

Analisi dei requisiti

Gruppo 02

December 8, 2024

1 Introduzione

Il programma implementa una rubrica per gestire i contatti telefonici e/o e-mail. Il programma consente agli utenti registrati di inserire, eliminare o modificare dei contatti all'interno della rubrica e permette ai nuovi utenti di registrarsi. A ogni contatto sono associate almeno le seguenti informazioni:

- Un nome e un cognome; una delle due informazioni può essere vuota (ma non entrambe)
- Da zero a tre numeri di telefono
- Da zero a tre indirizzi e-mail

2 Requisiti

2.1 Requisiti funzionali

- **Funzionalità individuali:**

IF-1 Il sistema deve permettere agli utenti non registrati di registrarsi inserendo una e-mail e una password.

IF-2 Il sistema deve permettere agli utenti registrati di accedere tramite e-mail e password. Dopo l'accesso vengono visualizzati i contatti salvati in precedenza.

IF-3 Il sistema deve permettere agli utenti registrati di inserire un nuovo contatto.

IF-4 Il sistema notifica con un messaggio se l'utente tenta di aggiungere un contatto duplicato. Due contatti sono uguali se hanno lo stesso nome e cognome.

IF-5 Il sistema deve permettere agli utenti registrati di eliminare un contatto selezionato.

IF-6 Il sistema deve permettere agli utenti di modificare i campi di un contatto selezionato.

IF-7 Il sistema deve permettere di ricercare un contatto tramite una sottostringa inserita in una barra di ricerca.

IF-8 Il sistema consente di aggiungere tag personalizzabili ai contatti.

IF-9 Il sistema consente di filtrare i contatti tramite la selezione di un tag.
IF-10 Il sistema deve permettere di ordinare le righe della tabella in base a nome o cognome e con un ordinamento che può essere crescente o decrescente.
IF-11 Una volta selezionato un criterio di ordinamento, questo viene memorizzato e viene applicato in automatico alle successive sessioni.
IF-12 Il sistema deve permettere agli utenti registrati di importare i contatti da un file o esportare i contatti in un file.

• **Esigenze di Dati e informazioni:**

DF-1 Dati del profilo utente:

- e-mail
- password

DF-2 Dati dei contatti:

- nome e cognome (almeno uno dei due campi deve essere non vuoto)
- da zero a tre numeri di telefono
- da zero a tre indirizzi e-mail
- zero o più tag

Un tag è un ulteriore attributo il cui valore può essere definito dall'utente durante l'aggiunta o la modifica di un contatto.

• **Interfaccia utente:**

UI-1 Pagina di login dove inserire l'e-mail e la password per il login.
UI-2 Pagina di Registrazione dove inserire l'e-mail e la password (e conferma password) per la registrazione.
UI-3 La rubrica viene visualizzata in forma tabellare. Ogni colonna della tabella corrisponde a un campo del generico contatto. Ogni riga della tabella corrisponde a un singolo contatto. I campi relativi al numero di telefono e all'indirizzo e-mail possono ospitare da zero a tre valori.
UI-4 La selezione di un contatto avviene cliccando la corrispondente riga con il tasto sinistro del mouse.
UI-5 Nella pagina principale della rubrica sono presenti diversi bottoni e menu per eseguire le operazioni.
UI-6 Messaggi di errore: in caso di errore nelle informazioni inserite dall'utente l'interfaccia deve indicare quale informazione è errata ed eventuali vincoli non rispettati.
UI-7 Se l'utente prova a inserire un contatto già esistente compare un messaggio di notifica.

• **Interfacce con sistemi esterni:**

IS-1 Database per salvare le informazioni.

3 Vincoli di progettazione

1. Il sistema deve essere implementato tramite un'applicazione Java compatibile con Maven.
2. La consegna del progetto è prevista per il 15/12/2024.

4 Casi d'uso

4.1 Attori

- Utente registrato: può effettuare il login; una volta effettuato il login, visualizza la pagina principale della rubrica contenente i contatti salvati.
- Utente: può registrarsi inserendo i dati richiesti.
- Database: gestisce il servizio di autenticazione e di memorizzazione dei contatti.

4.2 Descrizione dei casi d'uso

REGISTRAZIONE

Attori partecipanti: utente, database

Precondizioni:

- L'utente apre l'applicazione

Postcondizioni:

- L'utente ha un account e diventa un utente registrato

Flusso di eventi:

1. L'utente inserisce e-mail e password (e conferma password)
2. Verifica della correttezza dei dati inseriti
3. I dati vengono inseriti nel database
4. Registrazione completata

Flusso di eventi alternativo:

- 3.a L'indirizzo e-mail è già presente nel database. L'utente deve effettuare il login.
- 3.b La password inserita non rispetta le condizioni di sicurezza consigliate; si richiede l'inserimento di una password differente.
- 3.c Le password non corrispondono.

LOGIN

Attori partecipanti: utente registrato, database

Precondizioni:

- L'utente deve essere registrato

Postcondizioni:

- L'utente ha accesso alla homepage della rubrica

Flusso di eventi:

1. L'utente registrato inserisce e-mail e password.
2. Verifica delle credenziali (nel database).
3. Verifica andata a buon fine.

Flusso di eventi alternativo:

- 2.a E-mail non presente nel database → Registrazione.
- 3.a Credenziali errate

LOGOUT

Attori partecipanti: utente registrato

Precondizioni:

- L'utente ha effettuato il login

Postcondizioni:

- L'utente torna alla schermata iniziale

Flusso di eventi:

1. L'utente preme sul bottone per uscire dall'account.

OPERAZIONI RUBRICA

Attori partecipanti: utente registrato

Precondizioni:

- L'utente ha eseguito il login

Postcondizioni: Flusso di eventi:

1. L'utente visualizza la pagina principale della rubrica e sceglie quale operazione effettuare

Flusso di eventi alternativo:

- 1.a L'utente seleziona un contatto e l'operazione da effettuare su di esso.

AGGIUNGERE CONTATTO

Attori partecipanti: utente registrato

Precondizioni:

- L'utente preme sul bottone per l'aggiunta di un contatto

Postcondizioni:

- Un contatto viene aggiunto alla rubrica

Flusso di eventi:

1. L'utente specifica le informazioni da associare al nuovo contatto.
2. Le modifiche vengono salvate.

Flusso di eventi alternativo:

- 2.a I campi nome e cognome sono entrambi vuoti; il contatto non viene inserito in rubrica.
- 2.b Contatto duplicato.
- 2.c L'utente annulla l'operazione.

RIMUOVERE CONTATTO

Attori partecipanti: utente registrato

Precondizioni:

- Nella rubrica è presente almeno un contatto.
- L'utente indica che vuole rimuovere un contatto tramite l'apposito bottone.

Postcondizioni:

- Il contatto selezionato è eliminato dalla rubrica

Flusso di eventi:

1. Il contatto viene rimosso dalla rubrica.
2. Le modifiche vengono salvate.

Flusso di eventi alternativo:

- 2.a L'utente annulla l'operazione.

MODIFICARE INFORMAZIONI PER UN CONTATTO GIÀ ESISTENTE

Attori partecipanti: utente registrato

Precondizioni:

- Nella rubrica è presente almeno un contatto
- L'utente indica di voler modificare un contatto tramite l'apposito bottone

Postcondizioni:

- Le informazioni del contatto selezionato sono modificate

Flusso di eventi:

1. L'utente modifica le informazioni del contatto selezionato.
2. Le modifiche vengono salvate.

Flusso di eventi alternativo:

- 2.a Non è possibile salvare la modifica effettuata perché sono stati eliminati sia il nome che il cognome.
- 2.b Contatto duplicato.
- 2.c L'utente annulla l'operazione.

RICERCA CONTATTI

Attori partecipanti: utente registrato

Precondizioni:

- L'utente seleziona la barra di ricerca

Postcondizioni:

- L'utente visualizza solo i contatti corrispondenti.

Flusso di eventi:

1. L'utente inserisce la sottostringa da ricercare

IMPORTARE CONTATTI

Attori partecipanti: utente registrato

Precondizioni:

- L'utente preme il tasto per importare i contatti.

Postcondizioni:

- I contatti da importare sono aggiunti alla rubrica.

Flusso di eventi:

1. L'utente seleziona il file contenente i contatti da importare.
2. I contatti vengono aggiunti.

Flusso di eventi alternativo:

- 2.a Errore di importazione: messaggio d'errore.
- 2.c L'utente annulla l'operazione.

ESPORTARE CONTATTI

Attori partecipanti: utente registrato

Precondizioni:

- L'utente preme il tasto per esportare i contatti.

Postcondizioni:

- I contatti presenti in rubrica vengono esportati nel file indicato.

Flusso di eventi:

1. L'utente seleziona la directory in cui desidera salvare il file.
2. I contatti vengono esportati.

Flusso di eventi alternativo:

- 2.a Errore di esportazione: messaggio d'errore.
- 2.c L'utente annulla l'operazione.

ORDINARE CONTATTI

Attori partecipanti: utente registrato

Precondizioni:

- L'utente indica un criterio di ordinamento cliccando su uno dei campi della tabella

Postcondizioni:

- I contatti della rubrica sono visualizzati secondo il criterio scelto indicato

Flusso di eventi:

1. L'ordinamento è stato effettuato correttamente.
Nota: I contatti che non presentano il campo di ordinamento scelto vengono posizionati in fondo alla tabella e ordinati in base al valore dell'altro campo presente.

FILTRARE CONTATTI

Attori partecipanti: utente registrato

Precondizioni:

- L'utente preme sul bottone per filtrare

Postcondizioni:

- Sono visualizzati solo i contatti associati al/ai tag selezionato/i

Flusso di eventi:

1. L'utente seleziona i tag.

CONTATTO DUPLICATO

Attori partecipanti: utente registrato

Precondizioni:

- Durante un'operazione di aggiunta/modifica c'è un contatto duplicato.

Postcondizioni:

- La rubrica viene aggiornata.

Flusso di eventi:

1. Compare un messaggio di scelta.
2. Viene aggiunto un nuovo contatto.

Flusso di eventi alternativo:

- 2.a Il contatto esistente viene modificato.

SALVATAGGIO

Attori partecipanti: utente registrato, database

Precondizioni:

- L'utente indica di voler salvare una modifica.

Postcondizioni:

- Le modifiche vengono salvate.

Flusso di eventi:

1. La rubrica aggiornata viene salvata.

5 Descrizione dei requisiti

ID	Stato del requisito	Priorità	Rischio Tecnico
IF-1	Accettato per questa release	Alta	Medio
IF-2	Accettato per questa release	Alta	Basso
IF-3	Accettato per questa release	Alta	Basso
IF-4	Accettato per questa release	Bassa	Basso
IF-5	Accettato per questa release	Alta	Basso
IF-6	Accettato per questa release	Media	Basso
IF-7	Accettato per questa release	Media	Basso
IF-8	Rimandato a release successive	Bassa	Basso
IF-9	Rimandato a release successive	Bassa	Basso
IF-10	Accettato per questa release	Media	Basso
IF-11	Accettato per questa release	Basso	Basso
IF-12	Rimandato a release successive	Medio	Medio
DF-1	Accettato per questa release	Alta	Basso
DF-2	Accettato per questa release	Alta	Basso
UI-1	Accettato per questa release	Alta	Basso
UI-2	Accettato per questa release	Alta	Basso
UI-3	Accettato per questa release	Alta	Basso
UI-4	Accettato per questa release	Alta	Basso
UI-5	Accettato per questa release	Alta	Basso
UI-6	Accettato per questa release	Media	Basso
UI-7	Accettato per questa release	Bassa	Basso
IS-1	Rimandato a release successive	Bassa	Alto

Tabella dei requisiti

6 Diagramma dei casi d'uso

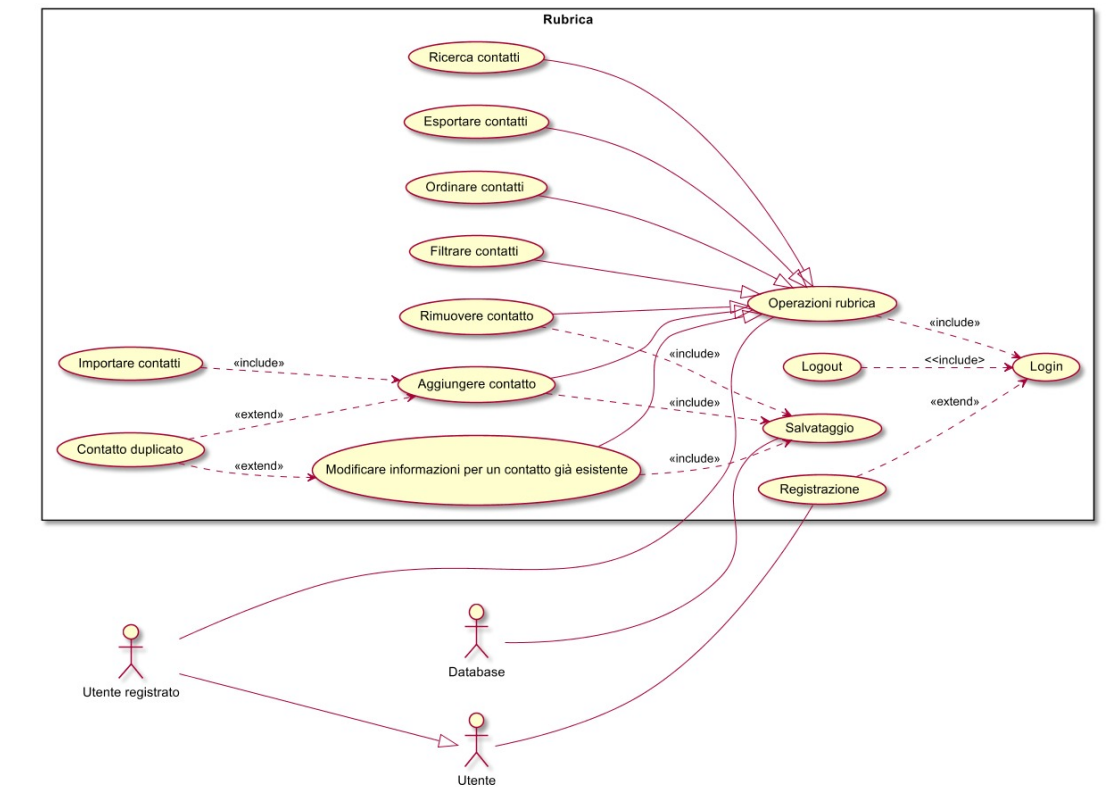


Figure 1: Diagramma UML

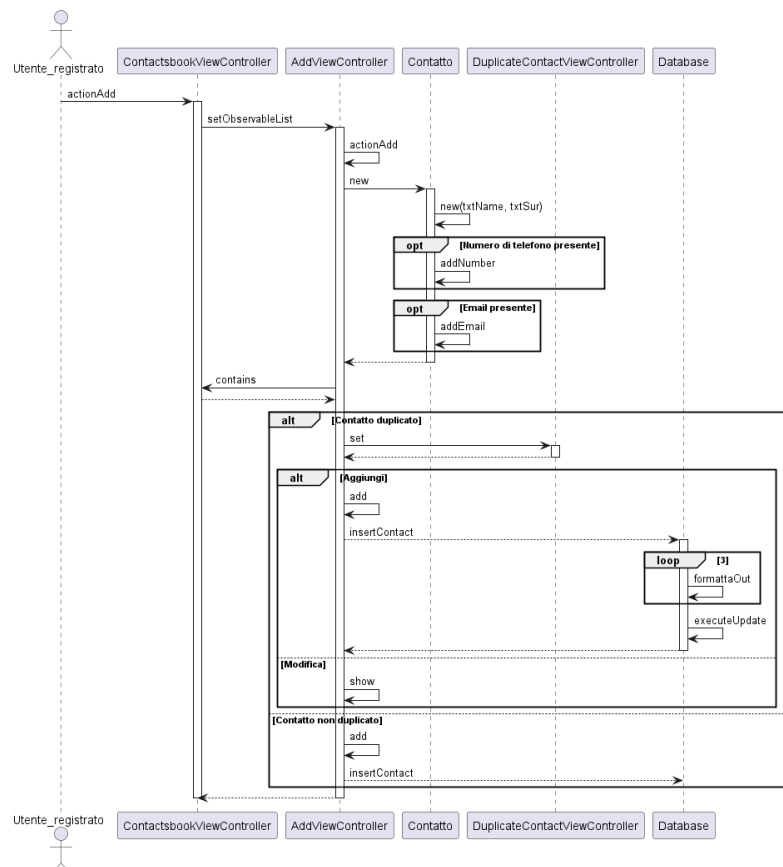


Figure 5: Diagramma delle sequenze relativo al login.

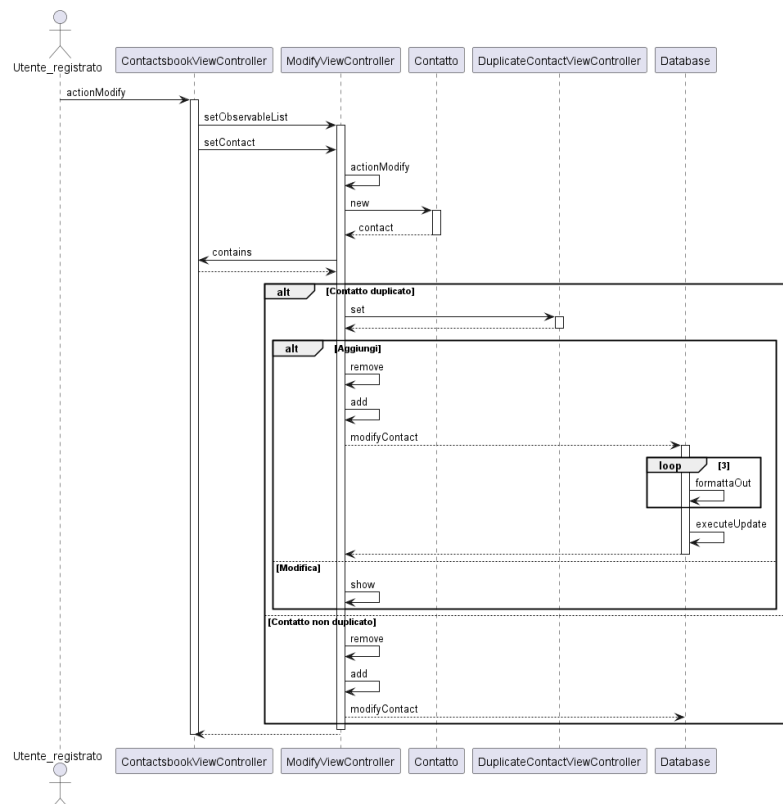


Figure 6: Diagramma delle sequenze relativo alla modifica di un contatto già esistente.

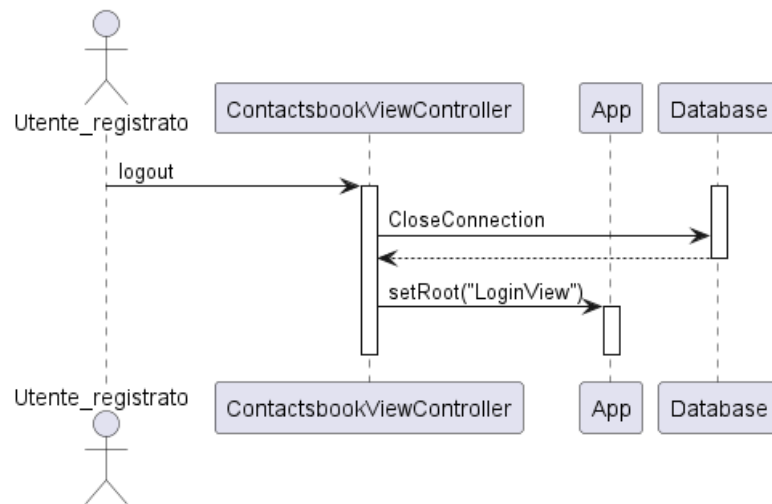


Figure 8: Diagramma delle sequenze relativo al logout.

9 Analisi di coesione e accoppiamento

Di seguito sono presenti le tabelle che analizzano coesione e accoppiamento di varie classi.

CLASSE	COESIONE
AddViewController	Funzionale
MOTIVAZIONE	
<p>La classe è responsabile di un singolo compito, ovvero gestire le interazioni dell'utente nella schermata per aggiungere un nuovo contatto.</p> <p>Tutti i metodi e i campi sono strettamente correlati a questo obiettivo:</p> <ul style="list-style-type: none"> I metodi come <code>actionAdd</code> e <code>actionCancel</code> implementano le azioni dell'utente. Gli elementi FXML definiti (ad esempio, <code>txtName</code>, <code>btnAdd</code>, <code>btnCanc</code>) sono usati per interagire con i dati immessi dall'utente. 	

CLASSE	TIPO DI COESIONE
LoginViewController	Funzionale
GIUSTIFICAZIONE	
<p>La classe è responsabile di un singolo compito, ovvero gestire la logica relativa alla pagina di login e registrazione, incluse validazione dei campi, interazioni con il database e gestione dell'interfaccia utente. Ogni metodo è strettamente correlato a questo obiettivo:</p> <ul style="list-style-type: none"> <code>Initialize</code>: configura la schermata e i binding per la disabilitazione dei pulsanti <code>txtSignMailInitialize</code>, <code>txtSignPassInitialize</code>, <code>txtConfirmPassInitialize</code>: gestiscono la validazione e gli aggiornamenti dei campi visivi <code>actionLogin</code>, <code>actionSignin</code>: implementano operazioni specifiche per login e registrazione 	

CLASSE	TIPO DI COESIONE
DuplicateContactViewController	Funzionale
GIUSTIFICAZIONE	
<p>La classe ha il compito specifico di mostrare all'utente un'interfaccia per decidere come gestire un contatto duplicato (aggiungere o modificare) e agire in base alla scelta</p> <p>I metodi implementati rispecchiano questo obiettivo:</p> <ul style="list-style-type: none"> <code>actionYes</code>: Gestisce il caso in cui l'utente sceglie di aggiungere il contatto. <code>actionNo</code>: Gestisce il caso in cui l'utente sceglie di non aggiungere il contatto, annullando dunque l'operazione e chiudendo la finestra. <code>Set</code>: Fornisce il contatto su cui verranno applicate le decisioni dell'utente. 	

CLASSE	TIPO DI COESIONE
ModifyViewController	Funzionale
GIUSTIFICAZIONE	
<p>La classe è responsabile di gestire la schermata di modifica di un contatto.</p> <p>Nello specifico:</p> <ul style="list-style-type: none"> Modifica delle informazioni del contatto. Aggiornamento della lista osservabile. Gestione dei contatti duplicati. 	

CLASSE	TIPO DI COESIONE
ContactsbookViewController	Comunicazionale
GIUSTIFICAZIONE	
<p>La classe gestisce la logica dell'interfaccia utente della rubrica, coordinando operazioni come l'aggiunta, la modifica, l'eliminazione e la visualizzazione dei contatti, la gestione della ricerca e dei filtri, e l'importazione/esportazione dei dati.</p>	

CLASSE	TIPO DI COESIONE
Contact	Funzionale
GIUSTIFICAZIONE	
<p>La classe Contact è focalizzata sul rappresentare un contatto con numeri di telefono, email, tag e un identificatore univoco (ID). Le operazioni che svolge sono tutte correlate a questa funzione centrale, come l'aggiunta di numeri di telefono, email e tag.</p> <p>I metodi come addNumber, addEmail, addTag e i rispettivi getter/setter lavorano direttamente sugli attributi della classe (number, email, tag), che sono tutti correlati alle informazioni di un contatto. Questo indica una buona coesione, dove tutti i metodi sono legati a una singola responsabilità.</p>	

CLASSE	TIPO DI COESIONE
Person	Funzionale
GIUSTIFICAZIONE	
<p>La classe si occupa di gestire le informazioni base di una persona, ovvero il nome e il cognome, e fornisce metodi per accedere e modificare questi dati. Le operazioni implementate, come getName, getSurname, setName, e setSurname, sono tutte coerenti con questo obiettivo di gestione di informazioni personali di base.</p>	

CLASSE	TIPO DI COESIONE
User	Funzionale
GIUSTIFICAZIONE	
<p>La classe User gestisce le informazioni di un utente, con un focus principale sulla password e sull'email. Le operazioni fornite sono coerenti con questo scopo, come setPassword per modificare la password e getEmail per ottenere l'email.</p> <p>User estende Person per ereditare la funzionalità di base per la gestione di un nome e un cognome, aggiungendo funzionalità specifiche per gli utenti (password ed email). Il metodo getRole restituisce il tipo di classe dell'utente, implementato come una stringa che riflette il ruolo specifico.</p>	

CLASSE	TIPO DI COESIONE
App	Funzionale
GIUSTIFICAZIONE	
<p>La classe si occupa di configurare la scena principale, di caricare il file FXML per la vista di login, e di impostare il titolo e l'icona dell'applicazione. Queste azioni sono tutte necessarie per l'avvio dell'applicazione, quindi sono coerenti con il suo ruolo di gestione dell'interfaccia utente.</p>	

CLASSI INTERESSATE	TIPO DI ACCOPPIAMENTO
LoginViewController, ContactsbookViewController	Controllo
GIUSTIFICAZIONE	
<ul style="list-style-type: none"> Nel caso di un accesso al database, ContactsbookController deve richiedere le informazioni al database nel caso di un accesso locale, ContactsbookController deve richiedere le informazioni al file di memoria locale <p>Dunque, in base al tipo di accesso che viene effettuato verranno effettuate operazioni diverse</p>	

CLASSI INTERESSATE	TIPO DI ACCOPPIAMENTO
ContactsbookViewController, AddViewController e ModifyViewController	Timbro
GIUSTIFICAZIONE	
<p>La classe ContactsbookViewController passa alle classi AddViewController e ModifyViewController la struttura su cui queste ultime devono effettuare delle operazioni</p>	

CLASSI INTERESSATE	TIPO DI ACCOPPIAMENTO
DuplicateContactViewController, AddViewController, ModifyViewController	Controllo
GIUSTIFICAZIONE	
DuplicateContactViewController può annullare o meno le operazioni di aggiunta e modifica del contatto avvenute per mezzo di AddViewController o ModifyViewController	