliorata: i flussi di lavoro, l'integrazione e i processi automatizzati dovrebbero rendere i team più efficienti
- Errori ridotti: i dati e le autorizzazioni convalidati dovrebbero impedire a molti tipi di errori umani di entrare e propagarsi all'interno del sistema. Riduzione dei costi della carta: i costi della carta aumentano nel tempo. La maggior parte viene stampata dozzine di volte solo per essere gettata nella spazzatura entro un giorno.

## 2.2 Requisiti per la sicurezza

Molte soluzioni per la sicurezza delle informazioni tentano di proteggere i documenti elettronici solo nella loro posizione di archiviazione o durante la trasmissione di essi. Immaginiamo infatti uno scenario dove un volta che il documento raggiunge il destinatario, il documento può essere intenzionalmente o meno inoltrato e visualizzato da destinatari non autorizzati. Con questo approccio la sicurezza rimane un problema poiché non si fornisce protezione effettiva per il contenuto di un documento elettronico per tutto il suo ciclo di vita. Quando si parla di *ciclo di vita di un documento* si fa riferimento a tutte le diverse fasi che il documento attraversa da quando venne creato fino a quando cessa di avere utilità. In ogni fase del ciclo di vita un documento, che sia la fase di trasmissione o archiviazione si affronta il pericolo che il documento venga rubato, perso o compromesso.

In generale, per garantire protezione e sicurezza è necessario affidarsi ad un sistema di gestione dei documenti che sia capace di rispettare i seguenti requisiti :

**Autenticazione:** l'autenticazione è un processo attraverso il quale viene verificata l'identità di un utente e obbliga coloro che tentano di accedere ai documenti a dimostrare di essere chi dicono di essere. Ciò richiede una solida gestione delle identità.

**Riservatezza:** riservatezza significa che le informazioni nel file rimangono private. La segretezza richiesta per proteggere il contenuto del file da coloro che non sono autorizzati a visualizzarlo viene applicata con la crittografia. La crittografia utilizza chiavi e algoritmi crittografici per codificare o crittografare il contenuto di un file in modo che diventi illeggibile. Pertanto, solo gli utenti e i destinatari validi, che possiedono la chiave crittografica corretta, possono de-crittografare e visualizzare il contenuto del file.

**Autorizzazione:** adeguate misure di sicurezza assicurano che i documenti siano disponibili agli utenti autorizzati. Oltre a specificare chi ha accesso, l'autorizzazio-

ne determina ciò che un utente può fare con un documento ad esempio: stampare o copiare il contenuto, compilare campi, aggiungere commenti o annotare il documento, inserire o rimuovere pagine, inoltrare il documento, accedere al documento offline ecc. Utilizzando il controllo dinamico dei documenti, le organizzazioni possono gestire e monitorare l'utilizzo dei documenti elettronici. Il controllo dinamico dei documenti determina i diritti di accesso e i permessi assegnati a un documento elettronico una volta che è stato distribuito e include le seguenti funzionalità:

- *Scadenza e revoca del documento*: ovvero revocare l'accesso a un documento mediante l'applicazione di date di scadenza limitandone l'accesso. Ad esempio, un autore può inviare un documento che scadrà tra due settimane in modo che i destinatari non possano accedervi una volta superata la data di scadenza. Oppure, l'accesso a un documento può essere revocato automaticamente se un destinatario autorizzato lascia il progetto o cambia reparto.
- *Gestione dell'accesso offline*: le organizzazioni possono gestire per quanto tempo un destinatario autorizzato può accedere a un documento offline. Una volta trascorso il periodo di tempo specificato, il destinatario non può più visualizzare il documento e deve tornare online per ottenere un ulteriore accesso.
- *Protezione con password*: dal punto di vista dell'accesso utente, l'abilitazione della protezione con password è il primo passo nella protezione dei documenti. È la prima barriera di sicurezza per impedire l'accesso non autorizzato ai file. Inoltre, è relativamente semplice da implementare, sebbene non del tutto infallibile.

**Integrità**: l'integrità garantisce che un file non sia stato alterato inavvertitamente o intenzionalmente, sia in stato di archiviazione che durante la trasmissione.

**Non ripudio**: garantisce che le parti coinvolte in una transazione non possano negare la loro partecipazione. Pertanto, un sistema di sicurezza dovrebbe essere in grado di dimostrare che qualcuno ha inviato, visualizzato o modificato un file. Il non ripudio si ottiene attraverso firme digitali impedisce al firmatario del documento di negare di averlo firmato. Le firme digitali forniscono l'autenticità del documento, cioè assicurano che il documento provenga dalla persona reale, verificando l'identità digitale del firmatario.

## 2.3 Il formato PDF

Il **PDF** (Portable Document Format) è un formato di file sviluppato da Adobe negli anni novanta e concepito per visualizzare i documenti in un formato elettronico, indipendente dal tipo di software, hardware o sistema operativo. In pratica un file in formato PDF viene visualizzato sempre nel solito modo, indipendentemente dai programmi che utilizziamo per visualizzarlo. Il successo del formato di file PDF è un ottimo esempio di come un prodotto diventa uno standard tecnico globale.

Alcuni motivi per cui il formato PDF è diventato così popolare:

- Il lettore PDF è stato reso gratuito da Adobe
- La formattazione rimane coerente sullo schermo e sulla stampa indipendentemente dal dispositivo, dal sistema operativo o dal software utilizzato. Non devi preoccuparti che il destinatario non sia in grado di visualizzare il documento o che veda una versione imperfetta.
- È possibile proteggere con password i file PDF e limitare la stampa, la modifica e la copia. Questo aiuta a proteggere le informazioni sensibili. È inoltre possibile aggiungere filigrane a documenti importanti per garantire l'integrità dei contenuti.
- La dimensione del file è inferiore rispetto ad altri formati, il che semplifica la condivisione e il download e consente di risparmiare spazio sul disco rigido.
- Puoi crearlo facilmente utilizzando stampanti e scanner, una funzionalità integrata in ogni computer. Se i tuoi documenti sono in un altro formato, la conversione in PDF è facile e richiede solo pochi clic.

L'utilizzo di un sistema per la document security consente quindi di controllare un documento in modo semplice e continuativo, anche dopo averlo distribuito. È possibile monitorarlo o impedire ad alcuni utenti di accedere al documento stesso, per avere la situazione sempre sotto controllo. Nei prossimi capitoli seguirà lo sviluppo e l'implementazione della mia soluzione proposta di un sistema di gestione per la document security.

# Capitolo 3

## Stato d'arte

### 3.1 Strumenti per la sicurezza

Nel capitolo precedente abbiamo descritto le principali proprietà che hanno reso famoso il formato PDF, tra cui la possibilità di proteggere i documenti attraverso l'uso di una password. L'aggiunta di una password a un file PDF consente di limitare l'accesso al documento solo alle persone autorizzate ed è importante per garantire la riservatezza delle informazioni particolarmente utile quando si condividono documenti. Vediamo quindi alcuni strumenti che consentono di sfruttare questa proprietà.

**Adobe Acrobat Reader**<sup>1</sup> è un software gratuito sviluppato da Adobe disponibile per dispositivi Windows, Mac OS e Android che consente di visualizzare, stampare e annotare documenti in formato PDF. La versione a pagamento del software offre funzionalità avanzate di modifica, creazione e soprattutto consente di impostare una **password** su un documento. Una volta impostata la password, il documento PDF sarà protetto dagli accessi non autorizzati e tutti coloro che non conoscono la password. Adobe ha reso disponibile gratuitamente online, senza dover scaricare e installare nulla, accessibile direttamente dalla finestra di browser qualsiasi, lo strumento Acrobat per aggiungere la protezione tramite password a un file PDF<sup>2</sup>. Lo svantaggio è che per utilizzare questo strumento online, Adobe chiede all'utente di registrarsi (gratuitamente), operazione che spesso può risultare fastidiosa.

---

<sup>1</sup><https://www.adobe.com/it/acrobat/pdf-reader.html>

<sup>2</sup><https://www.adobe.com/acrobat/online/password-protect-pdf.html>

Mentre alcuni esempi di tools gratuiti disponibili online, senza vincoli di registrazione, che consentono di impostare la password ad un documento PDF direttamente da browser sono:

- **SmallPDF**<sup>3</sup>, **PDF2Go**<sup>4</sup> e **Sejda**<sup>5</sup>.

Questi tre strumenti, molto simili tra loro, hanno la caratteristica di avere un'interfaccia semplice e consentono semplicemente di caricare il documento dal dispositivo o fonti cloud come Google Drive, Dropbox o da un URL e impostare la password. Una volta applicata la password il documento viene scaricato sul dispositivo. Alcuni strumenti online gratuiti potrebbero avere limitazioni sulle funzionalità disponibili. Ad esempio, potrebbero limitare il numero di file che puoi proteggere o la dimensione massima del file. Se si hanno esigenze complesse o file di grandi dimensioni, spesso è utile pensare ad alternative più avanzate, come software desktop dedicati.

Un altro svantaggio, quando si utilizzano servizi online è che potresti dover caricare i tuoi file sui loro server. Ciò significa avere meno controllo sulla sicurezza dei propri dati.

Un'ottima alternativa software gratuita è **PDFEncrypt**<sup>6</sup>.

PDFEncrypt è un'applicazione desktop open source per Windows, scritta per proteggere con password i file PDF in modo rapido e semplice.



Figura 3.1: Schermata PDFEncrypt

<sup>3</sup><https://smallpdf.com/split-pdf>

<sup>4</sup><https://www.pdf2go.com/it/proteggere-pdf>

<sup>5</sup><https://www.sejda.com/it/encrypt-pdf>

<sup>6</sup><https://pdfencrypt.net/>

Si presenta con una schermata molto semplice ed intuitiva: ci permette di selezionare il file da proteggere, la cartella dove verrà salvato una copia del file protetto e mostra il campo di testo dove inserire la password da applicare. In particolare, la password, può essere scelta dall’utente o in alternativa consigliata e quindi generata dal sistema(scelta più sicura). Il vantaggio principale è che, una volta installato localmente e avviata, l’applicazione funziona direttamente sul proprio PC senza la necessità di caricare i propri file PDF riservati su un server di terze parti e così mantenere il pieno controllo sui propri dati.

## 3.2 Software per la gestione

### OpenDocMan

OpenDocMan è un sistema di gestione dei documenti open source scritto in PHP, sviluppato sotto licenza GPL che ti permette di utilizzare il programma gratuitamente e di modificarlo come preferisci.



# Capitolo 4

## Sviluppo e progettazione

### 4.1 L'idea

La soluzione da me proposta si chiama SafeDoc Viewer.

L'idea di questo progetto è quella di realizzare un sistema che va a simulare i principali aspetti le caratteristiche di un sistema per gestione dei documenti *on-premise*, che dovrà consentire agli utenti di scaricare una serie di documenti esclusivamente in formato PDF e successivamente visualizzarli all'interno dell'applicazione. Il sistema permetterà unicamente a utenti autorizzati di accedere ed usufruire del servizio, quindi gli utenti dovranno essere in possesso delle credenziali. Dopo l'accesso, l'applicazione mostrerà una lista di serie di documenti disponibili per lo specifico utente. Per aprire e quindi visualizzare il contenuto dei documenti sarà necessario il download di ognuno di essi. Una volta scaricati, i documenti verranno salvati sul dispositivo e il sistema dovrà consentire di aprire visualizzare il contenuto dei documenti anche in modalità offline.

### 4.2 Analisi dei requisiti

Il lavoro di tesi mira alla realizzazione di un'applicazione mobile ma per una buona riuscita del sistema, prima di procedere con lo sviluppo vero e proprio è stata fondamentale la *fase di raccolta e analisi dei requisiti*, finalizzata all'acquisizione di tutte le informazioni utile per definire le funzionalità e caratteristiche che il prodotto finale dovrà offrire.

## Requisiti funzionali

I requisiti funzionali descrivono le funzionalità ed in particolare il comportamento del sistema a seguito di richieste o azioni eseguite da parte dell’utente. Di seguito sono riportate nel dettaglio le funzionalità che si vogliono sviluppare.

Il servizio permetterà unicamente ai clienti registrati di accedere ai documenti condivisi. Questa funzionalità permetterà al sistema di autenticare l’utente. In fase di login il sistema dovrà mostrare i campi in cui l’utente può inserire username e password, la correttezza dei dati inseriti verranno confrontati con i dati presenti nel database. Se i dati inseriti risultano corretti, l’utente verrà autenticato nel sistema e potrà accedere nell’applicazione, se i dati non sono corretti il sistema mostra l’errore all’utente e gli propone provare a reinserire i dati. Dopo il login l’utente avrà accesso ad una nuova schermata.

In questa schermata l’utente visualizzerà quali documenti sono disponibili per essere scaricati e quindi consultati. Il sistema mostrerà una lista documenti tra quelli disponibili per lo specifico utente. La lista di documenti disponibili non sarà uguale per tutti gli utenti, poiché gli utenti non avranno tutti gli stessi privilegi.

L’utente, una volta ottenuta la lista, avrà la possibilità di scaricare i documenti disponibili. I documenti scaricati saranno visibili in una finestra chiamata Archivio. L’applicazione permetterà di consultare l’archivio anche in modalità offline, quindi senza effettuare necessariamente il login. La funzione offline sarà disponibile solo per visualizzare i file scaricati dall’utente. Quindi sarà necessario, almeno una volta, loggarsi al sistema e scaricare i documenti.

Dopo che l’utente avrà scaricato i documenti, selezionando un determinato elemento, l’applicazione consentirà di aprire il documento PDF e l’utente visualizzerà il contenuto in una nuova schermata. L’utente dovrà avere un tempo limitato per consultare i documenti che infatti una volta scaricati dovranno essere dotati di scadenza.

Il sistema dovrà fornire agli utenti connessi la procedura per uscire dall’applicazione. Effettuando il logout l’utente verrà disconnesso dal sistema.

## Diagramma dei casi d'uso

Prendendo in considerazione quanto si è detto nell'analisi dei requisiti discussa nel paragrafo precedente, è stato utile costruire il diagramma dei casi d'uso mostrato in Figura 4.1. Il diagramma dei casi d'uso permette di descrivere le funzionalità e i servizi offerti dal sistema attraverso l'interazione che un utente il sistema stesso. Ogni *Caso d'uso* rappresenta una funzione del sistema.

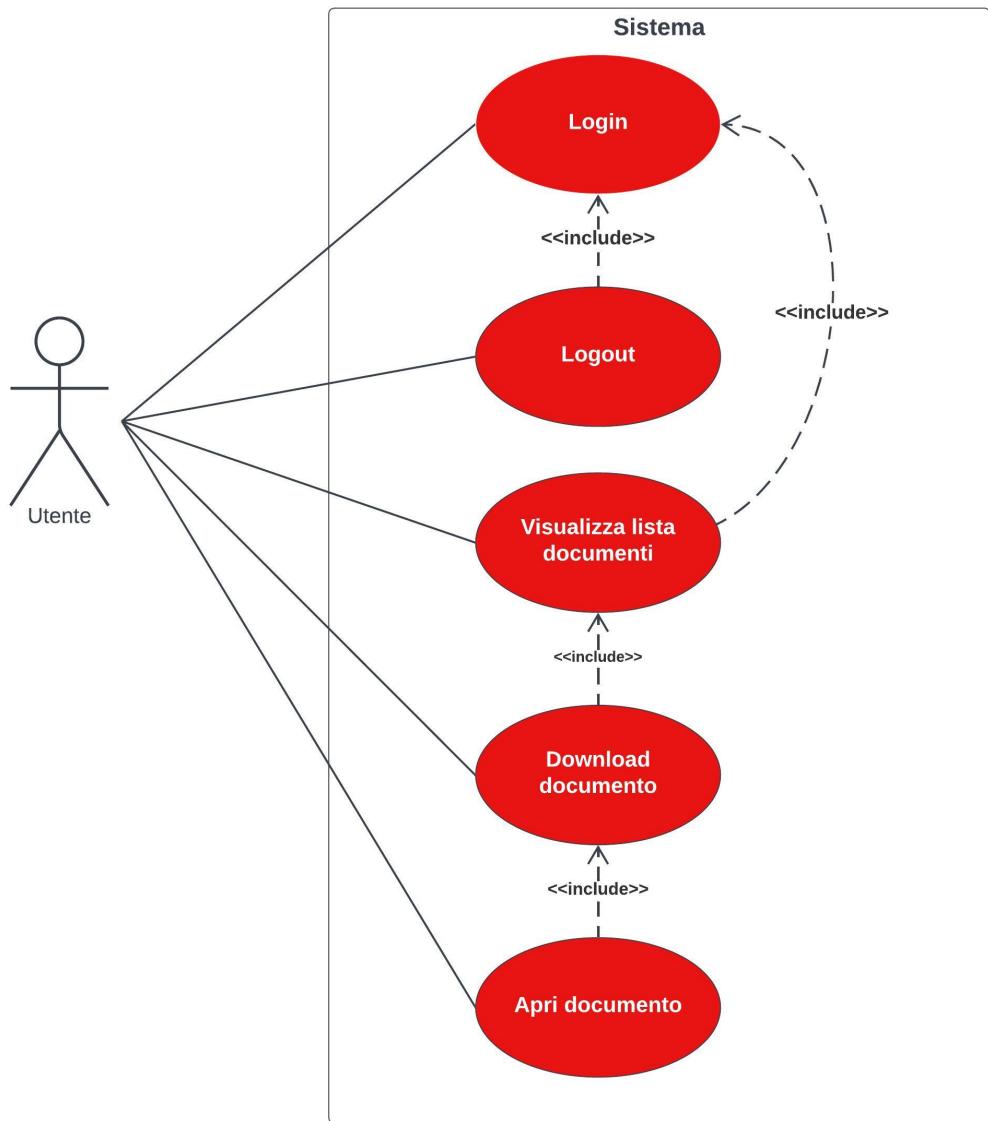


Figura 4.1: Diagramma dei casi d'uso

Per ciascuno dei *Caso d'uso* in seguito verrà illustrata una tabella dove viene riportato il nome , gli attori partecipanti, le precondizioni nelle quali sono eseguibili e la sequenza delle azioni svolte dagli attori e dal sistema, con l'obiettivo di mostrare il comportamento del sistema quando sollecitato da un attore.

## Login

<b>Nome</b>	Login
<b>Attori partecipanti</b>	Utente
<b>Precondizioni</b>	L'utente ha avviato l'applicazione
<b>Flusso di eventi</b>	<ol style="list-style-type: none"><li>Il sistema mostra i campi dove inserire le credenziali</li><li>L'utente inserisce l'username e password</li><li>Il sistema verifica se l'username e password sono corretti</li><li>L'utente accede nella sua area personale</li></ol>
<b>Eccezioni</b>	<p>Se le credenziali non sono corrette, l'utente non accede</p> <p>5.1 Il sistema permette all'utente di riprovare</p> <p>Si torna al passo</p>

## Visualizza lista dei documenti

<b>Nome</b>	Visualizza Lista
<b>Attori partecipanti</b>	Utente
<b>Precondizioni</b>	L'utente ha effettuato il login
<b>Flusso di eventi</b>	<ol style="list-style-type: none"><li>L'utente richiede di visualizzare la lista dei documenti</li><li>Il sistema mostra la lista dei documenti</li><li>L'utente visualizza i documenti disponibili per il download</li></ol>
<b>Eccezioni</b>	Nessuna

## Download documento

<b>Nome</b>	Download documento
<b>Attori partecipanti</b>	Utente
<b>Precondizioni</b>	L'utente ha effettuato il login e ha richiesto la lista di documenti
<b>Flusso di eventi</b>	<ol style="list-style-type: none"><li>Il sistema mostra la lista dei documenti</li><li>L'utente sceglie un documento della lista e clicca su Download</li><li>Il sistema scarica il documento scelto dall'utente</li><li>Il sistema mostra che il documento è stato scaricato</li></ol>
<b>Eccezioni</b>	Nessuna

## Apri documento

Nome	Apri documento
Attori partecipanti	Utente
Precondizioni	L'utente ha effettuato il download del documento
Flusso di eventi	<ol style="list-style-type: none"><li>Il sistema mostra la lista dei documenti scaricati</li><li>L'utente sceglie un documento della lista e clicca su Apri</li><li>Il sistema mostra il contenuto del documento</li><li>L'utente visualizza il contenuto del documento</li></ol>
Eccezioni	Se il documento è scaduto al passo 3 il documento non verrà mostrato

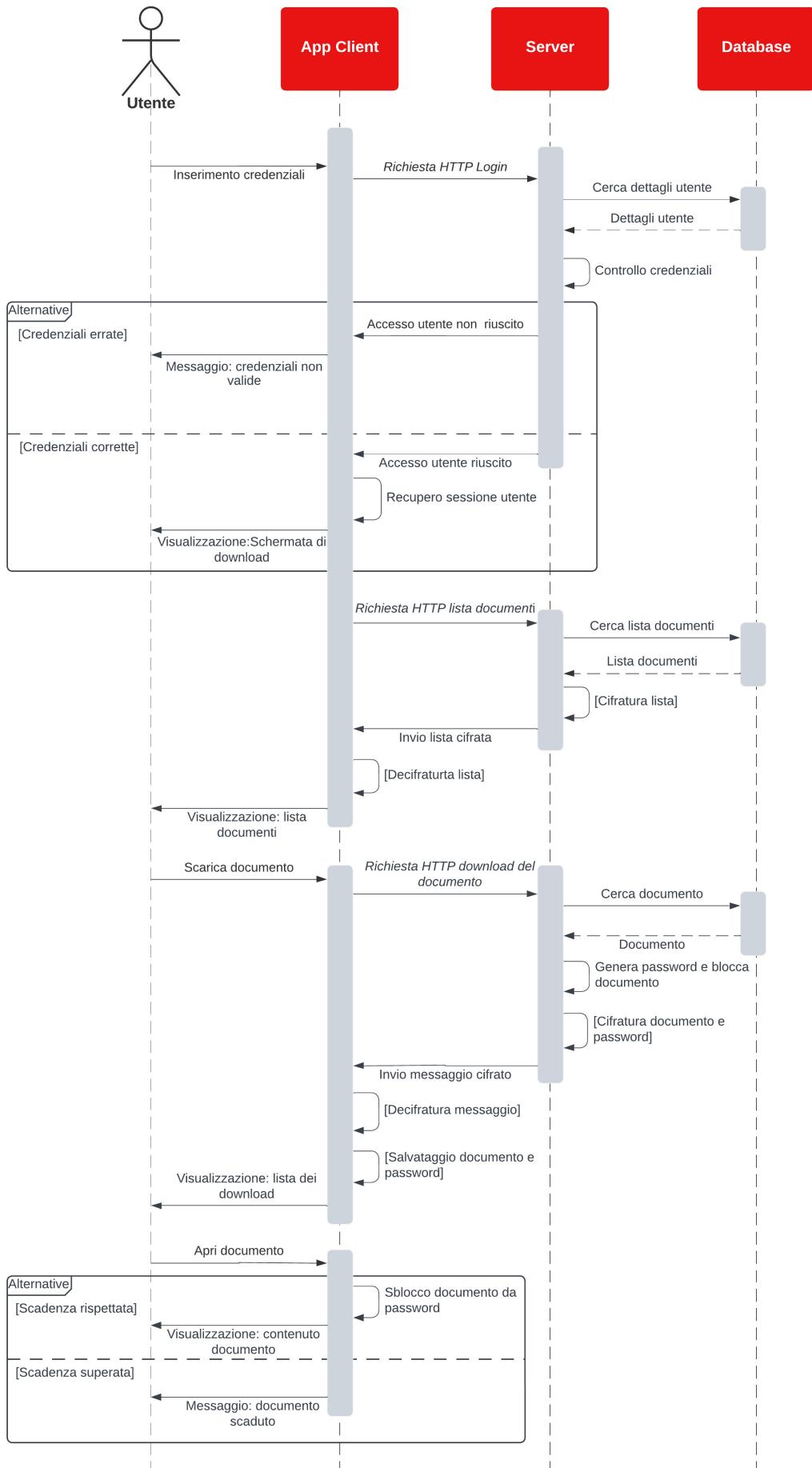
## Logout

Nome	Logout
Attori partecipanti	Utente
Precondizioni	L'utente ha effettuato il login
Flusso di eventi	<ol style="list-style-type: none"><li>L'utente clicca sull'icona di Logout</li><li>Il sistema chiede conferma all'utente</li><li>L'utente conferma</li><li>Il sistema disconnette l'utente</li><li>Il sistema mostra la schermata di login</li></ol>
Eccezioni	Se l'utente non conferma al passo 2 il logout non sarà eseguito

## Diagramma di sequenza

Oltre al *Diagramma dei Casi d'uso*, è stato utile costruire il *Diagramma sequenza* in Figura 4.2 con l'obiettivo di descrivere in maniera più chiara le interazioni tra le principali entità del sistema quando viene eseguita una determinata funzione in relazione al trascorrere del tempo e mettendo in evidenza l'ordine delle interazioni.

Figura 4.2: Sequence diagram



# Capitolo 5

## Implementazione

Una parte fondamentale dell'attività di questo progetto è stata lo studio dei diversi framework e librerie che avrei dovuto utilizzare per lo sviluppo dell'applicazione. In questo capitolo verranno illustrate e descritte in modo dettagliato tecnologie che ho approfondito ed utilizzato per la realizzazione dell'applicazione mobile SafeDoc Viewer.

SafeDoc Viewer è un applicazione mobile con architettura client/server, dove il client effettua richieste o accede alle informazioni messe disposizione da un server. È quindi stata necessaria l'implementazione di due componenti fondamentali: il front-end ovvero l'applicazione lato client, e il back-end ovvero le componenti software che risiedono e sono eseguite su un server. L'interfaccia con la quale l'utente può interagire appartiene al front-end, mentre la logica applicativa è suddivisa fra front-end e back-end. Un server back-end è un sistema software in grado di mettersi al servizio di un applicazione comunicando attraverso la rete, solitamente tramite il protocollo HTTP(hypertext transfer protocol): protocollo a livello di applicativo usato principalmente nella trasmissione di informazioni sul web. Il funzionamento di HTTP si basa su un meccanismo richiesta-risposta: solitamente il client manda una richiesta e il server è incaricato a risolvere le richieste del client. Un server back-end consente quindi alle applicazioni che vi si collegano di usufruire delle funzioni che mette a disposizione. Il client interagisce con il server attraverso l'uso delle REST API (REpresentation State Transfer e Application Programming Interface),che sono i metodi tramite cui un client può richiedere informazioni dal server seguendo il protocollo HTTP.

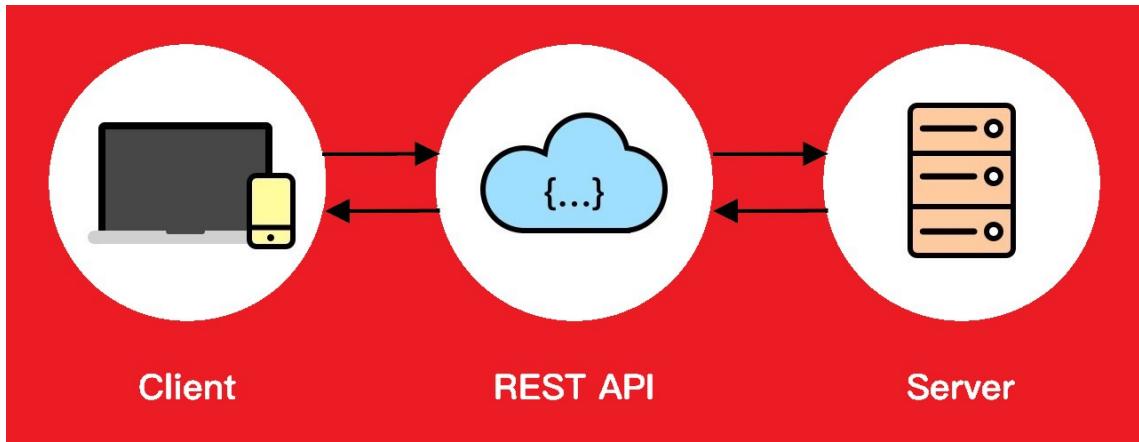


Figura 5.1: Struttura delle API REST

## 5.1 Applicazione lato server

Per lo sviluppo del modulo back-end ho utilizzato Node.js. **Node.js** è un runtime open source, sviluppato da Ryan Dahl e rilasciato come prima versione nel 2009, che permette di eseguire codice *Javascript* tramite il motore Javascript V8 di Google. Dopo un'attenta ricerca, ho scelto di usare Node.js principalmente per un motivo:



Figura 5.2: Logo del runtime Node.js

Fonte: [www.nodejs.org](http://www.nodejs.org)

Node.js infatti permette di eseguire codice JavaScript ed essendo JS attualmente uno dei linguaggi di programmazione più popolari e più richiesti nel mondo del lavoro, era di mio interesse personale approfondire ed apprendere questo linguaggio. Ho utilizzato questo runtime per lo sviluppo del modulo backend che fornisce le API necessarie all'applicazione: il modulo in questione, espone un indirizzo IP su cui è possibile inviare richieste con i metodi POST, GET a seconda delle azioni che si vogliono eseguire.

Le varie librerie e framework che ho usato nello sviluppo del modulo server:

- **Express** è un framework di essenziale importanza ed è stato ormai definito uno standard per lo sviluppo di applicazioni web e mobile con Node.js. Si tratta di una estensione di Node.js, utile per definire gli *endpoint* delle API, attraverso i

quali vengono inviate richieste a fronte dell'esecuzione di una particolare azione da parte dell'utente.

- **mongodb** è una libreria che consente agli sviluppatori di definire i modelli di dati per il database NOSQL MongoDB. Libreria essenziale per interfacciarsi con il database al fine di recuperare i dati necessari come credenziali utenti e documenti PDF.
- **crypto** è un modulo che fornisce funzionalità come crittografia, decrittografia e hashing di qualsiasi tipo di dati in un'applicazione Node.js. Usato per scopi di sicurezza come l'autenticazione dell'utente in cui si memorizza la password nel database in forma cifrata e per generare la chiave simmetrica.
- **qpdf** è una libreria che offre molte funzionalità utili per la gestione di documenti PDF, infatti oltre creare file PDF da zero è possibile modificare file già, manipolare l'elenco delle pagine, creare tabelle. In particolare, ho usato qpdf per proteggere i documenti PDF tramite password.

Una volta implementato il server, abbiamo un file con estensione `.js`, nel mio caso il nome completo è `mainServer.js`. Per avviare il server, su *Windows*, è necessario aprire il *Prompt dei comandi*, spostarsi nella directory dove è stato salvato il file ed eseguire il comando: '`node mainServr.js`'. Una volta avviato, server è impostato per restare in sull'indirizzo e sulla porta specificati. Nel mio caso l'indirizzo è `http://localhost:3000/` dove: *localhost* è un indirizzo interno al nostro computer e corrisponde all'indirizzo IP 127.0.0.1, mentre 3000 corrisponde al numero di porta. In generale il *localhost* viene utilizzato principalmente da sviluppatori che intendono testare programmi o applicazioni web poiché attraverso il localhost si può simulare una connessione ad internet.

## Struttura degli endpoint

Rispettando i principi REST, tutte le funzioni implementate sono accessibili attraverso l'esecuzione di richieste HTTP, da parte dei client, verso particolari endpoint. Ogni endpoint è identificato da un URL, quindi server API può ospitare uno o più endpoint, il che significa che accetterà ed elaborerà le chiamate dirette agli URL di tali endpoint. Un URL è un percorso che può gestire diversi metodi di richiesta. Ogni tipo di richiesta è un endpoint separato e in generale esistono cinque tipi di richiesta: GET, POST, PUT, PATCH, DELETE. Nella Tabella 3.1 sono riportati gli endpoint delle API sviluppati per gestire i vari processi e funzionalità dell'applicazione sul lato server.

Endpoint	Metodo	Descrizione
/login	POST	<p>Endpoint utilizzato per l'autenticazione.</p> <p>Vengono passati come parametri l'username password inseriti dall'utente. Username e password vengono confrontati con quelli presenti del DB, se l'username e password coincidono viene avviata una sessione per lo specifico utente.</p>
/getkey	POST	<p>Endpoint utilizzato per lo scambio della chiave simmetrica.</p> <p>Lo scambio della chiave simmetrica avviene tramite il sistema di cifratura a chiave pubblica/privata.</p> <p>Viene passata come parametro la chiave pubblica del client. Chiave pubblica del client e chiave privata del server cfreranno il messaggio che contiene la chiave simmetrica.</p> <p>Il messaggio cifrato e la chiave pubblica del server vengono inviati al client.</p>
/getAllPdf	GET	<p>Endpoint utilizzato per richiedere la lista dei documenti PDF.</p> <p>Il server si collega al database per recuperare la lista dei documenti disponibili.</p> <p>Una volta recuperata, la lista viene inviata all'utente.</p>
/getPdf/:id	GET	<p>Endpoint utilizzato per il download di un documento PDF.</p> <p>Viene richiesto il download di un documento specificando l'ID al quale il documento è associato.</p> <p>L'ID viene confrontato con quelli presenti nel DB, se l'ID coincide il documento PDF viene inviato all'utente.</p>
/logout	GET	<p>Endpoint utilizzato per il logout dell'utente.</p> <p>La sessione dello specifico utente viene chiusa.</p>

Tabella 5.1: Lista degli endpoint

## 5.2 Persistenza dei dati

La persistenza o l'archiviazione dei dati è affidata a un database, nello specifico **MongoDB**. MongoDB è un database NoSQL open source, *document-oriented*, che sfrutta il formato BSON, simile al JSON, per la memorizzazione e la rappresentazione dei dati.

**JSON** è l'acronimo di JavaScript Object Notation e si tratta di un formato testuale per la strutturazione di dati basato su elenchi ordinati. JSON è comunemente usato per scambiare informazioni tra client web e server web.



Figura 5.3: Logo MongoDB

Fonte: [www.mongodb.com](http://www.mongodb.com)

MongoDB consente di memorizzare i dati in *Collezioni* e *Documenti*, invece che in tabelle e righe come nei database relazionali tradizionali. Le *Collezioni* che sono l'equivalente delle tabelle di un database relazionale, comprendono insiemi di *Documenti*. Un Documento è l'analogia a un record o a una riga in una tabella di un database relazionale. Ho deciso di usare MongoDB insieme a Node.js poiché entrambi utilizzano il formato JSON ciò semplifica il passaggio dei dati tra le due tecnologie senza doverli convertire in un formato diverso.

Durante la fase di progettazione, è stato necessario costruire un modello dei dati per comprendere come verranno organizzati i dati all'interno del database.

La Figura 3.1 mostra il modello entità-relazione contenente i dati che saranno trattati dal sistema: vengono riportati gli attributi e le relazioni tra i dati.

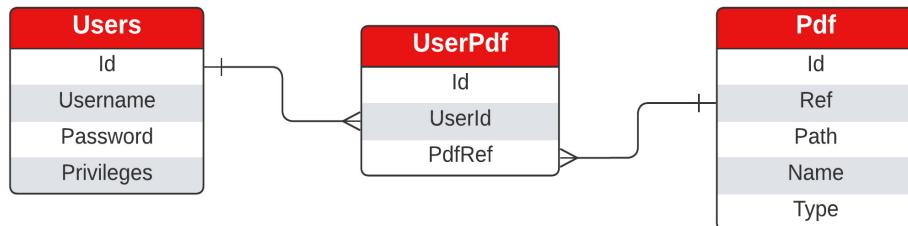


Figura 5.4: Diagramma ER del database

Quindi una volta costruito, il database dovrà gestire i dati del sistema in tre collezioni, ogni collezione è composta da diversi documenti dove ogni documento contiene un campo `_id` riservato all'uso della chiave primaria, ed ha valore univoco ed immutabile. Le collezioni gestite dal database sono le seguenti:

- **Users:** in questa collezione vengono gestiti e archiviati i dati relativi agli utenti che hanno accesso al sistema. Ogni documento in questa collezione tiene traccia e memorizza i dati di ogni singolo utente. Per ogni utente si tiene conto di *Username*, *Password* e tipo di *Privilegi*. In particolare la password viene archiviata il formato MD5 e non in chiaro. Nella Figura 3.2 viene mostrato

```
{
  "_id": {
    "$oid": "640b7e80521e03f677c6e712"
  },
  "username": "marioRosa",
  "password": "a6fb6cbdebb42d6cd94370e24ffd46b2",
  "privileges": "1"
}
```

Figura 5.5: Esempio di documento della collezione Users

un esempio di documenti nella collezione Users.

- **Pdf:** in questa collezione vengono gestiti i dati relativi ai documenti PDF che verranno condivisi con gli utenti. Ogni elemento di questa collezione tiene traccia di un documento PDF, in particolare per ogni documento PDF abbiamo i seguenti parametri: *Name* che è appunto il nome del PDF; *Ref* è un riferimento di tipo numerico che serve ad identificare il modo univoco il PDF al di fuori del DB, quindi quando viene richiesto o condiviso con l'utente; *Path* è utile per tracciare il percorso del file PDF; *Type* è un parametro utile per classificare i PDF, poiché in base al tipo, il documento PDF viene condiviso o meno con un tipo di utente. Nella Figura 3.2 viene mostrato un esempio di

```
{
  "_id": {
    "$oid": "63fbab8c82038e133428c79c"
  },
  "ref": 1,
  "path": "pdf/Unib.pdf",
  "name": "Logo Unibas",
  "type": "base"
}
```

Figura 5.6: Esempio di documento della collezione Pdf

documento nella collezione Pdf.

- **UserPdf:** questa collezione viene costruita dinamicamente all'interno del database e ogni documento tiene traccia dei PDF scaricati da uno specifico utente. Sono memorizzati i dati relativi all'*Id* dell'utente coinvolto e del *Ref* del documento PDF scaricato. Esempio mostrato in Figura 3.4.

```
{
  "_id": {
    "$oid": "64159e06f1356f98cb7c3130"
  },
  "idUser": "640b7f3c521e03f677c6e714",
  "idPdf": 1
}
```

Figura 5.7: Esempio di documento della collezione UserPdf

Per poter comunicare in modo corretto con il database, il server Node.js avrà bisogno di collegarsi al server MongoDB. Il collegamento MongoDB – Node.js verrà effettuato attraverso la libreria ***mongodb*** citata in precedenza, che permette l'utilizzo delle classiche operazioni CRUD con un database MongoDB. Una volta avviato il server, esso avvierà la connessione verso il database MongoDB che è in ascolto sulla porta predefinita *localhost:27017*. Le seguenti immagini mostrano il codice di avvio del server e la connessione al database:

```
app.listen(3000, () => {
  console.log(`Example app listening at http://localhost:3000`)
  connectMongo()
})
```

Figura 5.8: Codice avvio server

```
async function connectMongo() {
  client = new MongoClient("mongodb://127.0.0.1:27017")
  await client.connect()
  console.log("Connected successfully to server")
  db = client.db("serverTesi")
}
```

Figura 5.9: Codice connessione al DB

### 5.3 Applicazione lato client

In generale, le applicazioni mobile sono dei programmi software sviluppati per funzionare su dispositivi mobili come smartphone, tablet, smartTv ecc. L'interfaccia grafica è la parte fondamentale di un'applicazione mobile, la quale deve garantire un'interazione efficace ed efficiente tra l'utente e le funzionalità offerte. Un'app deve essere quanto più veloce possibile, con un'interfaccia fluida e reattiva.

**SafeDoc Viewer** è un applicazione mobile per Android sviluppata con *target API version 33* e *minimum API version 28*, cioè può essere installata su dispositivi con

almeno versione Android 9.0 (Pie) fino ai più moderni con versione Android 13(Tiramisù). Se il livello API di un dispositivo Android è inferiore al livello API minimo specificato per l'app, il dispositivo Android impedirà all'utente di installare l'app. SafeDoc Viewer è stata sviluppata su Android Studio, in linguaggio **Java**, seguendo il pattern architetturale MVC dove la struttura software è suddivisa in tre componenti principali Modello Vista e Controllo, ed ogni componente lavora in modo indipendente e disaccoppiato dagli altri.

**Android Studio** è l'IDE(*Integrated Development Environment*) ufficiale per lo sviluppo di applicazioni Android. Tra le varie funzionalità, Android Studio consente, tramite un emulatore, di creare e configurare un dispositivo virtuale Android(AVD) che ci permette di testare la nostra applicazione. Le librerie Java usate durante lo sviluppo dell'applicazione:

- **OkHttp**: libreria open-source per Java e applicazioni Android che permette di creare le richieste HTTP verso una risorsa specificando l'URL e il tipo di richiesta che può essere GET, PUT, POST o DELETE. L'utilizzo di questa libreria è essenziale per gestire la comunicazione client - server e le relative richieste/risposte HTTP.
- **Gson**: libreria open-source che permette di trasformare in modo veloce ed efficace un oggetto Java nel formato JSON e viceversa. Ho usato questa libreria per tradurre i messaggi di richiesta/risposta tra client e server che usano il formato JSON per lo scambio dei dati.
- **Android PdfViewer**: Libreria che consente gestire in modo completo i documenti PDF su Android. Tra le varie funzioni, consente la visualizzazione di documenti PDF sul dispositivo.

Avendo analizzato le principali librerie utilizzate durante lo sviluppo dell'applicazione, nei prossimi paragrafi vedremo come sono state utilizzate per la realizzazione delle varie funzionalità. Ogni funzionalità verrà discussa nel dettaglio tramite l'illustrazione delle interfacce grafiche e il flusso di interazioni tra utente e le schermate.

## Schermata Home

Dopo aver avviato l'applicazione, la *Schermata Home* è la prima schermata che viene mostrata all'utente. Essendo una schermata appunto di benvenuto, è stata pensata per essere semplice ma non banale e composta di pochi elementi chiave, in modo da essere il più intuitibile possibile per l'utente. Nella Figura 5.10 viene riportato lo *screenshot* della Schermata Home. Dallo screenshot possiamo notare gli elementi chiave che compongono la schermata: nella parte superiore il nome della applicazione, verso il centro il logo dell'Università degli Studi della Basilicata e solo nella parte inferiore ci sono due icone interagibili, che porteranno l'utente in altre due nuove schermate.

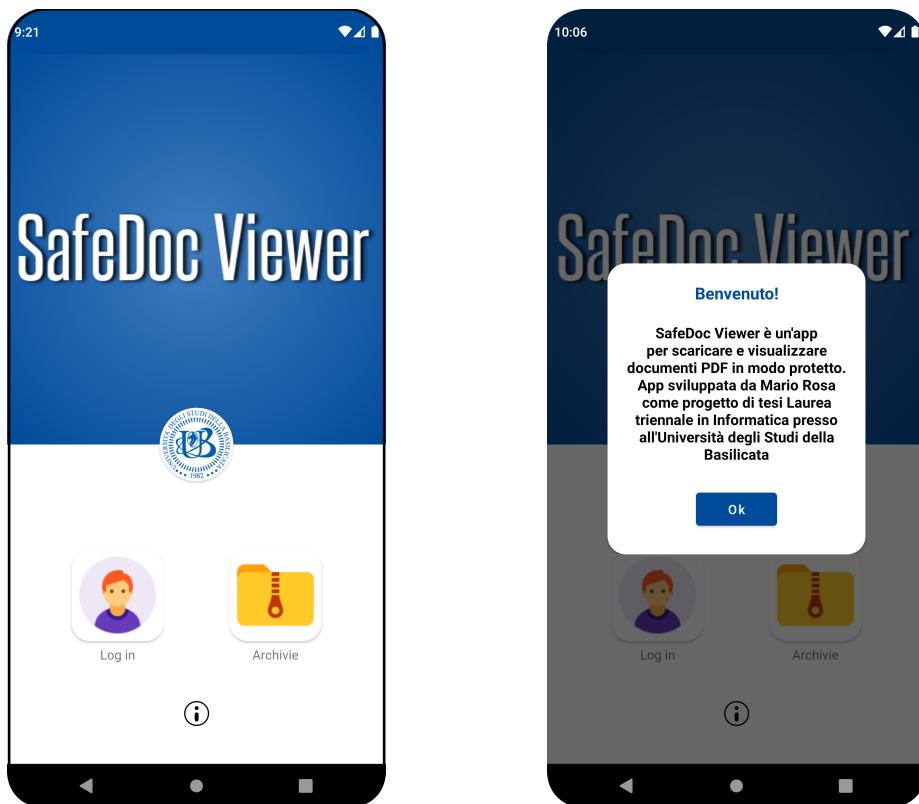


Figura 5.10: Screenshot schemata home



Figura 5.11: Screenshot messaggio di benvenuto

Cliccando sul tasto *Info* si aprirà una piccola finestra con un messaggio di 'Info e benvenuto!', mostrato in Figura 5.11.

## Schermata Login

Questa schermata permette all’utente di accedere alla propria area riservata attraverso l’inserimento delle credenziali ovvero *username* e *password*. Qualora i dati inseriti non siano corretti, l’accesso viene negato. In questa fase il client effettua

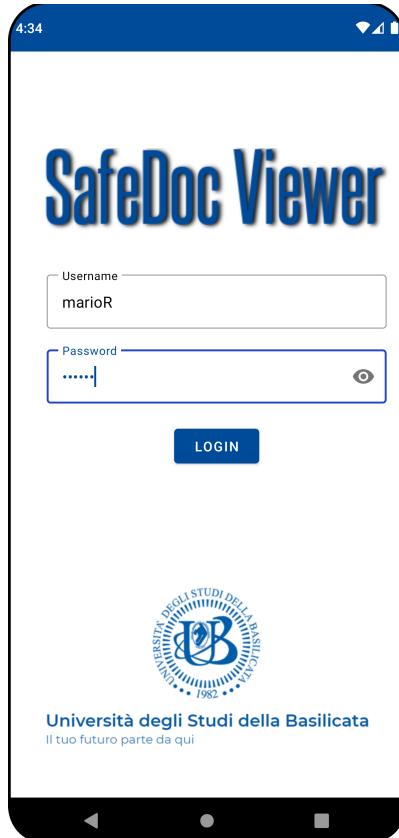


Figura 5.12: Schermata Login

una richiesta HTTP POST all’ URL *localhost:3000/login* dove vengono inviati come parametri della richiesta l’username e password inseriti dall’utente. Sarà il server a confrontare le credenziali ricevute dal client con quelle già presenti nel database. Come detto in precedenza all’interno del database sono salvate le hash MD5 della password e non la password in chiaro.

**MD5** (Message Digest Method 5) è un algoritmo di hashing che prende in input una stringa di lunghezza arbitraria producendone in output un’altra a 128 bit, rappresentata come una sequenza di 32 cifre esadecimale. L’algoritmo utilizza una funzione hash per convertire i dati standard in un formato non riconoscibile. Queste funzioni hash sono un insieme di calcoli matematici che trasformano le informazioni originali nei loro valori hash, noti come hash digest o digest.

Quindi la password presa in input prima di essere inviata al server viene data in

pasto ad un algoritmo di hashing. Quando un utente tenta di accedere, inserisce la password; la password inserita passa attraverso la funzione hash per generare un *digest*; se il *digest* appena creato corrisponde a quello sul server, l'accesso viene concesso. Come possiamo vedere nell'esempio in Figura 5.5 la password è una salvata in formato MD5.

#### L'algoritmo di hashing MD5

```
public static String md5(final String s) {
    try {
        // Create MD5 Hash
        MessageDigest digest = java.security.MessageDigest.getInstance("MD5");
        digest.update(s.getBytes());
        byte[] messageDigest = digest.digest();
        // Create Hex String
        StringBuilder hexString = new StringBuilder();
        for (byte b : messageDigest) {
            String h = Integer.toHexString(0xFF & b);
            while (h.length() < 2)
                h = "0" + h;
            hexString.append(h);
        }
        return hexString.toString();

    } catch (NoSuchAlgorithmException e) {
        e.printStackTrace();
    }
    return "";
}
```

## Schermata Download

Dopo il login l'utente ha finalmente accesso a quella che possiamo definire la schermata più importante: la *Schermata di Download*. In questa schermata, come si può vedere in Figura 5.13, viene mostrata all'utente una serie di documenti PDF. Per scaricare/aggiornare la lista dei documenti è necessario cliccare sul pulsante **UPDATER**, questo farà partire una richiesta HTTP GET all'URL *localhost:3000/getAllPdf* verso server. A questo punto il client riceverà dal server come risposta un array in formato JSON, dove sono riportati i dati dei file PDF disponibili per l'utente. Attraverso la funzione chiamata *parsePdfListFromJson*, la lista JSON viene analizzata e tradotta in una lista di Oggetti di tipo *Documento* in modo da essere gestita e modellata all'interno dell'applicazione. A questo punto la lista viene mostrata sullo schermo. Selezionando un singolo documento compare l'opzione '**Scarica**'. Cliccando su **Scarica** il client farà partire una richiesta HTTP GET all' URL *localhost:3000/getPdf/:id* ma al posto di *:id* viene passato il valore **ref** del documento

### Risposta server: Lista JSON

```
[
  {
    "ref":1,
    "name":"Logo Unibas"
  },
  {
    "ref":2,
    "name":"Use case"
  },
  {
    "ref":3,
    "name":"Sequence diagram"
  },
  {
    "ref":4,
    "name":"Document Security"
  }
]
```

selezionato. Il server risponderà inviando il file PDF con il **ref** specificato, il client una volata scaricato il file lo salverà nella memoria del dispositivo.

Nella Figura 5.14 viene mostrata l'opzione **Scarica** una volta selezionato il documento "Sequence diagram"; i documenti "Logo Unibas" e "Use case" sono già stati scaricati e salvati nella memoria del dispositivo e questo ci viene segnalato, attraverso un icona che solitamente indica il *download*, a destra del nome dei documenti.

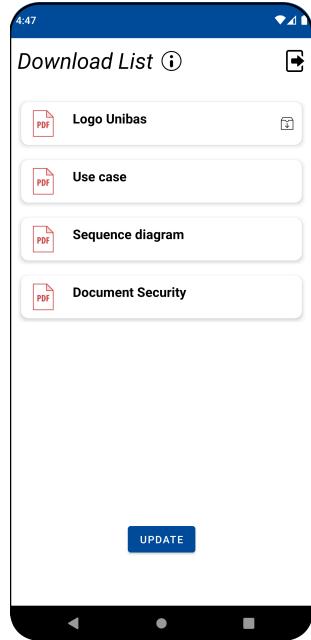


Figura 5.13: Screenshot schemata download

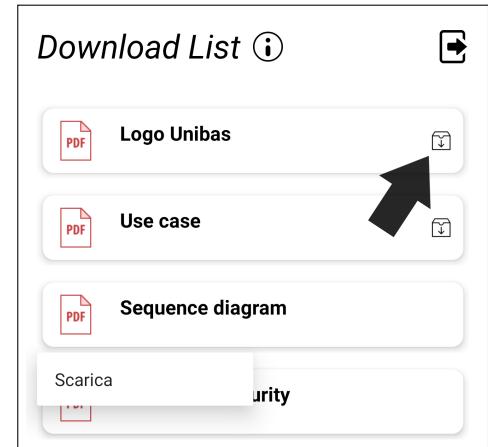


Figura 5.14: Opzione Scarica dei documenti e icona download

## Schermata Archivio

Nella schermata Archivio è possibile visualizzare tutti i documenti PDF che sono stati scaricati dall'utente. Per ogni documento scaricato viene riportata il nome e la data di scadenza. Oltre la data di scadenza i documenti non possono essere aperti. La data di scadenza viene inviata in fase di download insieme al documento. Per verificare se il documento è scaduto, la data viene confrontata con la data segnata dal dispositivo con il metodo *isExpired*. Oltre il giorno di scadenza i documenti verranno segnati come *Scaduti* (Figura 5.16) e non posso essere consultati.

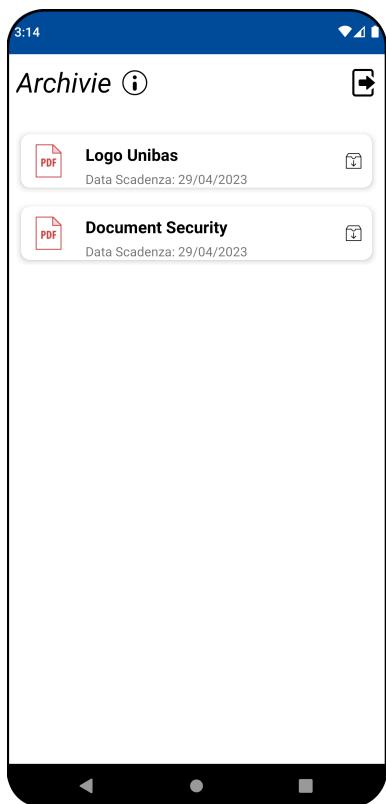


Figura 5.15: Screenshot Archivio

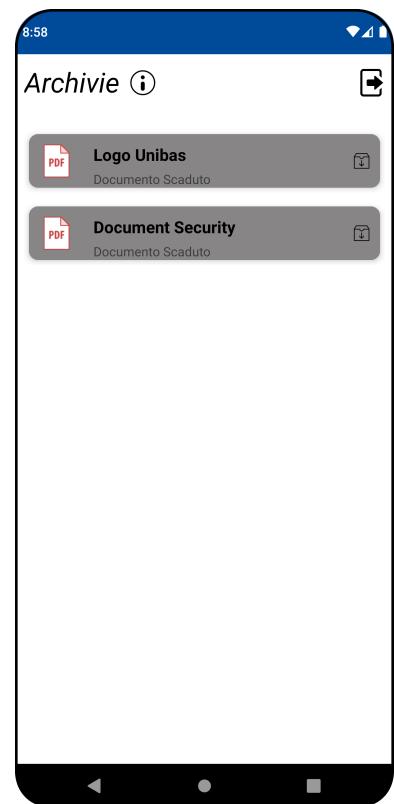


Figura 5.16: Documenti scaduti

```
private boolean isExpired(Date scadenza){  
    Date today = new Date();  
    System.out.println(today.getTime());  
    if(today.getTime() > scadenza.getTime()) {  
        return true;  
    }  
    return false;  
}
```

L'archivio è accessibile in modalità offline ed è quindi possibile consultare i documenti senza effettuare necessariamente l'accesso. Anche in modalità offline, i documenti saranno consultabili fino al giorno della loro scadenza.

## Schermata di Visualizzazione del Documento

Cliccando su uno dei documenti presenti nell'archivio, passiamo nella schermata di Visualizzazione. come detto in precedenza, se il documento è ancora valido, fino alla data di scadenza è possibile aprirlo e visualizzarne il contenuto anche in modalità offline. Una volta cliccato sul documento, esso viene caricato dalla memoria e mostrato sullo schermo. Il documento può essere anche zoomato e in caso si documenti multi pagina, è possibile scorrere tra le altre pagine.

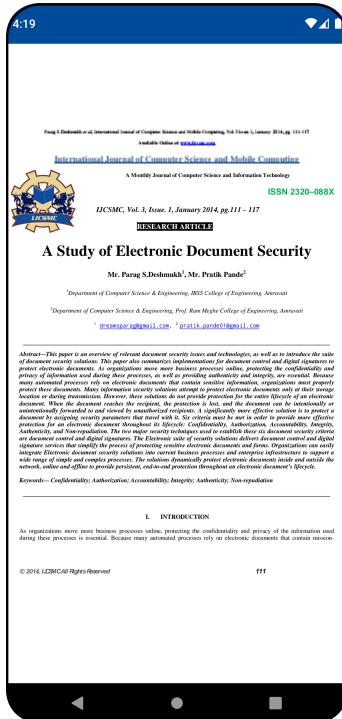


Figura 5.17:

La libreria che in Java e su Android offre la completa gestione del formato PDF è **Android PdfViewer**. Per usare PdfViewer è necessario aggiungere nel layout grafico il seguente componete:

```
<com.github.barteksc.pdfviewer.PDFView
    android:id="@+id/pdfView"
    android:layout_width="match_parent"
    android:layout_height="match_parent"/>
```

Poi, nel *Fragment* relativo alla visualizzazione dei documenti, nel metodo *onViewCreated()* il seguente codice carica e mostra sullo schermo il *File PDF* selezionato:

```
pdfView.fromFile(File).load();
```

## 5.4 Riservatezza

Per mantenere la riservatezza dei documenti e dei dati in fase di distribuzione, quindi durante lo scambio di messaggi tra il client e il server, sono stati implementati metodi crittografici. La crittografia è una tecnica con cui rendere indecifrabili i messaggi tramite degli algoritmi, consentendone la comprensione solo grazie a specifiche chiavi di lettura. Le chiavi di lettura sono note solo agli utenti autorizzati, che possono così accedere alle informazioni decodificandole.

**Crittografia simmetrica:** nella crittografia simmetrica esiste una sola chiave e tutte le parti coinvolte utilizzano la stessa chiave per crittografare e decrittografare le informazioni. I principali algoritmi usati nella crittografia simmetrica sono DES (Data Encryption Standard), 3DES (Triple DES) e AES (Advanced Encryption Standard).

**AES** o *Advanced Encryption System* è uno degli algoritmi di crittografia simmetrica più comunemente utilizzati oggi, è disponibile in implementazioni a 128-bit, 192-bit e 256-bit. AES-256 bit fornisce il più forte livello di crittografia.

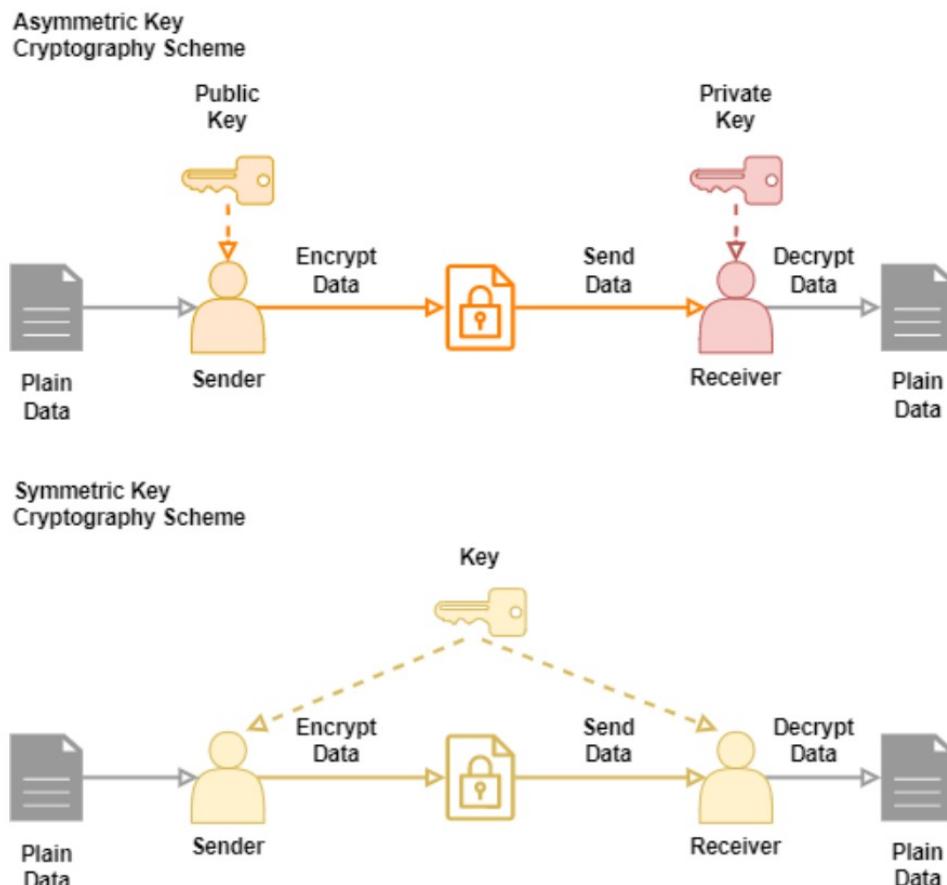


Figura 5.18: Schema di crittografia a chiave asimmetrica e simmetrica  
Fonte: [www.baeldung.com](http://www.baeldung.com)

**Crittografia asimmetrica:** conosciuta anche come crittografia a coppia di chiavi, è un tipo di crittografia dove a tutte le parti coinvolte nella comunicazione è associata una coppia di chiavi: la chiave pubblica, e la chiave privata. Entrambe funzionano complementari all'altra, il che significa che un messaggio crittografato con una di esse può essere decifrato solamente dalla sua controparte. Visto che la chiave privata non può essere calcolata dalla chiave pubblica, quest'ultima è generalmente disponibile al pubblico. **RSA** (*Rivest-Shamir-Adleman*) è uno dei sistemi di crittografia asimmetrici più popolare e più riusciti ad oggi.

I metodi di crittografia simmetrica presentano molti vantaggi per le comunicazioni tra client e server. Ad esempio, questi metodi in genere sono più performanti durante la codifica e decodifica di messaggi di grandi dimensioni rispetto ai metodi di crittografia asimmetrica che sebbene siano più sicuri risultano computazionalmente più onerosi. Tuttavia, la sfida principale relativa alla crittografia simmetrica è trovare un modo sicuro per condividere la chiave crittografica tra le entità autorizzate. Tra le diverse strategie per lo scambio sicuro della chiave simmetrica tra le entità, ovvero il client e il server, la strategia da me adottata è chiamata **meccanismo di incapsulamento della chiave** o più semplicemente **KEM**, dall'inglese **key encapsulation mechanism**. In pratica, una delle entità coinvolte nella comunicazione crea la chiave simmetrica e la cifra utilizzando la chiave pubblica fornita da una seconda entità. Quindi, la prima entità invia la chiave simmetrica codificata alla seconda entità, che la riceve e la decrittografa con la chiave privata adeguata. Oltre a codificare semplicemente la chiave simmetrica con una chiave pubblica, la prima entità può anche firmare la chiave simmetrica con una chiave privata. Pertanto, la prima entità fornisce anche una chiave pubblica alla seconda, consentendole di verificare l'autenticità della chiave simmetrica ricevuta.

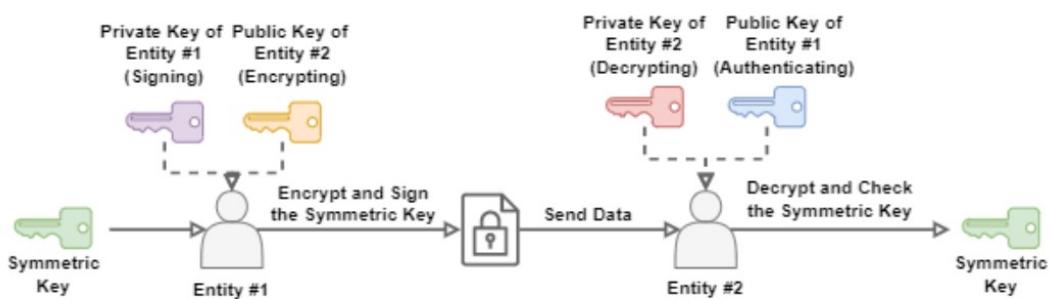


Figura 5.19: Meccanismo di incapsulamento della chiave  
Fonte: [www.baeldung.com](http://www.baeldung.com)

## Lato client

Sul lato client ho usato JCA o Java Cryptography Architecture è il framework java per la crittografia che fa parte della API di sicurezza di Java, ed è stato introdotto nei pacchetti *java.security* e *java.crypto*. In fase di avvio, quando l'utente avvia l'app SafeDoc Viewer, viene generata la coppia di chiavi asimmetriche pubblica e privata con il metodo *generateKeys*. Il framework da utilizzare è KeyPairGenerator, come mostrato nel codice seguente. La coppia di chiavi è restituita nel bean KeyPair caratterizzata da due sole proprietà in sola lettura: *publicKey* e *privateKey*. Una volta generate le chiavi vengono conservate all'interno dell'applicazione in formato PEM. PEM (originariamente un acronimo per "Privacy Enhanced Mail") è un formato contenitore molto comune per certificati e chiavi digitali.

Codice della funzione che genera le chiavi RSA

```
public static void generateKeys() {  
    try {  
        // Generate a key pair  
        KeyPairGenerator keyPairGenerator =  
            KeyPairGenerator.getInstance("RSA");  
        keyPairGenerator.initialize(3000);  
        KeyPair keyPair = keyPairGenerator.generateKeyPair();  
        // Get the public and private keys  
        byte[] publicKeyBytes = keyPair.getPublic().getEncoded();  
        byte[] privateKeyBytes = keyPair.getPrivate().getEncoded();  
        // Convert the keys to PEM format  
        String publicKeyPEM = toPEM(publicKeyBytes, "PUBLIC KEY");  
        String privateKeyPEM = toPEM(privateKeyBytes, "PRIVATE KEY");  
        // Store the keys  
        Modello.putBean("clientPublicKeyPEM", publicKeyPEM);  
        Modello.putBean("clientPrivateKeyPEM", privateKeyPEM);  
    } catch (NoSuchAlgorithmException e){  
        e.printStackTrace();  
    }  
}
```

### Esempio di chiave pubblica in formato PEM

```
-----BEGIN PUBLIC KEY-----  
MIIBkjANBgkqhkiG9w0BAQEAAOCAX8AMII BegKCAXEAxprFMOnbh  
FcsJjTsYko0XtsBjdLdxqHR2rSFX4YIokEYc61hJC3qLrjdJQjd0  
stI2TGBgEM0+E2ZYQuB8zuLr9FiDsHfy4Z7HxtR+R2oboH3Zrm4uv  
P1ToXd13NgWbYKrsrFP/J12MN/5FdYmyVpLshboZsp1ZzHtdHuV4e  
uLS4fYx6WLwgRP7Z8CPJW1+EZW4IH2uHJSPOnRej1+Jpi2OV0Ntt  
YKC1T2M74L3JSPXENvGXvj2fqyhuHekB9C7FTu0cY4xS8j4nhmsBk  
V7yL8nOoexbgX3ZDb+Irhk5ZKVjrH1Udk/Hgw2/vV3FI/xfLi+VIi  
mtK2LERd5d47/VGOJcbYQkD93yqFUynV9iNPZsEYuri52kliVi7vn  
1jfj8YjuCoaU6mv9V9GTkSH9/XR2Qh/B2ThpNSyYg7ETQI0Z06UFn  
2f2X1NbipxTEaqqsyUUe8LX1g5VvWzndVn8KdJpB5Dyqr5AiEJsur  
gtnYECAwEAAQ==  
-----END PUBLIC KEY-----
```

### Esempio di chiave privata in formato PEM

```
-----BEGIN PRIVATE KEY-----  
MIIGtQIBADANBgkqhkiG9w0BAQEFAASCBp8wggabAgEAAoIBcQDGmsUw6duEVywmN0x  
iSiVe2wGN0t3GodHatIVfhgi1QRhzrWEkLeouuN01CN3Sy0jZMYGAQw74TZlhC4Hz0  
4uvOWIOwd/LhnsfG1H5Hahugfdmubi68/V0hd3Xc2BZtgquysU/8mXYw3/kV1ibJWku  
yFuhmynVnMe10e5Xh64tLh9jHpYvCBxE/tnwI8laX4R1bggfa4c1I/SdF6PX4mmLY5X  
Q221goLPYzvgvc1I9cQ28Ze+PZ+rKG4d6QH0LsV045xjjFLyPieGawGRXvIvyc6h7F  
uBfdkNv4iuGT1kpW0sfVR2T8eDDb+9XcUj/F8uL5UiKa0rYsRF313jv9UY4lxthCQP3  
fKoVTKdX2I09mwRi6uLnaSWJLu+fWMWPxi04KhpTqa/1X0ZORIf39dHZCH8HZ0Gk1L  
JiDsRNAihk7pQWfZ/ZeU1uKnFMRqqzJRR7wtfWD1W9b0d1Wfwp0mkHkPKqvkCIQmy6  
uC2dgQIDAQABaoIBcA+LA029r6Fe1SS0VZ+noB96xkyCPVS37choEsL92F0z13U0IT6  
buNPqtGxmP5RRfNndRCYqJNP23zP4sUWJA0cDtYTHQeHxMLQyVLcgqSw1DAKK4fD2dZ  
CBeF6uJ6wt6VMGzOtazWmcrHbI4J8YUmn0/2HQAWbjPCU4wfM+63aXJi01N1oFdyCCa  
NWI22sZI3Wq2vsn+t1fvxXAZ7oexv4LaK1pqIdhDG413/bKsB7m/Cjkm1TKdTf3Y/84  
kPC1y7q2Hq877XmUvzMeDvSrt18RFx0JFI/VN9cLEvwQE13YY0rN0rG4DIHuAW/oUBC  
Z4pkKwkMEKf++ZxM+Bc7vXRNzAHTs0k0cAaccRkPQmKpgMU8Af7utHtGiLzYQYh63uE  
2zc8WXs/Lcty5M8cUGUiP3BiGhaX/q/LmhbfwwUxLtZXiKSA/VvVguEXYg7/2/zRdYL  
1DWXfyBX0k7G+iRltbw6qAkyFfHLBONSELLmF11AoG5A0wXX/rha9ar6LWoRtGoLsJF  
NsVs/mMwn67Mw1ErqXM0hi0PEHyTqGEsDTfiAaNanduXmUdkE8Yk+t/zx/qoKxAFX8  
OhycpsrPLwD82H4ybk+rfzhmhFhz1x7yglEGuqTm2W+iaoYlpSZAu2d38tZHNPSGhEX  
T0qcPC84Lz5j+VonvPltumzTqyFAajP8qKCezG6JqRomk611czsa8S0m0CAjuyIgTsz  
z0CMoIbElb2ZGqDx3FifKcCgbkA11omW0pG1w+ymEwFLAETuz6Q0Vx3tFL0FXxAABae  
Ghb02124SfiS9FzRh8bFUWJ5REHEw0L15/GR22d7D1fCVBs8BviK01hPrUAE1Xz81kK  
HLMe8YNQQccpF1Vj8Y113ke1mQABV9KSvq5vZln/SMUhcxZapZeb17HcuF66WddMXzJ  
YiQWufbeEkvF7GEEMn9UzwSBNoWz7ff5Ur3wYASS1Gsiy075FmHNoaBC/DJw90pnTM8  
3v5lwKBuQC/bMyDzNb1Pnn3ILqbNnSDTD975XL1wNhEkP41CHPiV0obqsVBXS/aagp1  
rziIt9JFIBY1gm0J1PHK1tyIEm0ohmVnmc338RVWNvRvzGCUT2kF3eVsXnJhh+YFiWX  
sQ8QuaS/oT8h+dAITiAlatpY52+GfLMvMurwTYDS1jzC3iYovPUgiXQK2fsFCCIemn3  
ogo77h5Qd8ruZLWjzNqMf6rtBEutBekewG5aAyYeTi4fDVdHf39QCtAoG4Zg8IS21D5  
QWAe2LTtKYSBtRA56et51zE5oXLAv+1D+I/Sln0I+hVJM/0UsenjclgdTPCH1PVnIXE  
vT10fV6j7x41FYympkCdChtj0MC6RoSi8Vr7cNwvqu3UilHw019aJmNe6Qf/tIBz7J1  
gwk1zfP2xIH1tbKi/4JAe3QRVtCNFjSS9uUljbMAFm+xCa6SYBE2HpCnsVHKJszor33  
dFnWadZYWAfmTnAtduKvvDvfgokwUBuIbRowKBuHkwweDgVF2gjUz3em15WhGsm8KrT  
fU304gT5bm3u7VRp7YmskHOHNc2PvxjcDtISvkOKPvXINdFvSdQ3fNLg0TrkujdHiiH  
IjYBXSp6D0jL5By/+9eFU6byzoGiSuVqhWUVe1h7KGW/Ngalug6gocP4WtfGnTL1w0Q  
7JIxDpr7GeUbb0yLg6A9mod/WqhzATA+J1Xp+85Gpu5SXqatiMp83Zg0H0JN6/bAe8G  
mf aBo8zXq2+zBr6FA=  
-----END PRIVATE KEY-----
```

## Lato server

Sul lato server invece, una volta avviato, vengono generate sia la coppia di chiavi pubblica e privata che la chiave simmetrica. Per generare le chiavi ho usato la libreria **crypto**.

Metodi per generare chiavi asimmetriche e chiave simmetrica sul server

```
//asymmetric keys
const { publicKey, privateKey } = crypto.generateKeyPairSync("rsa", {
  modulusLength: 2048,
  publicKeyEncoding: {
    type: "spki",
    format: "pem",
    padding: crypto.constants.RSA_PKCS1_OAEP_PADDING,
  },
  privateKeyEncoding: {
    type: "pkcs8",
    format: "pem",
    padding: crypto.constants.RSA_PKCS1_OAEP_PADDING,
  },
})
//symmetric key
const initVector = crypto.randomBytes(16)
const securitykey = crypto.randomBytes(32)
```

## Trasporto delle chiavi

Dopo il login, il client invia una richiesta HTTP POST a *http://localhost:3000/getKey* verso il server e nel corpo della richiesta viene passata la chiave pubblica del client. A questo punto il server avvierà il meccanismo di incapsulamento KEM descritto il precedenza come segue:

1. Firma: Il server codifica la chiave simmetrica con la sua chiave privata;
2. Dopo la firma, il server codifica il messaggio ottenuto nel passaggio 1) con la chiave pubblica del client.
3. Il server invia al client il messaggio JSON contenente la chiave incapsulata e la chiave pubblica del server;
4. Il client riceve il messaggio cifrato e a questo punto, per estrarre la chiave simmetrica, decifra il messaggio con la propria chiave privata e poi con la chiave pubblica del server.
5. D'ora in avanti, la comunicazione tra client e server sarà sicura e protetta poiché tutti i messaggi scambiati saranno cifrati/decifrati con la chiave simmetrica.

Quindi nelle varie funzionalità viste in precedenza in cui il server invia la lista dei documenti o i singoli documenti Pdf, i dati vengono prima crittografati attraverso un algoritmo AES-256 e in fase di ricezione vengono decrittografati dal client.

#### Metodo del server per crittografare messaggi in AES-256

```
encriptMessage: function (message, key, iv) {  
    var encrypt = crypto.createCipheriv('aes-256-cbc', key, iv)  
    var encrypted = encrypt.update(message, 'utf8', 'base64')  
    encrypted += encrypt.final('base64')  
    return encrypted  
}
```

#### Metodo del client per decrittografare messaggi in AES-256

```
public static String decrypt(String encrypted) {  
    String aesKey = (String) Modello.getBean("securityKey");  
    String initVector = (String) Modello.getBean("initVector");  
    try {  
        IvParameterSpec iv = new IvParameterSpec(Base64.getDecoder().decode(initVector));  
        SecretKeySpec skeySpec = new SecretKeySpec(Base64.getDecoder().decode(aesKey), "AES");  
  
        Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5PADDING");  
        cipher.init(Cipher.DECRYPT_MODE, skeySpec, iv);  
  
        byte[] original =  
            cipher.doFinal(Base64.getDecoder().decode(encrypted));  
        return new String(original);  
    } catch (Exception ex) {  
        ex.printStackTrace();  
    }  
    return null;  
}
```

## 5.5 Controllo e permessi

Non tutti gli utenti sono uguali è quindi stato necessario implementare una soluzione capace di offrire diversi servizi in base ai permessi di uno specifico utente. In particolare, esistono 3 livelli, in base al livello l'utente ha più o meno privilegi.

- **Livello 1:** L'utente di livello ha accesso a tutti i documenti presenti nel database. Una volta scaricati i documenti non hanno nessuna scadenza.
- **Livello 2:** L'utente di livello 2 ha anche lui accesso a tutti i file disponibili ma una volta scaricati i documenti possono essere consultati per un massimo di 14 giorni.
- **Livello 3:** L'utente di livello 3 ha accesso solo ad un determinato tipo di documenti, ed una volta scaricati i possono essere consultati per un massimo di 7 giorni.

La seguente tabella riassume i diversi privilegi degli utenti:

Privilegi	Accesso ai documenti	Scadenza documenti
<b>Livello 1</b>	Tutti	Illimitata
<b>Livello 2</b>	Tutti	14 giorni
<b>Livello 3</b>	Documenti di tipo 'base'	7 giorni

Tabella 5.2: Permessi e privilegi utente

Come abbiamo visto paragrafo **3.2 Persistenza dei dati** di ogni utente si tiene traccia di *username*, *password* e *privileges*. Quest'ultimo valore infatti rappresenta il livello di permessi dell'utente. Mentre i documenti Pdf di tiene traccia del valore *type*.

Quando l'utente richiede la lista dei documenti, il server controlla il valore di *privileges* dell'utente, se il valore è uguale a '1' o '2' allora il server invierà come risposta la lista completa di documenti presenti nel DB, mentre se il valore *privileges* è uguale a '3' in questo caso invierà come risposta un lista di documenti che hanno come valore *type = base*.

Ragionamento simile quando l'utente vuole scaricare un documento. Il server esegue un controllo sul valore *privileges* dell'utente:

- Se *privileges* uguale a 1: al documento non viene impostata nessuna data di scadenza;

- Se *privileges* uguale a 2: al documento viene impostata una data di scadenza di 14 giorni;
- Se *privileges* uguale a 3: al documento viene impostata una data di scadenza di 7 giorni;

## 5.6 Protezione e sicurezza dei documenti

Per proteggere i documenti PDF una volta condivisi, per evitarne la divulgazione e l'accesso ad utenti non autorizzati sono state applicate misure di protezione affinché i documenti possano essere consultati solamente tramite l'applicazione client da me realizzata. Le misure adottate sono le seguenti:

- **Screenshot disabilitati:** Un volta aperto il documento tramite l'applicazione non è possibile la cattura dello schermo. Questo impedisce agli utenti di effettuare un screenshot al documento ed eventualmente condividerne l'immagine con utenti esterni o non autorizzati. Per disabilitare lo screenshot nell'applicazione, nel metodo *onCreate()* della MainActivity ho aggiunto semplicemente il seguente codice:

```
getWindow().setFlags(WindowManager.LayoutParams.FLAG_SECURE,
    WindowManager.LayoutParams.FLAG_SECURE);
```

Usando FLAG\_SECURE come flag del parametro di layout, il contenuto della finestra viene trattato come sicuro, impedendo che appaia negli screenshot o venga visualizzato su display non sicuri.

- **Password di apertura dei documenti:** I documenti PDF sono protetti da password, ma gli utenti non la conoscono. Fruttando la proprietà di un PDF è possibile limitare l'accesso a un documento impostando apposite password. Il server infatti prima di inviare un documento all'utente protegge il documento PDF con una password. La password viene generata con la funzione *random-Password* e applicata al documento con il metodo *qpdf.encrypt* della libreria **qpdf**.

Codice per generare una password

```
randomPassword: function(length){  
    charset = "!#$%&()*+,.-./:;<=>?@[\]^_{}~0123456789ABCDEFGHIJ  
    KLMNOPQRSTUVWXYZ!#$%&()*+,.-./:;<=>?@[\]^_{}~0123456789abcd  
    efg hij klmnopqrstuvwxyz",  
    password = ""  
    for (var i = 0, n = charset.length; i < length; i++) {  
        password += charset.charAt(Math.floor(Math.random() * n))  
    }  
    return password  
}
```

Alcuni esempi di password da 16 caratteri generate randomicamente

```
Wz-$Id;xSJ..&G.  
D)04CB[5ra:5ok:4  
}5Rh[]~‘:oUG}{(h!  
%%^J]s+b{k+fe8e,
```

Processo di codifica di un documento

```
const password = utility.randomPassword(16)  
const options = {  
    keyLength: 256,  
    password: password,  
    outputFile: encPath,  
}  
. . .  
await qpdf.encrypt(newPath, options)
```

Una volta bloccato da password, il file e la password stessa vengono inviati al client. In fase di ricezione, i file del documento vengono salvati nella memoria interna del telefono, mentre alcune informazioni relative al documento, come password e data di scadenza vengono salvate localmente nell'applicazione tramite le **SharedPreferences**, una modalità offerta da Android per il salvataggio di dati in maniera persistente, che di default rende le informazioni accessibili esclusivamente ai componenti dell'app in cui vengono memorizzate. L'utente quindi non viene mai a conoscenza della password e non c'è rischio che venga divulgata, poiché in fase di apertura del documento, il sistema andrà a recuperare la password SharedPreferences, il documento viene sbloccato dalla password e visualizzato direttamente sullo schermo. Questo è stato possibile grazie alla libreria **Android PdfViewer** che oltre a consentire la visualizzazione di documenti PDF su Android, consente di sbloccare eventualmente documenti bloccati da password.



# Capitolo 6

## Risultati sperimentali

Questo capitolo ha come obiettivo quello di mostrare il funzionamento ed il comportamento del sistema, sottoponendolo a vari test. Lo scopo dei test è verificare che l'applicazione soddisfi i requisiti funzionali specificati e garantire che tutte le funzionalità funzionino come previsto e correttamente su dispositivi reali. Quando si testa un'applicazione su dispositivi reali, si eseguono i test direttamente su dispositivi fisici che gli utenti utilizzano effettivamente, smartphone, tablet, computer portatili o desktop anziché su emulatori o simulatori.

### 6.1 Set-up sperimentale

Come detto in precedenza una volta che il server viene avviato, sarà in ascolto sul "*localhost*" all'indirizzo IP 127.0.0.1 e sulla porta specificata nel caso la porta 3000. Questo significa che il server sarà in grado di accettare richieste HTTP provenienti dallo stesso computer in cui viene eseguito. Per testare il progetto su un reale dispositivo mobile collegato ad una rete qualsiasi è necessario rendere pubblico il *localhost*. La soluzione: **Ngrok**<sup>1</sup>.

Ngrok è uno strumento che consente di esporre un server locale alla rete pubblica attraverso un "tunnel" sicuro. Questo significa che è possibile accedere a un server in esecuzione sul proprio computer attraverso una qualsiasi rete, anche se il server non è disponibile direttamente su internet e non ha un indirizzo IP pubblico. Una volta avviato ngrok bisogna specificare l'indirizzo su cui il server locale è in ascolto e ngrok assegnerà un URL pubblico (come "<https://abcdef123.ngrok-free.app>"). I dispositivi esterni possono quindi accedere al server tramite l'URL pubblico generato da ngrok e tutte le richieste saranno inoltrate sul nostro *localhost* dove è in ascolto il server. In questo modo è stato possibile testare il comportamento dell'applicazione su diversi dispositivi fisici.

Al seguente link è possibile vedere alcuni video e/o screenshot dei risultati ottenuti:

---

<sup>1</sup><https://ngrok.com/>

```

Prompt dei comandi - ngrok http 3000
ngrok by @inconshreveable
Session Status      online
Account            Mario (Plan: Free)
Version             2.3.41
Region              United States (us)
Web Interface      http://127.0.0.1:4040
Forwarding          http://[REDACTED].ngrok-free.app -> http://localhost:3000
Forwarding          https://[REDACTED].ngrok-free.app -> http://localhost:3000

Connections        ttl     opn     rt1     rt5     p50     p90
                   0       0     0.00    0.00    0.00    0.00

```

Figura 6.1: Esempio di avvio di Ngrok  
(Indirizzo pubblico generato è stato oscurato per motivi di sicurezza)

[https://drive.google.com/drive/folders/1yzrBIvh3hX\\_X0WFP10k0rg3v5G1c0Wd0?usp=sharing](https://drive.google.com/drive/folders/1yzrBIvh3hX_X0WFP10k0rg3v5G1c0Wd0?usp=sharing)

## 6.2 Test della protezione dei documenti

Una volta completato lo sviluppo dell'applicazione ho concentrato la mia attenzione in modo particolare su alcuni potenziali problemi riguardo la protezione dei documenti PDF una volta condivisi con l'utente. Arrivati a destinazione infatti, non si ha più il totale controllo sui documenti e le uniche certezza che abbiamo riguardo la sicurezza e il controllo sono affidate alla password di accesso e alla data di scadenza. La password ci assicura che il documento può essere aperto solo da chi la password la conosce e nel nostro caso specifico non c'è modo che l'utente venga a conoscenza di essa poiché il sistema gestisce in automatico la fase di apertura e sblocco. Mentre la scadenza dovrebbe assicurarci che oltre una certa data, il documento non sia più accessibile e consultabile all'interno dell'applicazione.

Partendo dalla data di scadenza, purtroppo, ho riscontrato un modo semplice che potrebbe trovare l'utente per aggirare questo vincolo. Come già detto in precedenza, il sistema è stato pensato per funzionare anche in modalità offline, perciò, in fase di verifica la scadenza dei documenti viene confrontata con la data del dispositivo. Quindi la domanda sorta spontanea è stata: che succede se il documento scaduto e si prova a "tornare indietro nel tempo"? Ovvero modificare la data del dispositi-

tivo e tornare ad una data precedente a quella di scadenza. Il tentativo ha dato solo conferma a ciò che si poteva facilmente intuire: il documento torna ad essere a disposizione dell'utente e non risulta scaduto. Una possibile soluzione a questo problema poteva essere quella di eliminare definitivamente il documento dal dispositivo una volta verificatane la scadenza. Se ad un primo approccio questa soluzione poteva sembrare valida, ragionandoci ho avuto subito modo di ricredermi. Anche in questo caso infatti la data di scadenza del documento viene confrontata con la data del dispositivo e in nessun modo ci si può assicurare che almeno una volta si verifica la scadenza, che porterebbe alla conseguente all'eliminazione del documento. Con qualche accortezza in più infatti basterebbe che l'utente faccia in modo di non superare mai la data indicata nella scadenza con il dispositivo. Quindi, in merito a quanto detto fin ora, il sistema è rimasto vulnerabile a questo tipo di problema poiché non ho trovato un metodo che in modalità offline garantisca che la scadenza venga rispettata prevenendo che l'utente possa provare ad ingannare il sistema cambiando la data.

Per quanto riguarda la password, come abbiamo già discusso in precedenza, per limitare l'accesso ad un documento il formato PDF implementa nativamente una funzionalità che consente di impostare una password di apertura o di accesso. Una volta scaricato, il documento risiederà nella memoria interna del dispositivo. Attraverso il *file manager* è possibile trovare la cartella dove risiedono tutti i documenti scaricati, anche se in questo caso la cartella è nascosta, all'utente basterebbe attivare l'impostazione *Mostra file nascosti* per visualizzare tutti i file e le cartelle. A questo punto tutti i file dei documenti sono visibili e posso essere spostati, copiati o manipolati dall'utente. Da questo momento in poi, ho provato ad aprire il documento con diverse applicazioni Android per la visualizzazione di PDF e tutti i tentativi hanno portato allo stesso risultato: non è possibile visualizzare il contenuto del PDF se non si conosce la password. Quindi sono chiesto se esistesse un modo per rimuovere la password dal documento. Cercando in rete è possibile trovare svariati software, perlopiù a pagamento, che promettono di ritrovare la password "smarrita" dei documenti tramite *attacchi di forza bruta*. Come abbiamo già visto nel paragrafo 3.6 del Capitolo 3 usando la funzione *randomPassword*, viene generata randomicamente una password di 16 caratteri che è la combinazione di caratteri alfanumerici, caratteri speciali e in più sono comprese sia lettere minuscole che maiuscole. In questo caso, essendo la password estremamente compressa, un attacco di forza bruta dovrebbe generare miliardi e miliardi di combinazioni per tentare di indovinare la password e questo potrebbe richiedere svariati anni. Secondo diversi siti, come ad esempio

*www.passwordmonster.com* mostrato in Figura 6.2, per tentare decifrare una password con le caratteristiche sopradescritte sono necessari "73 mila triliioni di anni". Insomma, in base a quanto detto finora abbiamo la certezza che un documento PDF protetto può essere aperto esclusivamente se si conosce la password, quest'ultima però dovrà essere abbastanza complessa per evitare che il documento possa essere vittima di un attacco di forza bruta.



Figura 6.2: Test su PasswordMonster

Un'ultima verifica per mettere alla prova il sistema, riguarda la possibilità che ha l'utente di effettuare uno screenshot ad un documento aperto. Anche questo aspetto è stato già chiarito precedentemente nel paragrafo 3.6 del Capitolo 3 e sappiamo con certezza infatti che mentre l'app è avviata sul dispositivo, il FLAG SECURE impedisce sia di effettuare screenshot che l'acquisizione video dell'display. Tuttavia è stato utile verificare come si comporta il sistema con la funzionalità di **screencast**. Lo screencast è una funzione presente nei dispositivi Android che consente di trasmettere in modo istantaneo lo schermo del dispositivo su una TV o un altri dispositivi di visualizzazione, ad esempio lo schermo di un PC. È stato constatato che anche in fase di condivisione, in FLAG SECURE tratta il contenuto della schermata come sicuro e ne impedisce la visualizzazione oscurando il contenuto della finestra. In questo modo utente non ha modo di visualizzare i documenti sullo schermo di un PC ed è privato della possibilità di usare lo *Strumento di cattura* per acquisire uno screenshot del documento. In conclusione, possiamo osservare che l'unica opzione che resta possibile all'utente per continuare a visualizzare il contenuto dei documenti al di fuori della applicazione è quella di fotografare il display che mostra la schermata con il documento aperto.



# Capitolo 7

## Conclusioni

In conclusione, questa tesi di laurea rappresenta un’opportunità per esplorare il tema della sicurezza dei documenti e per acquisire conoscenze e competenze tecniche fondamentali per la progettazione e lo sviluppo di soluzioni di sicurezza dei documenti altamente efficaci e affidabili. L’obiettivo di questa tesi mirava principalmente allo sviluppo e alla realizzazione di un’applicazione Android che permettesse agli utenti visualizzare, in modo esclusivo, i documenti in formato PDF sul dispositivo Android, ma allo stesso tempo offrire tutte le funzionalità tipiche di un buon sistema di gestione dei documenti digitali, in particolare la sicurezza. Prima dello sviluppo è stato fondamentale infatti analizzare quelli che sono i requisiti e le caratteristiche principali che un sistema deve garantire per una buona protezione per i documenti e inoltre approfondire la mia conoscenza riguardo sul formato PDF dei documenti. Mentre in fase di sviluppo ho avuto modo sia di approfondire le mie conoscenze riguardo al linguaggio Java, in questo caso legato allo sviluppo Android, sia modo apprendere nuove tecnologie, in particolare il linguaggio JavaScript, il runtime NodeJs ed il framework Express. In generale mi ritengo abbastanza soddisfatto del prodotto finale sia dal punto di vista applicativo ed in particolare dal punto di vista grafico, l’obiettivo era quello dare una propria identità al prodotto e di badare quindi anche all’estetica delle varie interfacce. Da punto di vista tecnico, l’aspetto più complicato ed interessante da realizzare durante lo sviluppo e l’implementazione è stato la parte relativa alla *riservatezza* e quindi lo studio e la ricerca dei vari meccanismi ed algoritmi per generare e scambiare le chiavi simmetriche e asimmetriche, e l’applicazione di questi meccanismi per cifrare e decifrare il contenuto dei messaggi al fine di realizzare la comunicazione sicura tra l’applicazione client e il server.

D’altra parte, come visto nel capitolo precedente, alcune soluzioni implementate possono essere migliorate e altre funzionalità aggiunte per rendere il sistema più completo e sicuro.

## 7.1 Sviluppi futuri

Per rendere l'applicazione completa, può essere aggiunta la procedura di registrazione al sistema e quindi dare la possibilità ad un nuovo utente di creare un account, applicando regole e misure rigide per evitare che uno stesso utente possa creare più account magari usando diversi username. Un'altra funzionalità interessante potrebbe essere quella di realizzare un nuovo livello di privilegio che consenta all'utente non solo di scaricare i documenti ma anche caricarli direttamente dal dispositivo mobile al server, cosicché che il nuovo documento sia disponibile per tutti gli altri utenti che si collegano. Mentre la funzionalità di visualizzazione offline può essere migliorata per i motivi già ampiamente discussi nel capitolo precedente, una possibile soluzione potrebbe essere quella di realizzare un sistema che permetta di visualizzare i documenti offline un numero limitato di volte oppure un numero limitato di giorni e al termine dovrà essere necessario tornare online per sbloccare il sistema(ma si tratterebbe ancora di date di scadenze!). Ed infine potrebbe essere utile aggiungere filigrane a documenti importanti per garantire l'integrità dei contenuti.

### Codice sorgente

Il codice sorgente dell'intero progetto, è stato pubblicato ed è accessibile a tutti su GitHub al link:

- <https://github.com/mario-rosa/SafeDocViewer> .

# Bibliografia