Università Politecnica delle Marche
Facoltà di Ingegneria
Dipartimento di Ingegneria dell'Informazione
Corso di Laurea in Ingegneria Informatica e dell'Automazione

Tesi di Laurea

Progettazione e implementazione di un software gestionale a supporto delle imprese

Design and implementation of management software to support enterprises

Relatore

Prof. Domenico Ursino

Candidato

Razvan Alexandru Dediu

Anno Accademico 2020-2021

Negli ultimi 33 anni, mi sono guardato ogni mattina allo specchio chiedendomi:
"Se oggi fosse l'ultimo giorno della mia vita, vorrei fare quello che sto per fare oggi?".
E ogni qualvolta la risposta è no per troppi giorni di fila,
capisco che c'è qualcosa che deve essere cambiato.

Steve Jobs, "Discorso all'Università di Stanford"

Indice

1 Introduzione

La tesi si pone l'obiettivo di illustrare la progettazione e l'implentazione di un software realizzato sia per desktop che per dispositivi mobili.In particolare, si tratta di un programma gestionale per i professionisti che operano nel settore edilizio.
Un software gestionale è un programma che automatizza i processi di gestione delle aziende semplificandone l'utilizzo. Il gestionale sviluppato ha lo scopo di facilitare le attività tipiche delle aziende che operano nel settore delle costruzioni edili rendendo immediatamente disponibili informazioni sull'azienda o sull'area amministrativa.
Le attività delle imprese che sono state prese in considerazione sono, principalmente, la gestione dei documenti, la gestione dei dipendenti e l'organizzazione del magazzino dell'azienda.
Le categorie sono state scelte con l'idea di semplifacare il controllo delle aree più importanti.Per questo motivo il software non viene considerato completo, ma pone le basi per lo sviluppo di un programma efficiente ed affidabile.

2 Analisi del sistema

2.1 Funzionalità

Il programma è composto da cinque macrofunzionalità: *Gestione dei documenti, Gestione dei dipendenti, Gestione del magazzino,*
Gestione obiettivo e Statistiche. Nella prima l'amministratore (o il datore di lavoro) ha la possibilità di salvare documenti nel database caricandone i dati o inserendo direttamente il documento intero tramite un apposito pulsante.I documenti sono classificati in tre categorie differenti in modo da rendere facile il loro ritrovamento.
Per la gestione dei dipendenti le funzionalità sono la visualizzazione dei dati di un determinato dipendente e la visualizzazione dei lavori in corso,degli orari di lavoro e dell'avanzamento lavori (con descrizione inserita dal dipendente).
Con l'ultima macrofunzionalità si ha la possibilità di visualizzare, cancellare ed inserire (CRD) materiali nel magazzino dell'impresa considerata.
Inoltre è possibile effetturuare una ricerca tramite una apposita barra per velocizzare il ritrovamento del materiale. Tra i dati inseriti per i prodotti (o materiali) è presente anche la disponibilità che sarà aggiornata dopo ogni movimento all'interno del magazzino. Per la gestione dell'obiettivo l'amministratore potrà inserire un obiettivo giornaliero che apparirà nelle dashboard dei dipendenti. Nella sezione delle statistiche sarà possibile visualizzare l'utile mensile ed una statistica sulle ore di lavoro dei dipendenti.

2.2 Diagramma dei casi d'uso

Con il seguente diagramma si vanno a rappresentare i possibili casi d'uso del software gestionale.

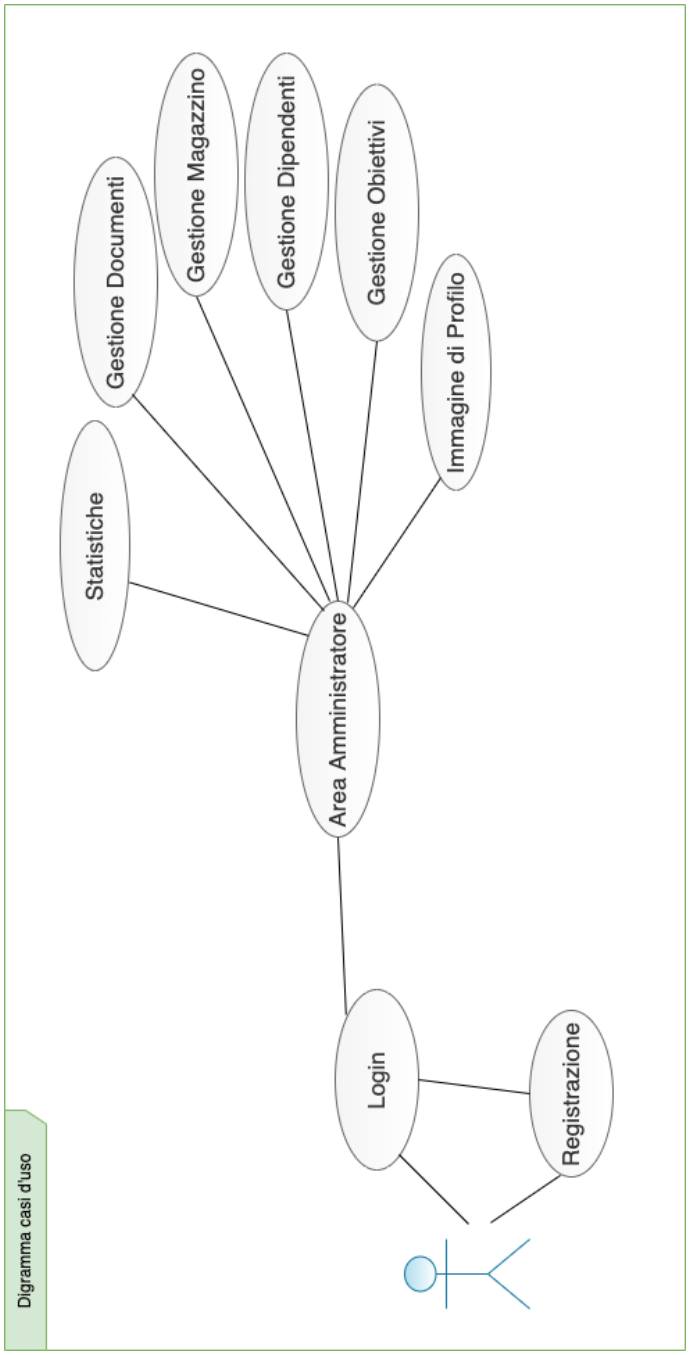


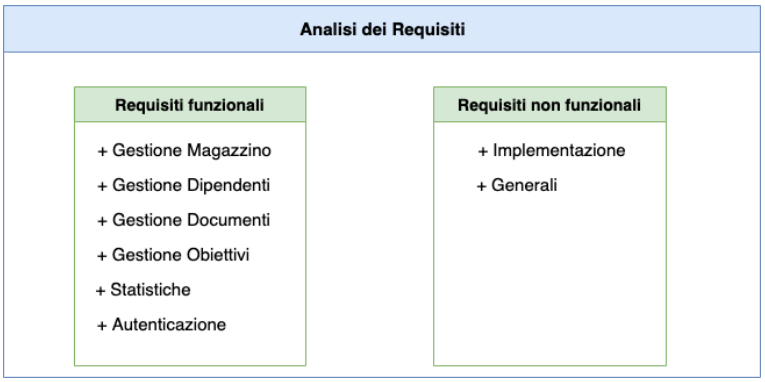
Diagramma dei casi d'uso

3 Sviluppo Software Desktop

3.1 Linguaggio di Programmazione

Per lo sviluppo del software si è utilizzato il linguaggio di programmazione Python. Python è uno dei linguaggi più utilizzati al mondo ed ha tra i principali obiettivi: dinamicità, semplicità e flessibilità. Oltre ad essere un linguaggio di "alto livello", supporta diversi tipi di paradigma come la metaprogrammazione ed i paradigmi procedurale, funzionale, a oggetti e lo scripting.
Le caratteristiche più immediatamente riconoscibili in Python sono le variabili non tipizzate e l'uso dell'indentazione per la sintassi delle specifiche, al posto delle più comuni parentesi.

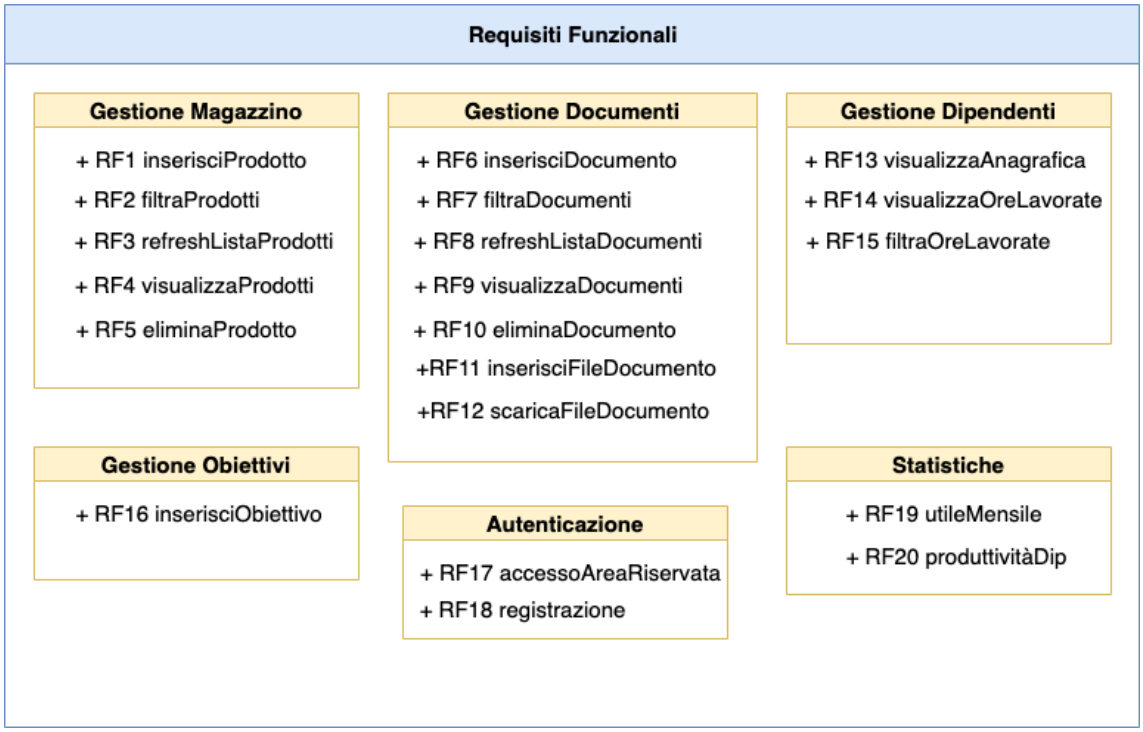
3.2 Analisi dei Requisiti



Analisi Dei Requisiti

3.2.1 Requisiti Funzionali

Dopo una breve introduzione al linguaggio utilizzato, si descrivono di seguito gli aspetti funzionali del software.



Requisiti Funzionali

	Requisiti	Descrizione
RF1	inserisciProdotto	il sistema dovrà gestire l'inserimento di un prodotto nel magazzino
RF2	filtraProdotto	il sistema dovrà gestire la ricerca di uno o più prodotti. In particolare dovrà filtrare i prodotti per nome
RF3	eliminaProdotto	il sistema dovrà gestire l'eliminazione di un prodotto selezionato
RF4	visualizzaProdotti	il sistema dovrà gestire la visualizzazione dei prodotti presenti nel magazzino
RF5	refreshListaProdotti	il sistema dovrà gestire l'aggiornamento della pagina visualizzata senza interrompere l'esecuzione del programma
RF6	inserisciDocumento	il sistema dovrà gestire l'inserimento degli estremi di un documento
RF7	filtraDocumento	il sistema dovrà gestire la ricerca di uno o più documenti. In particolare dovrà filtrare i documenti per nome
RF8	refreshListaDocumenti	il sistema dovrà gestire l'aggiornamento della pagina visualizzata senza interrompere l'esecuzione del programma
RF9	visualizzaDocumenti	il sistema dovrà gestire la visualizzazione dei documenti presenti nell'archivio
RF10	eliminaDocumento	il sistema dovrà gestire l'eliminazione di un documento selezionato
RF11	inserisciFileDocumento	il sistema dovrà gestire l'inserimento di un file che rappresenta il documento fisico nell'archivio
RF12	scaricaFileDocumento	il sistema dovrà gestire il download del file che rappresenta il documento fisico
RF13	visualizzaAnagrafica	il sistema dovrà gestire la visualizzazione dell'anagrafica di un dipendente selezionato
RF14	visualizzaOreLavorate	il sistema dovrà gestire la visualizzazione delle ore lavorate e dei dati associati
RF15	filtraOreLavorate	il sistema dovrà gestire la ricerca dello storico delle ore lavorate. In particolare, dovrà filtrare i dati per nome dipendente
RF16	inserisciObiettivo	il sistema dovrà gestire l'inserimento di un obiettivo e dei dati associati
RF17	accessoAreaRiservata	il sistema dovrà gestire l'accesso degli utenti all'area riservata
RF18	registrazione	il sistema dovrà gestire la registrazione di un nuovo utente
RF19	utileMensile	il sistema dovrà gestire la visualizzazione delle entrate, delle uscite e l'utile mensile
RF20	produttivitàDip	il sistema dovrà gestire la visualizzazione dei dipendenti con più ore lavorate

Descrizione dei Requisiti Funzionali

La schermata iniziale è la pagina di Login, dalla quale l'utente può decidere se:

- effettuare il Login: l'utente inserisce email e password per accedere alla propria area riservata
- effettuare la Registrazione : l'utente inserisce email, password ed alcuni dati personali in una form ,e se la registrazione avviene con successo, l'utente potrà effettuare l'accesso descritto nel punto precedente.

Dopo aver effettuato l'autenticazione, l'utente accede alla propria area riservata. All'interno della propria dashboard sono presenti sei funzionalità:

- *Immagine di profilo*: l'utente può caricare una immagine di profilo, la quale verrà salvata ed inserita nell' apposito widget. L'immagine sarà visualizzabile accanto al messaggio di benvenuto personalizzato.
- *Gestione Magazzino*: nella prima schermata dopo l'accesso, l'utente può:
  - Visualizzare i dati principali dei prodotti presenti nel magazzino
  - Effetturare una ricerca con la quale saranno stampati sul video i prodotti che contengono, nel proprio nome, la stringa di ricerca inserita nel widget di input dall'utente.
  - Aggiungere un prodotto all'interno del magazzino, inserendo i dati richiesti.
  - Effettuare un Refresh della pagina, in caso di modifica dei prodotti del magazzino.
- *Gestione Documenti*: tramite il menù è possibile spostarsi sulla schermata in cui vengono gestiti i documenti. In particolare è possibile:
  - Visualizzare i dati essenziali dei documenti registrati presenti nell'archivio
  - Effetturare una ricerca con la quale saranno selezionati i documenti che contengono, nel proprio nome, la stringa di ricerca inserita nel widget di input dall'utente.
  - Aggiungere un documento nell'archivio, in particolare:

    * inserendo i dati del documento;
    * inserendo il documento stesso in formato PDF.
  - Scaricare un documento in formato PDF.
  - Effettuare un Refresh della pagina, in caso di modifica dei documenti nell'archivio.

- *Gestione Dipendenti*: questa sezione è utilizzata per facilitare l'accesso ai dati dei dipendenti; l'utente può accedere:
  - ai Dati Anagrafici dei dipendenti;
  - alle Ore lavorate dei dipendenti e al lavoro svolto in una determinata data.

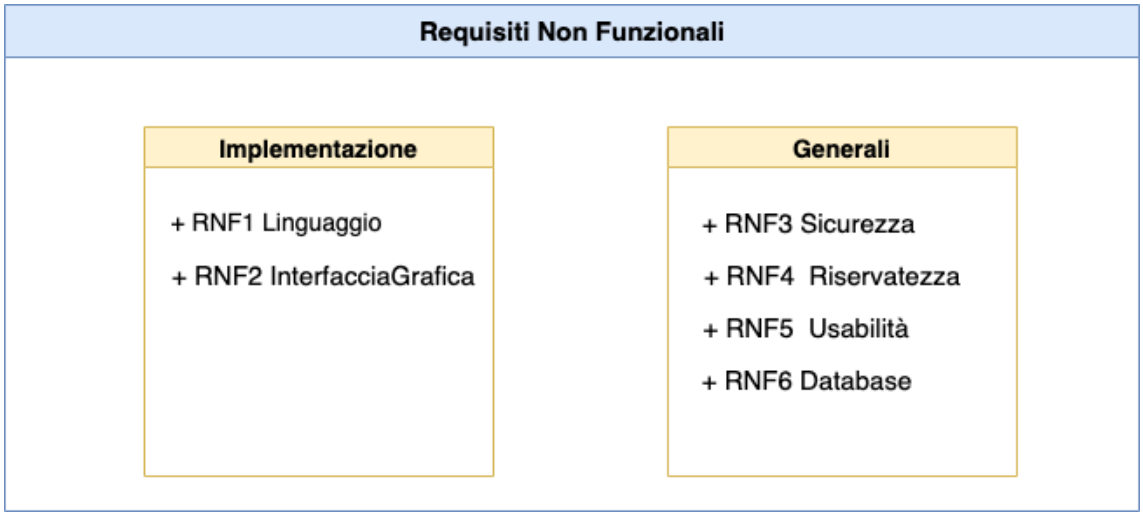
  Inoltre, è possibile filtrare i dati per nome del dipendente, in modo da facilitare la ricerca.

- *Gestione Obiettivi*: questa funzionalità è stata implementata per inoltrare ai dipendenti gli obiettivi giornalieri richiesti dal datore di lavoro. La schermata è composta da due widget che offrono la possibilità di inserire una data e una descrizione da comunicare ai dipendenti.

- *Statistiche*: questa sezione è utilizzata per effetturuare statistiche sui dati; in particolare:
  - *Profitto Mensile*: ottenuto come differenze fra entrate e uscite mensili.
  - *Ore Lavorate*: ottenute sommando le ore lavorate di ogni dipendente in un mese.

3.2.2 Requisiti Non Funzionali

I Requisiti descritti in questo paragrafo evidenziano quelle caratteristiche del software che descrivono come il sistema opera per svolgere determinati compiti.



Requisiti Non Funzionali

	Requisiti	Descrizione
RNF1	Linguaggio	il sistema dovrà essere realizzato con il linguaggio di programmazione Python3
RNF2	InterfacciaGrafica	il sistema dovrà essere dotato di interfaccia grafica
RNF3	Sicurezza	il sistema dovrà funzionare senza recare danno a persone o cose
RNF4	Riservatezza	il sistema dovrà tutelare le informazioni in esso contenute
RNF5	Usabilità	il sistema dovrà essere intuitivo e facile da usare anche per utente poco esperti
RNF6	Database	il sistema dovrà essere dotato di database per la persistenza dei dati

Descrizione dei Requisiti Non Funzionali

L'utilizzo di un database è stato fondamentale per la persistenza, la visualizzazione e l'elaborazione dei dati.
In particolare, è stato necessario per la visualizzazione e l'aggiornamento dei prodotti e dei documenti, ma anche per l'inserimento dell'immagine di profilo dell'utente e per gli obiettivi.
Il tipo di database utilizzato e i software ad esso correlati saranno descritti approfonditamente in seguito, in quanto la spiegazione del loro funzionamento ed utilizzo è necessaria per la comprensione del software.

3.2.3 Architettura

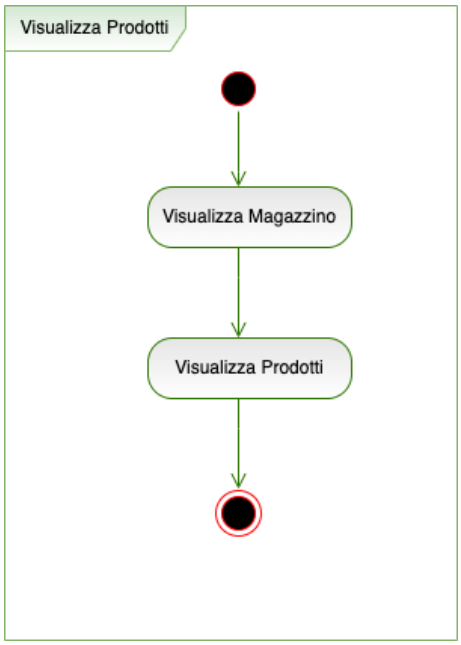
Il software è stato progettato secondo una architettura modulare. L'intero progetto può essere suddiviso in cinque macro-directory; in particolare:

- *BE*: in questa directory sono presenti tutti i file con estensione .py contenenti il codice sorgente del software; le classi che compongono il codice sono suddivise per funzionalità.
- *FE* : all'interno di questa directory sono presenti i file con estensione .ui che rappresentano le schermate con cui l'utente interagisce. I file di FE sono stati generati dal software QT Designer che verrà descritto in seguito.
- *Docs*: questa cartella viene utilizzata come 'appoggio' per l'upload e il download dei documenti che l'utente seleziona all'interno delle schermate.
- *Images*: come per la cartella precedente, anche questa directory è stata utilizzata per memorizzare temporaneamente le immagini di profilo che l'utente inserisce tramite l'interazione con l'interfaccia grafica
- *Icons*: all'interno di questa directory sono presenti tutte le icone utilizzate per i widget del software ed il logo dell'azienda.

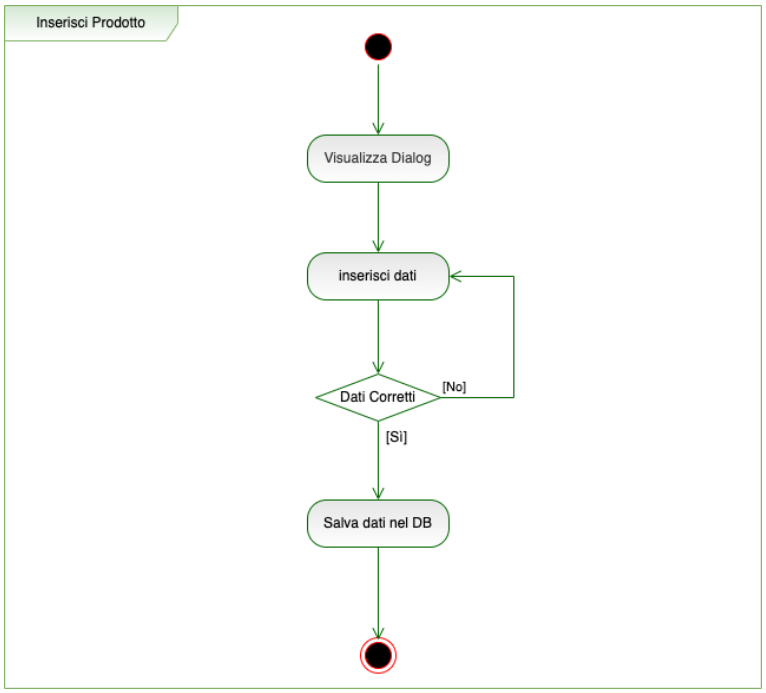
3.3 Diagramma delle attività

Di seguito sono riportati i diagrammi delle attività delle funzioni citate nei paragrafi precedenti.

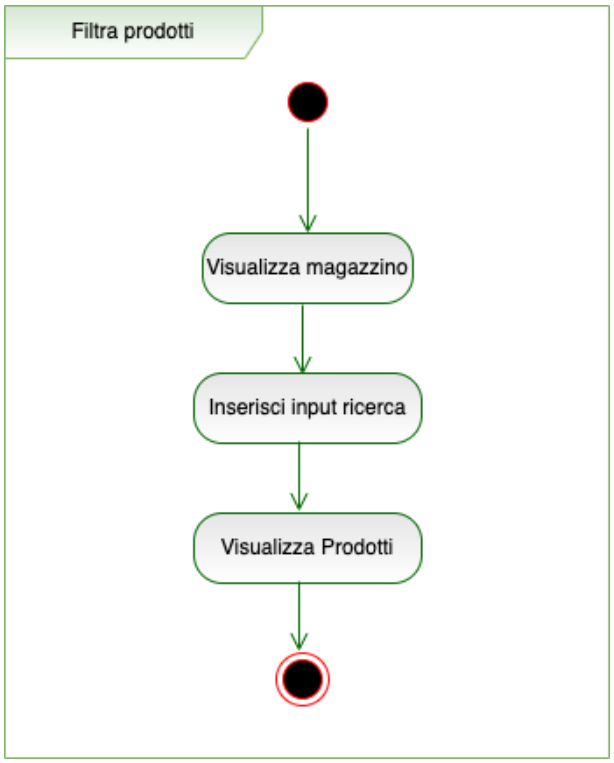
3.3.1 Gestione Magazzino



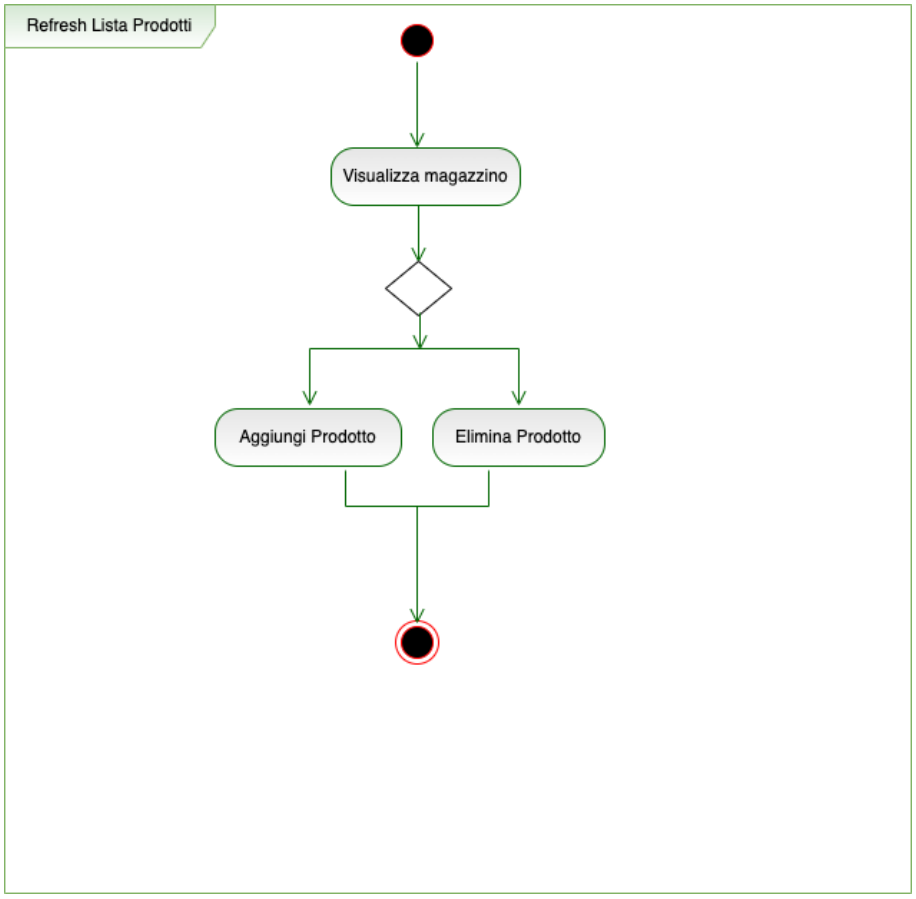
Visualizza Prodotti



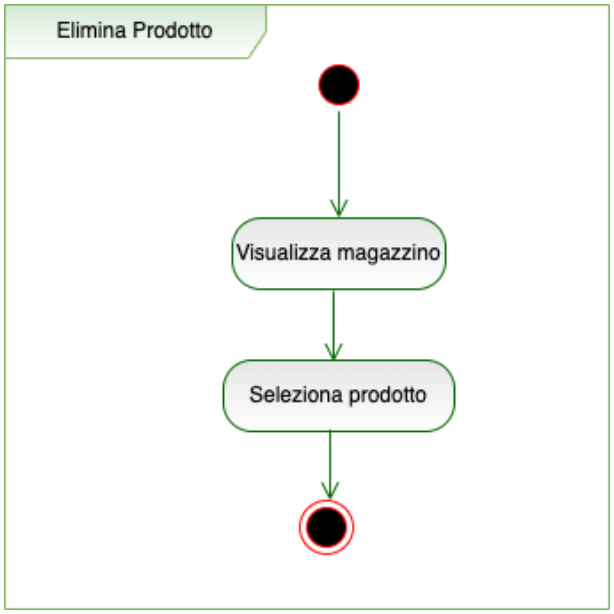
Aggiungi Prodotto



Filtra Prodotti

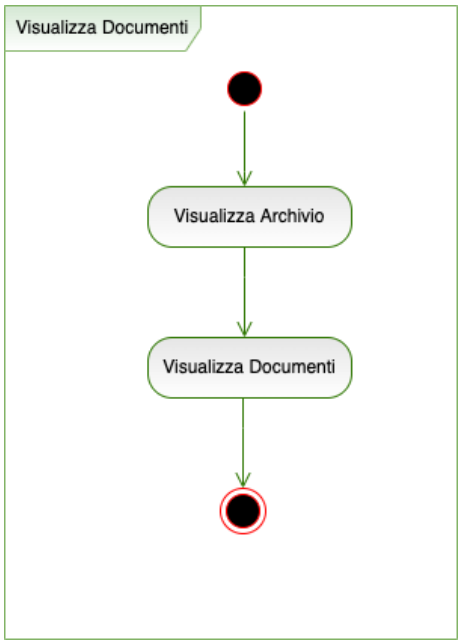


Refresh Lista Prodotti

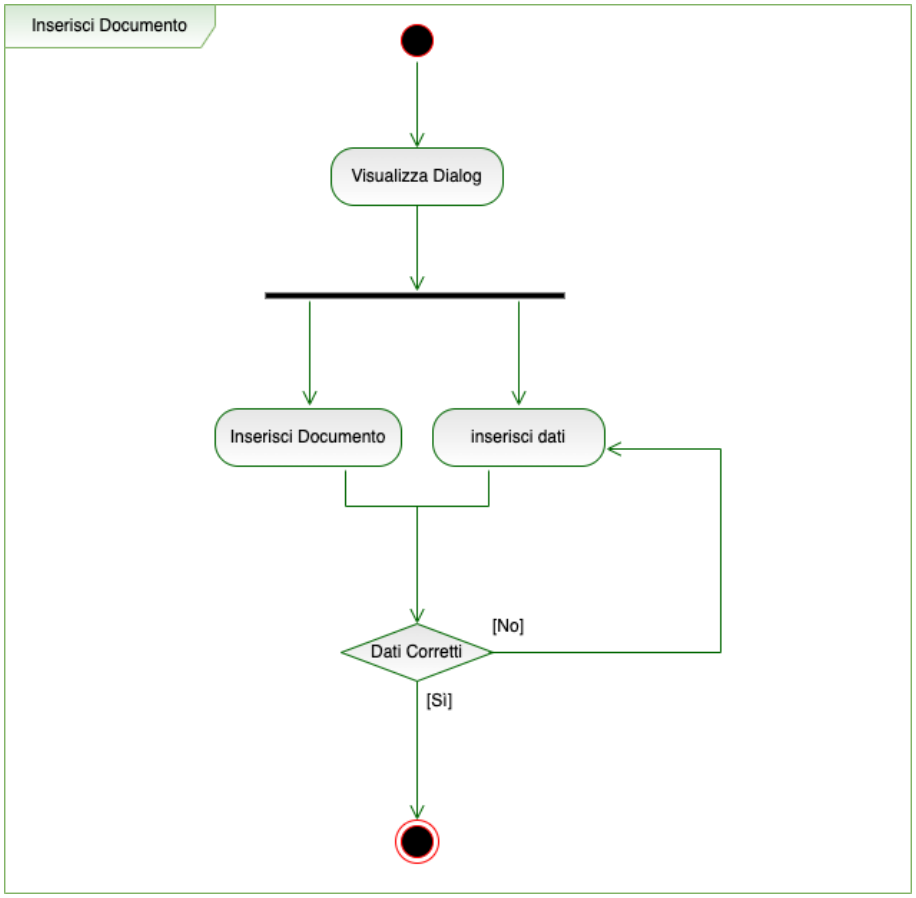


Elimina Prodotti

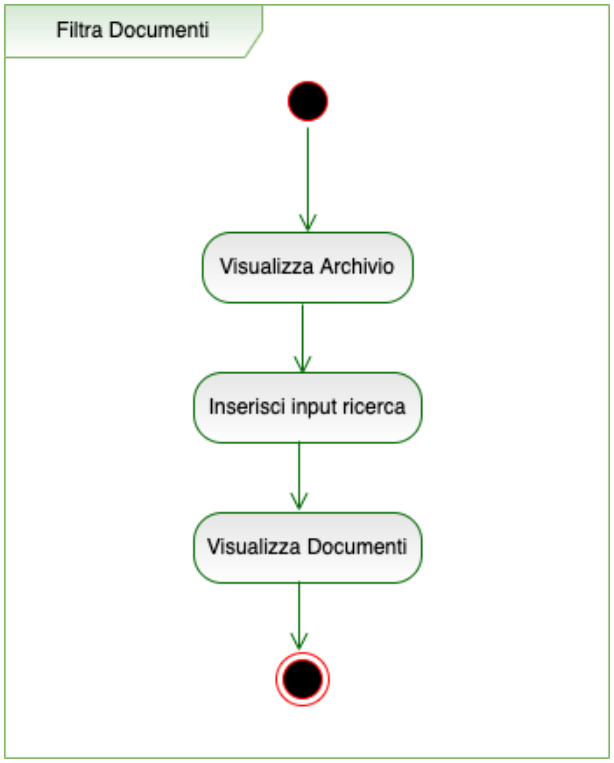
3.3.2 Gestione Documenti



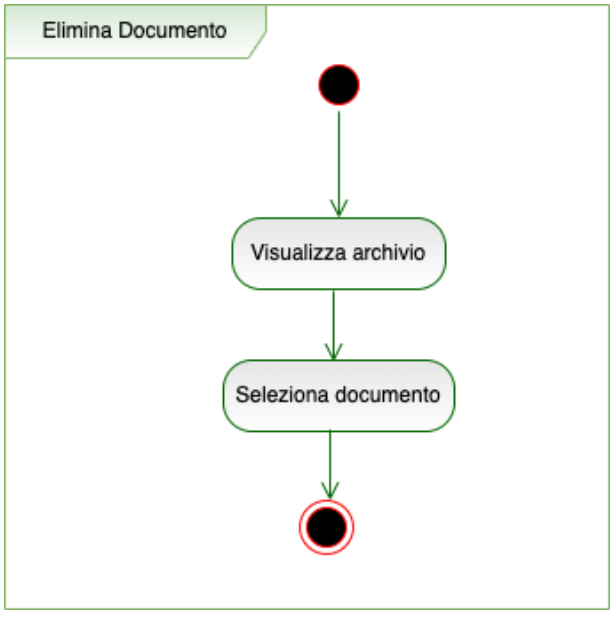
Visualizza Documenti



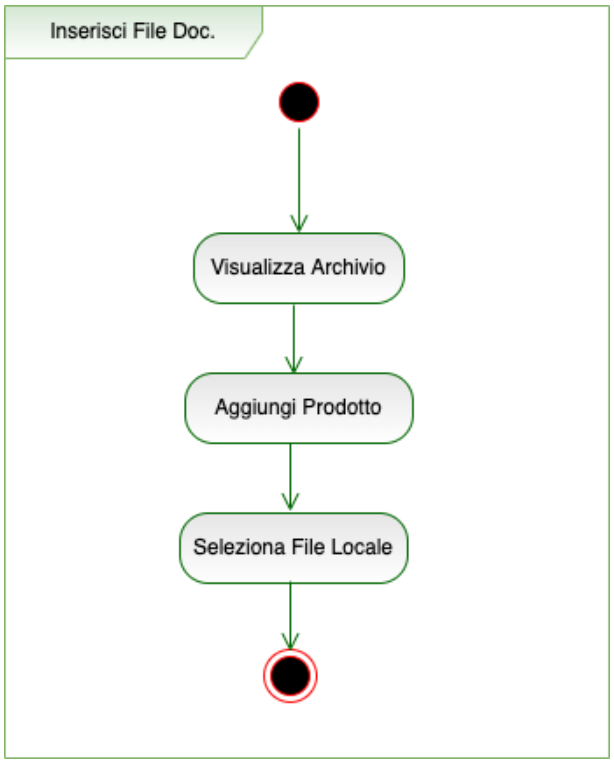
Aggiungi Documenti



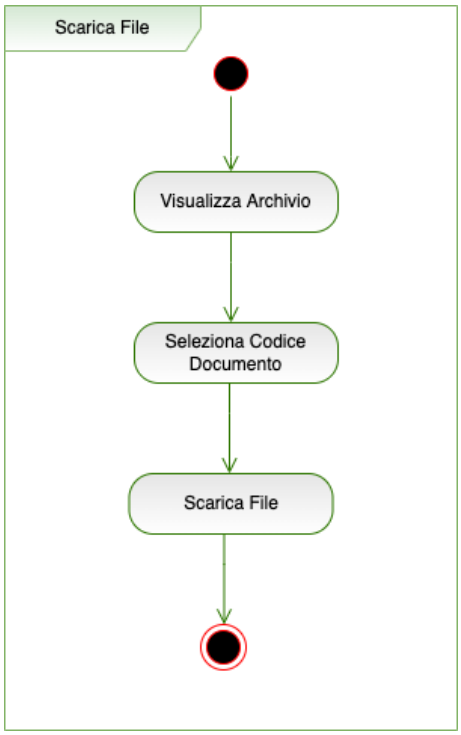
Filtra Documenti



Elimina Documento

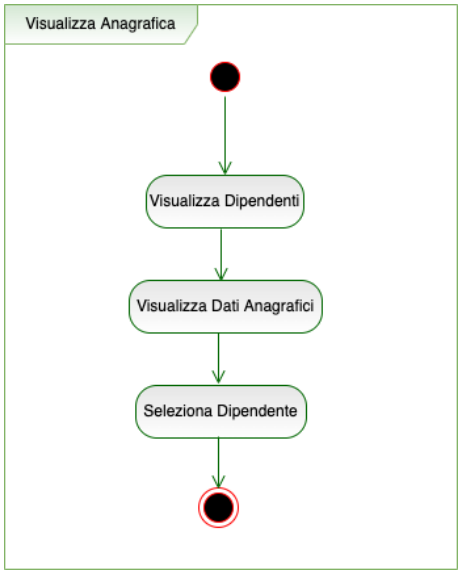


Inserisci File

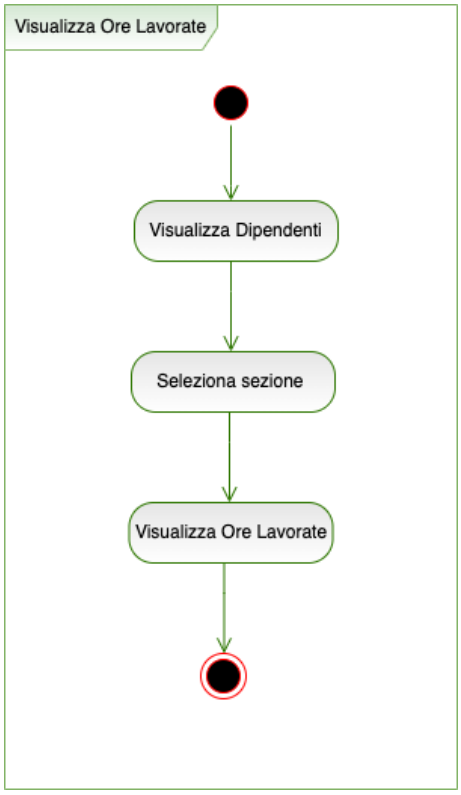


Scarica File

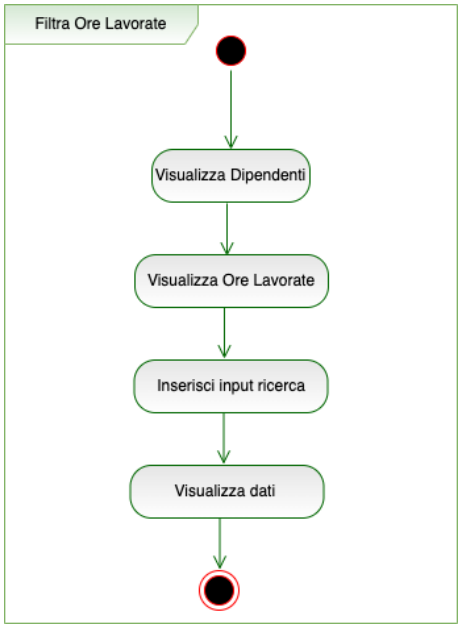
3.3.3 Gestione Dipendenti



Visualizza Anagrafica Dipendenti

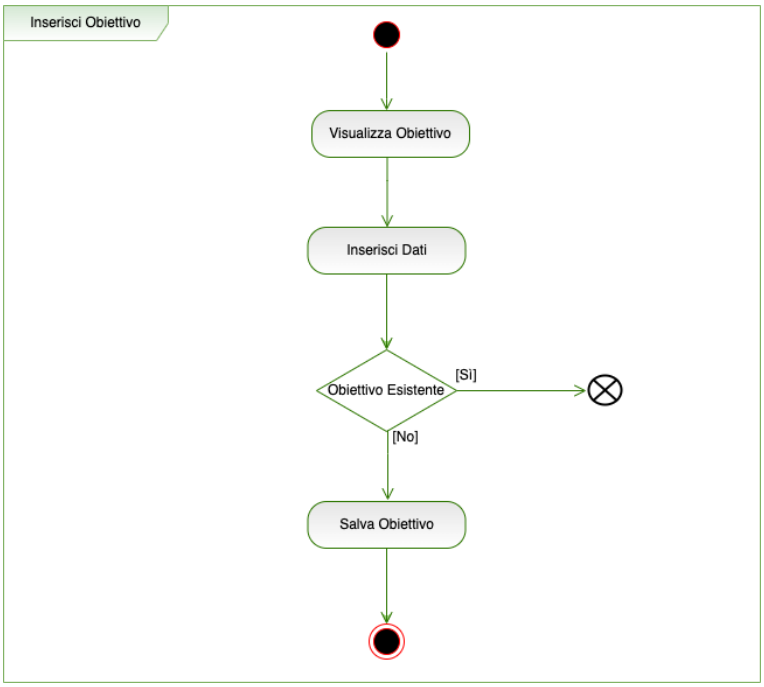


Visualizza Ore Lavorate



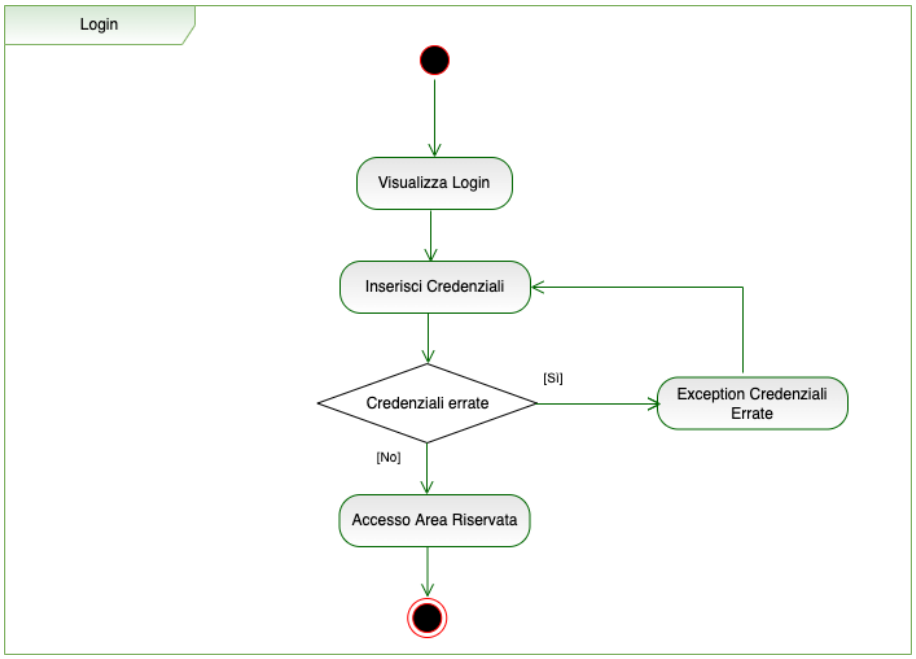
Ricerca Ore Lavorate

3.3.4 Gestione Obiettivo

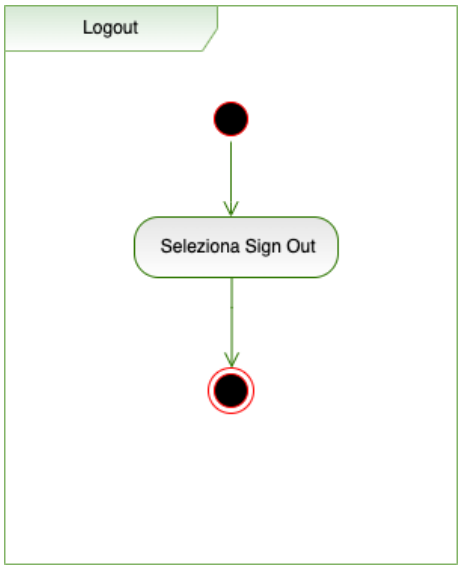


Inserisci Obiettivo

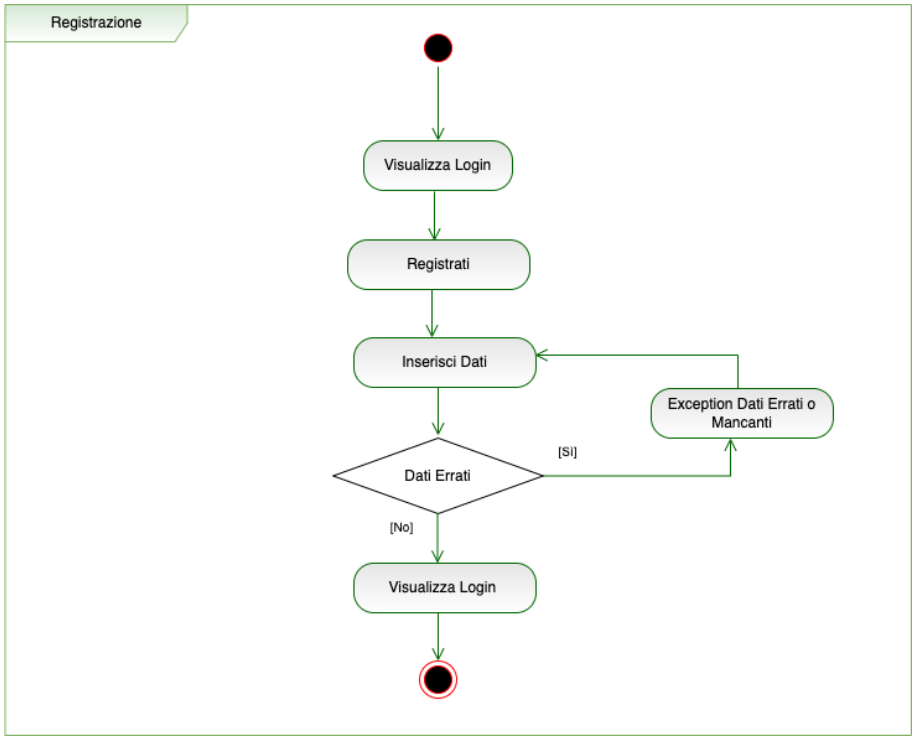
3.3.5 Autenticazione



Login



Logout



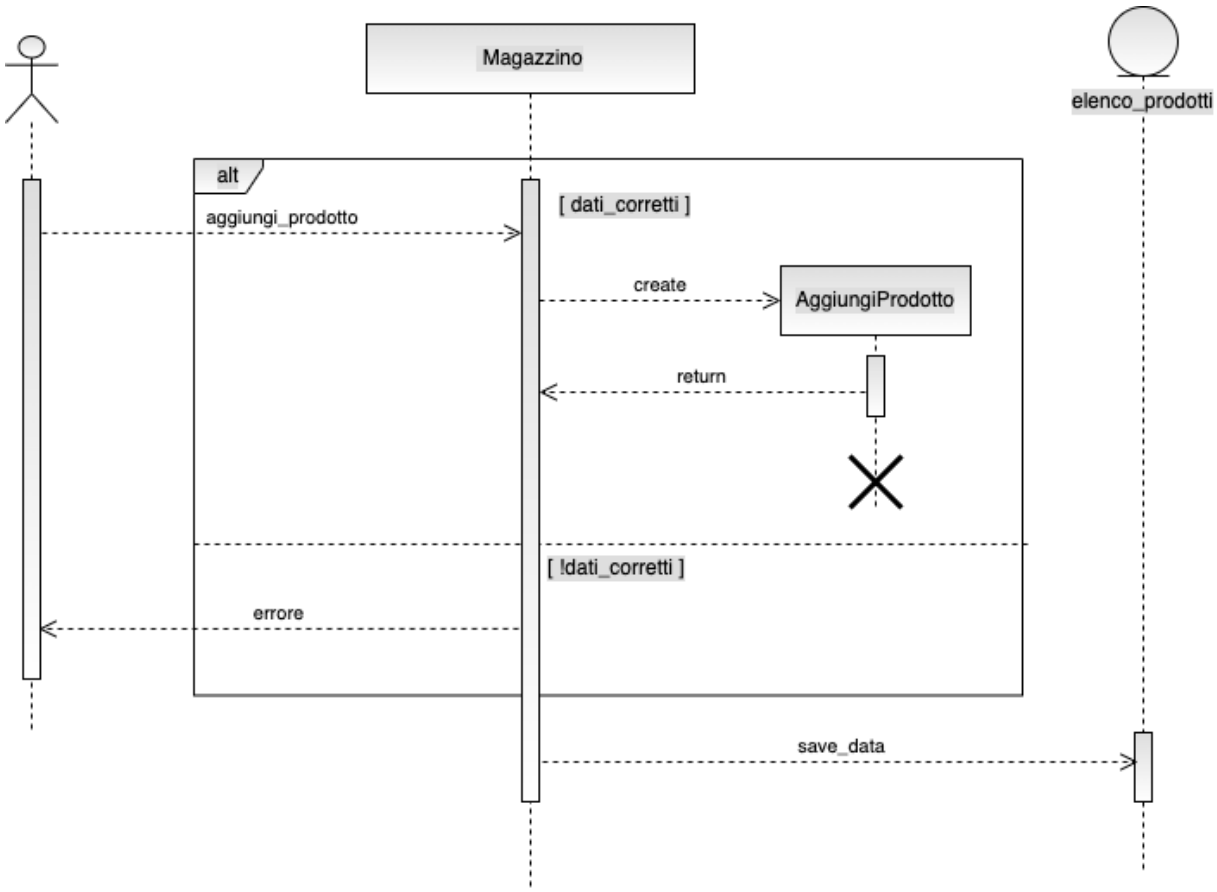
Registrazione

3.4 Diagramma delle sequenze

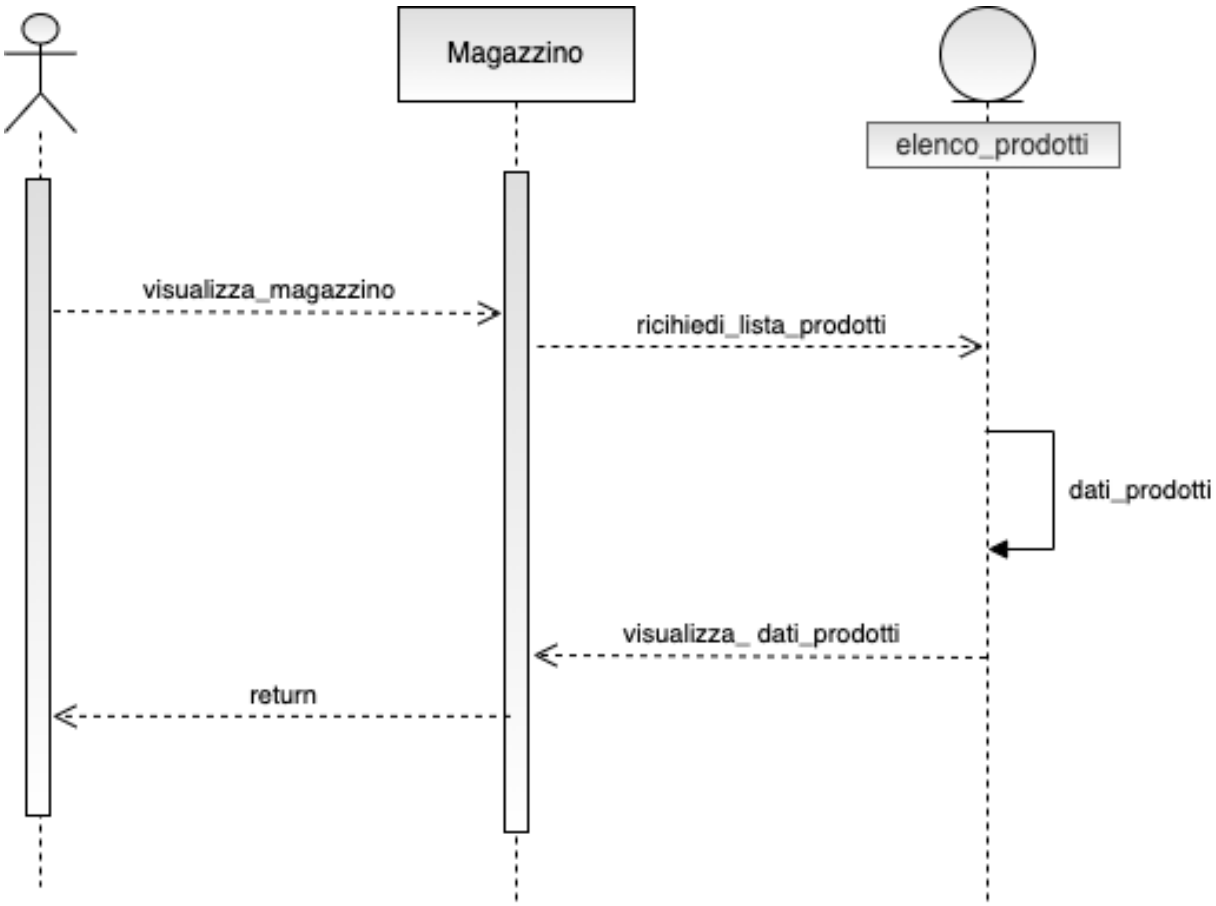
In questo paragrafo vengono riportati i diagrammi delle sequenze, mettendo in evidenza gli oggetti presenti nel sistema.

3.4.1 Gestione Magazzino

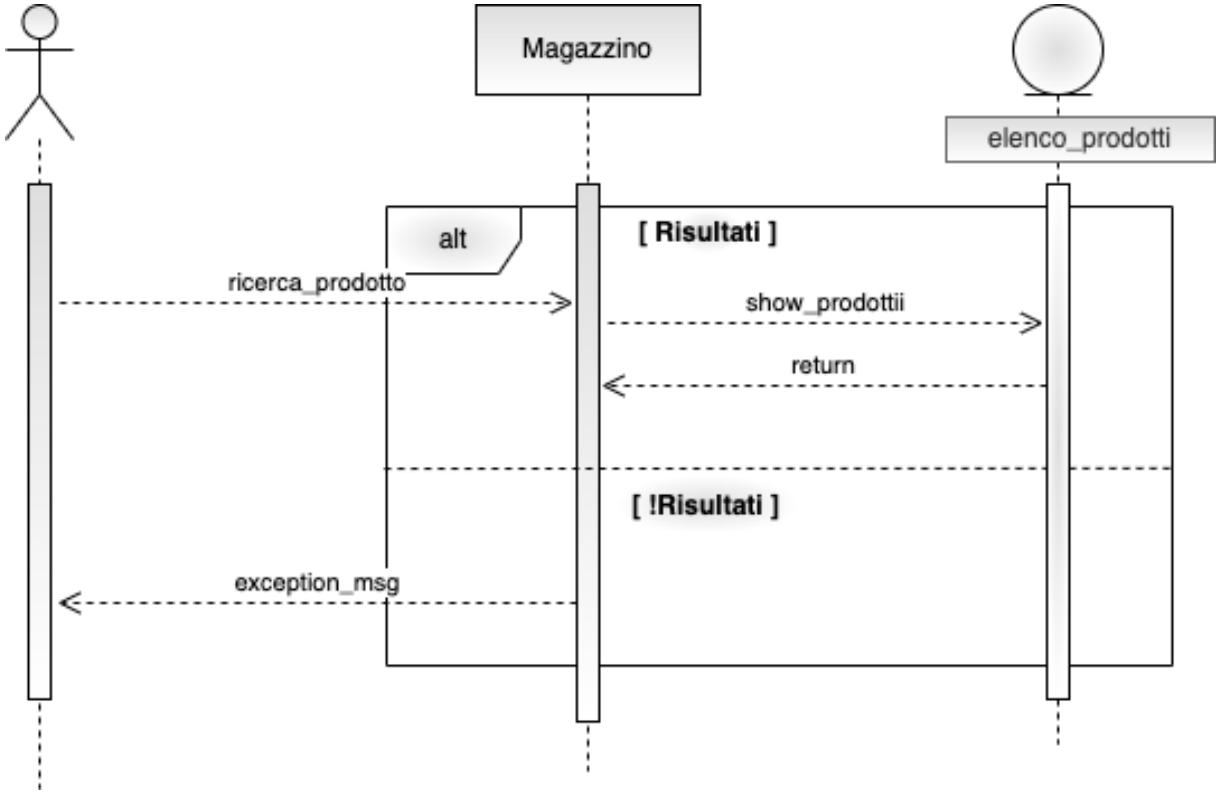
- Inserisci prodotto
- Visualizza prodotti
- Filtra prodotti
- Elimina prodotto
- Refresh lista prodotti



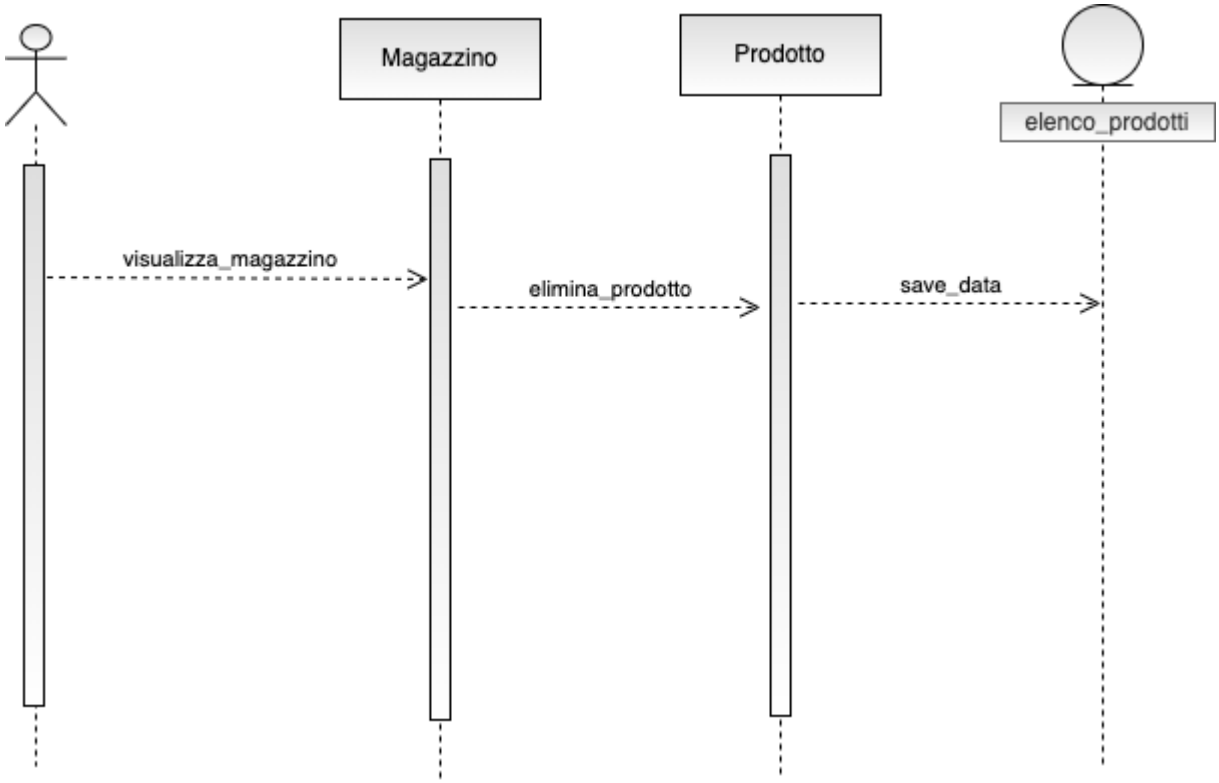
Inserisci Prodotto



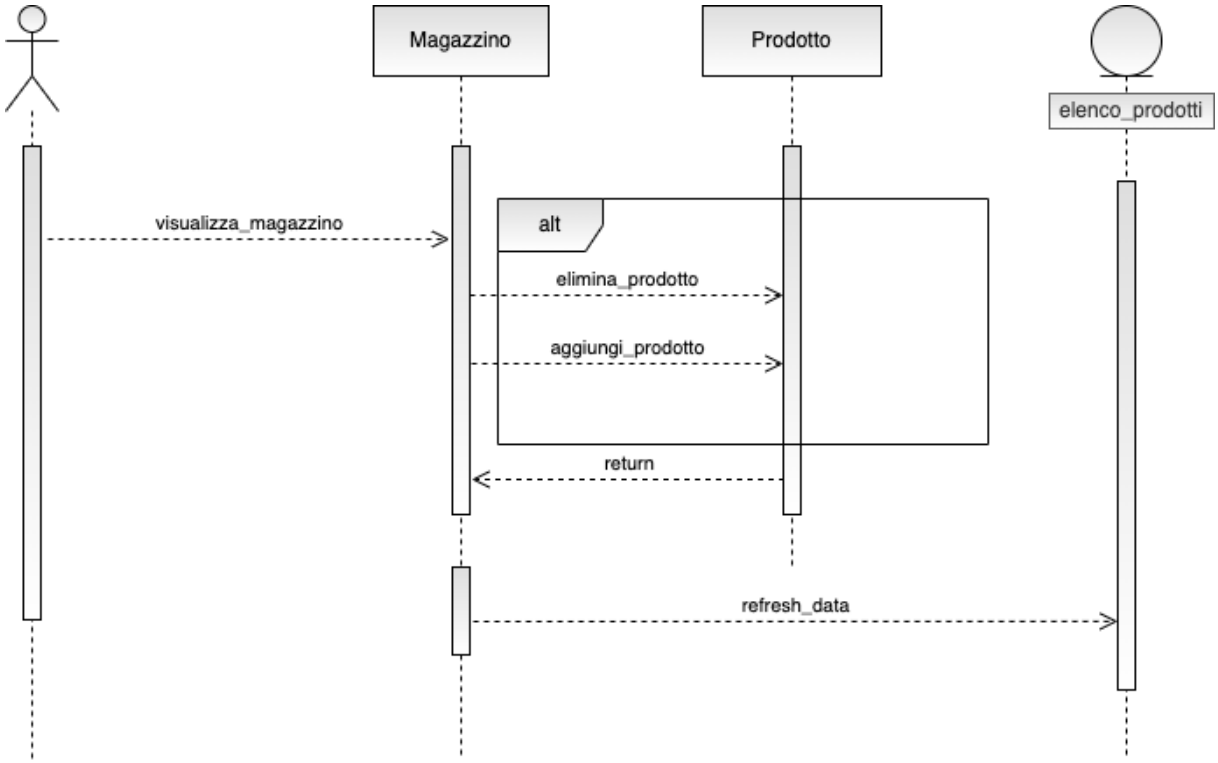
Visualizza Prodotti



Filtra Prodotti



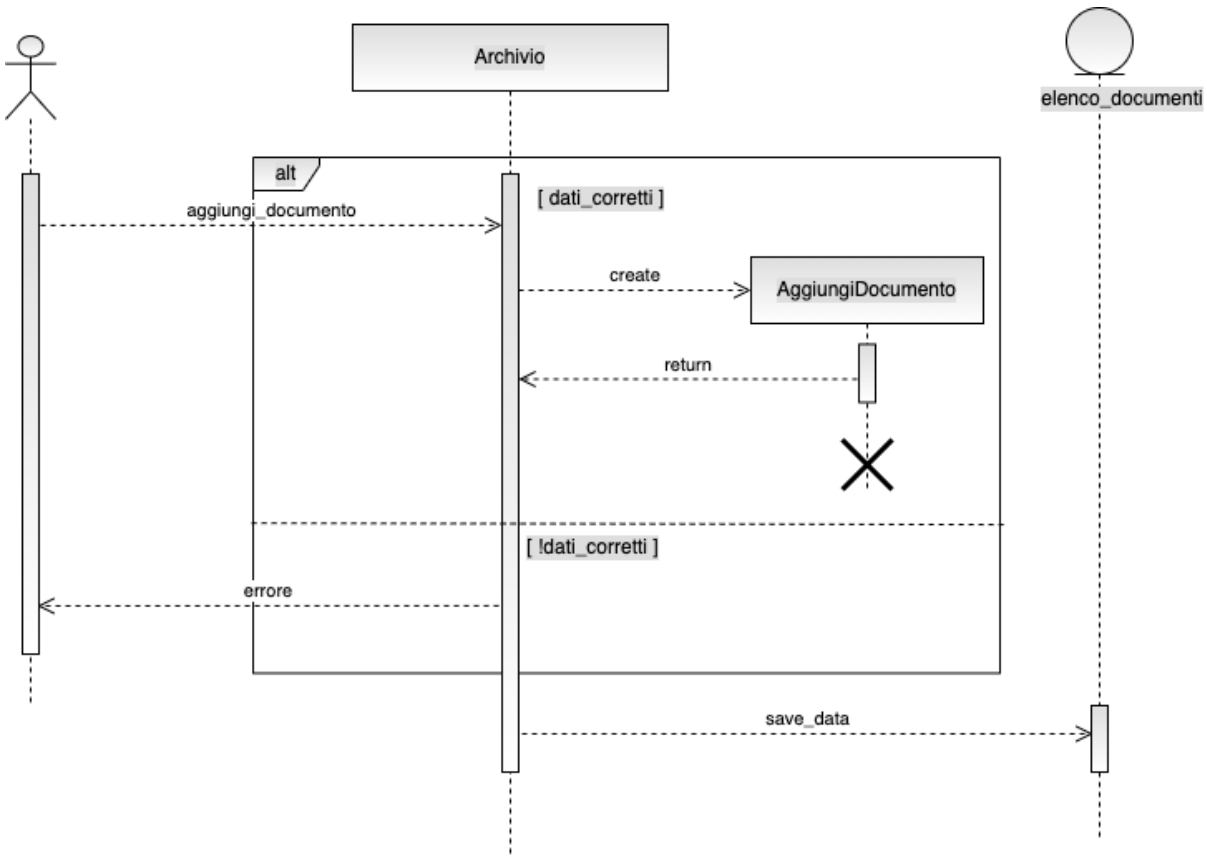
Elimina Prodotto



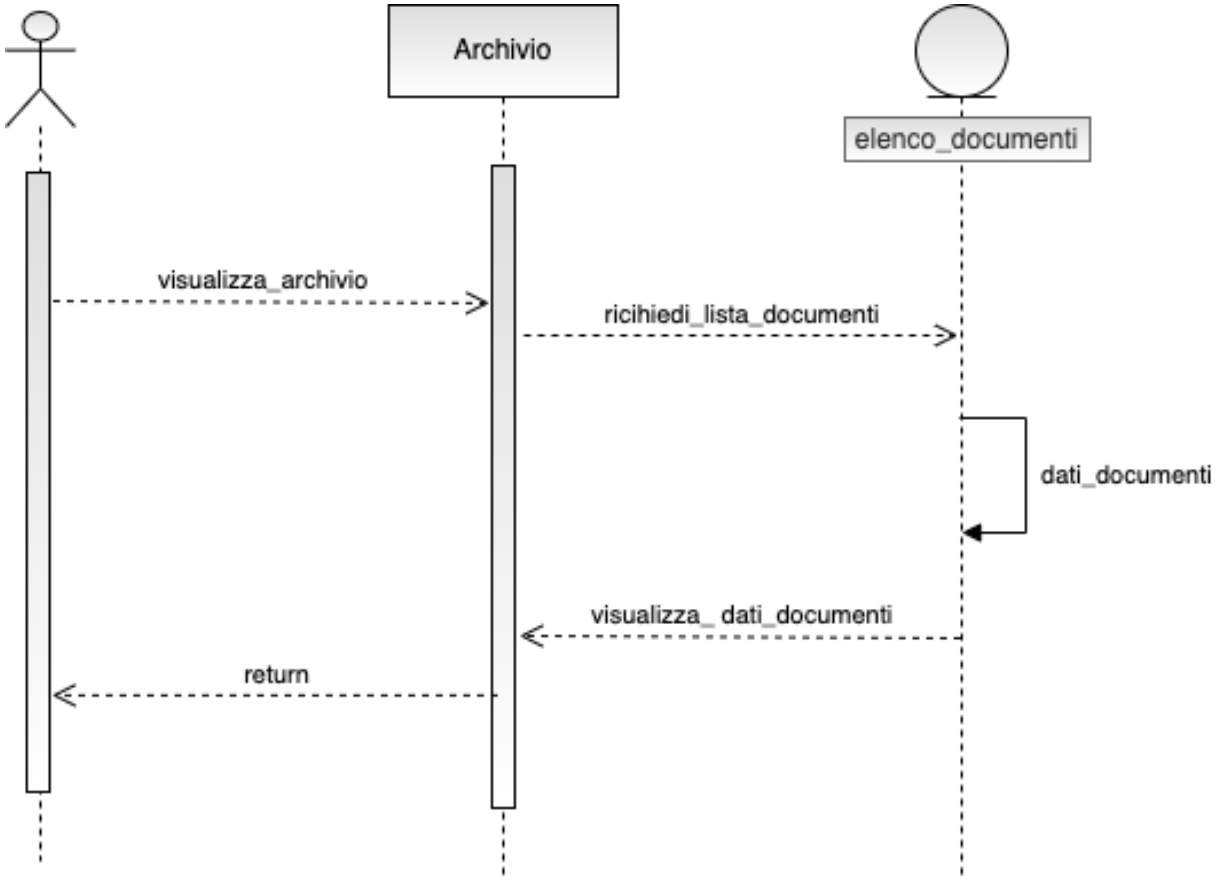
Refresh Lista Prodotti

3.4.2 Gestione Archivio

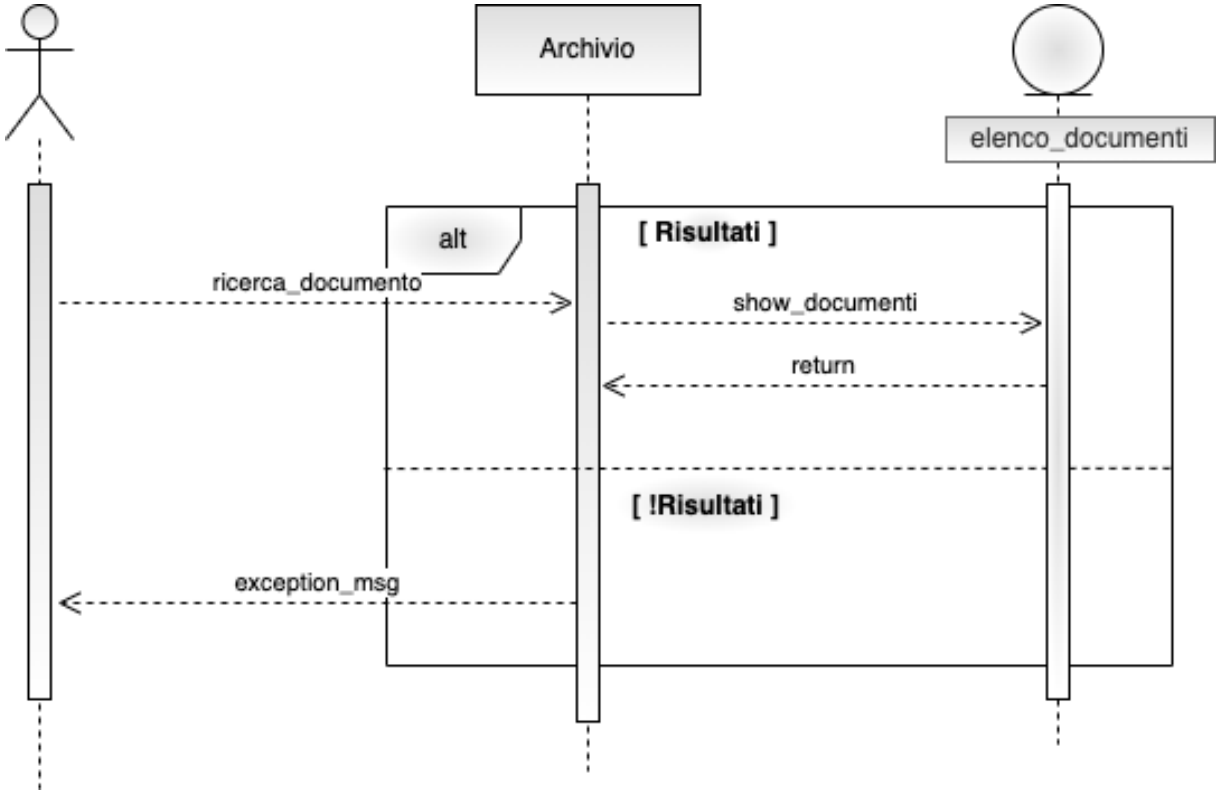
- Inserisci documento
- Visualizza documenti
- Filtra documenti
- Elimina documento
- Refresh lista documenti
- Inserisci file
- Scarica file



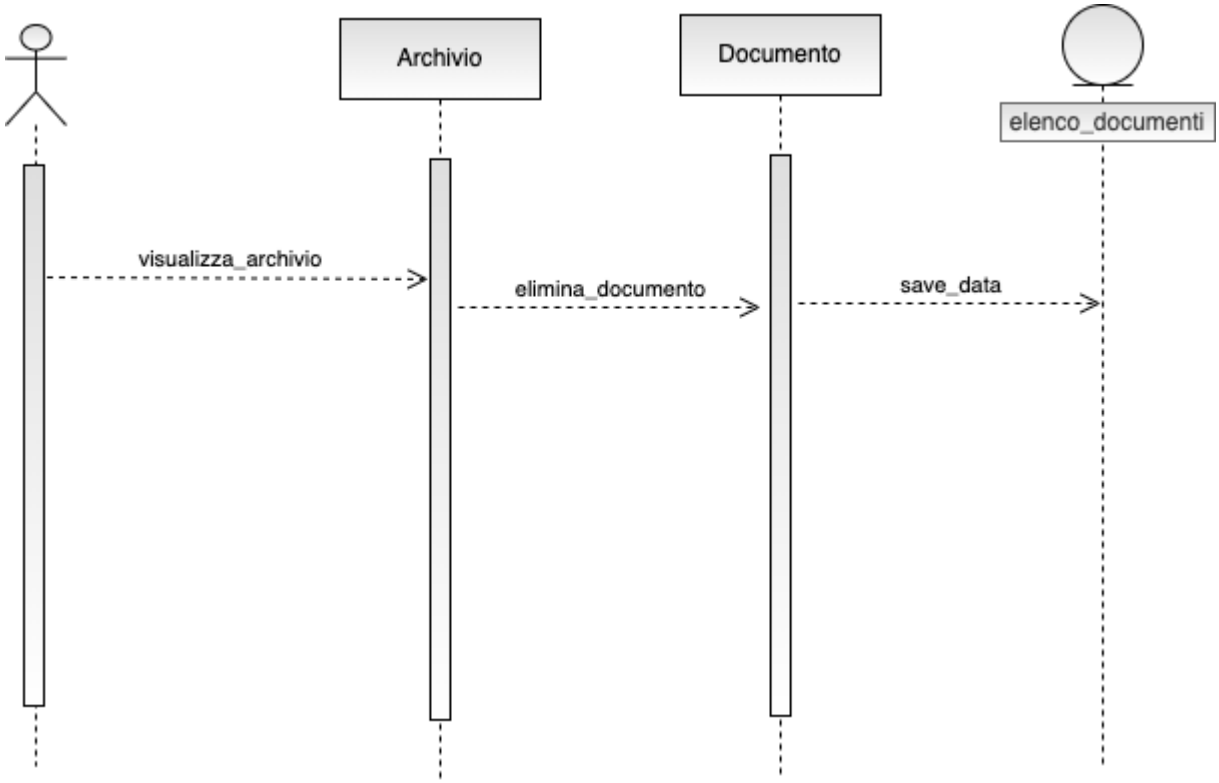
Inserisci Documento



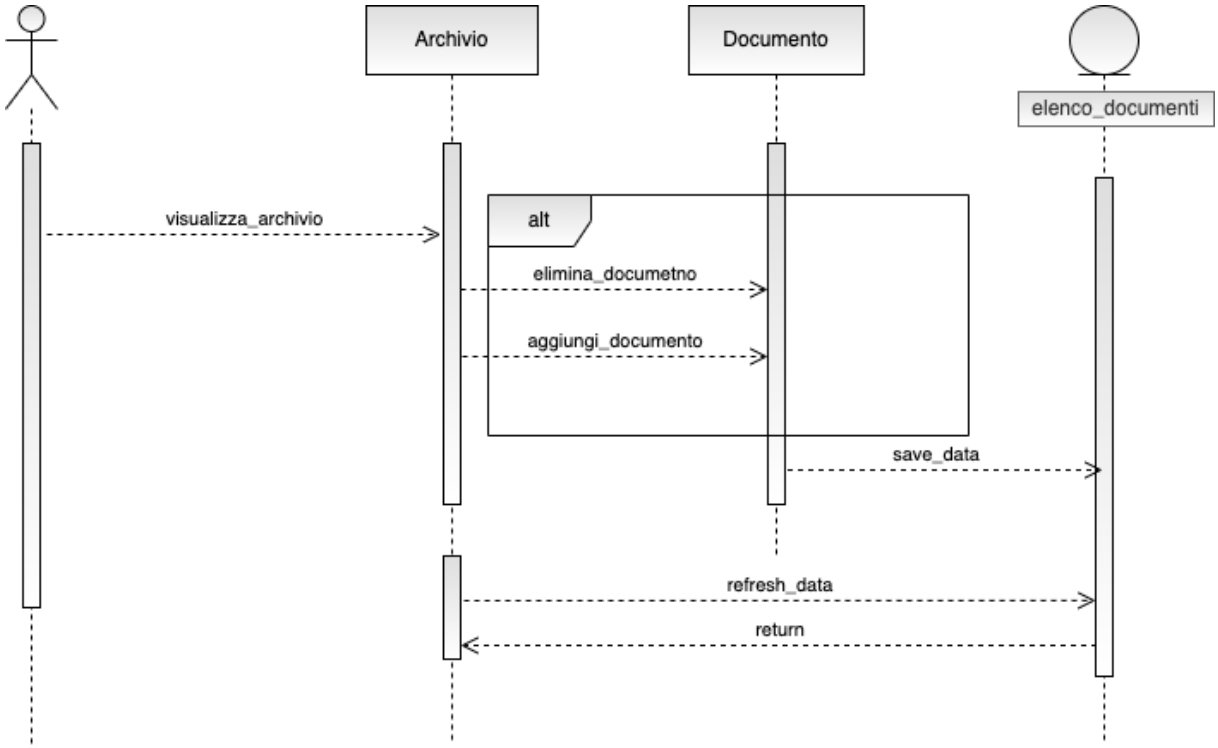
Visualizza Prodotti



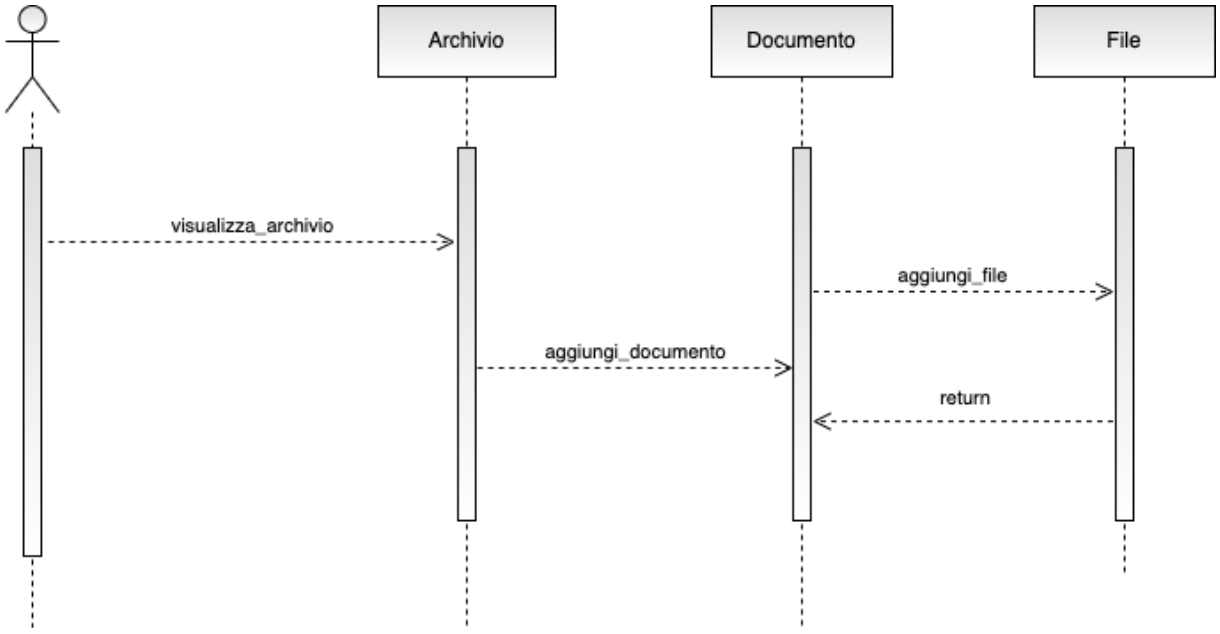
Filtra Prodotti



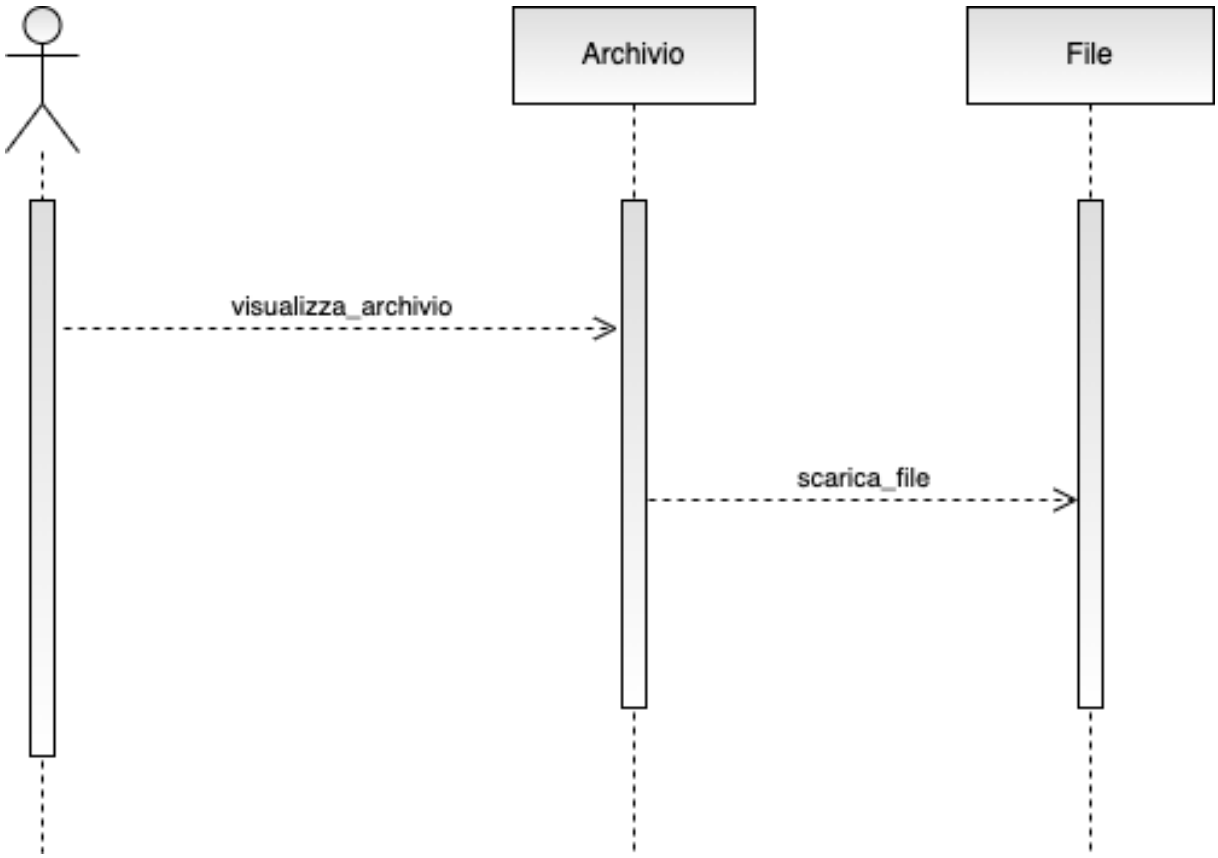
Elimina Prodotto



Refresh Lista Prodotti



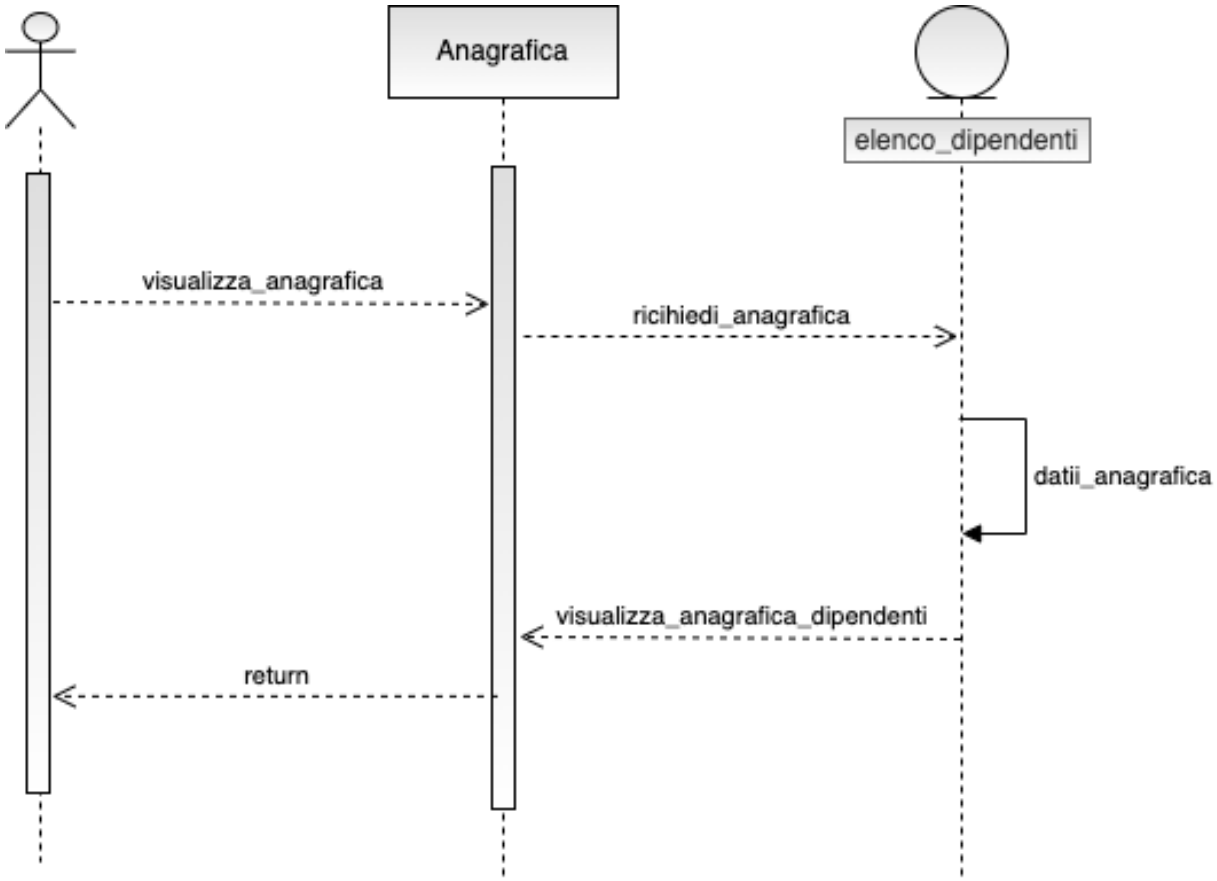
Inserisci File Documento



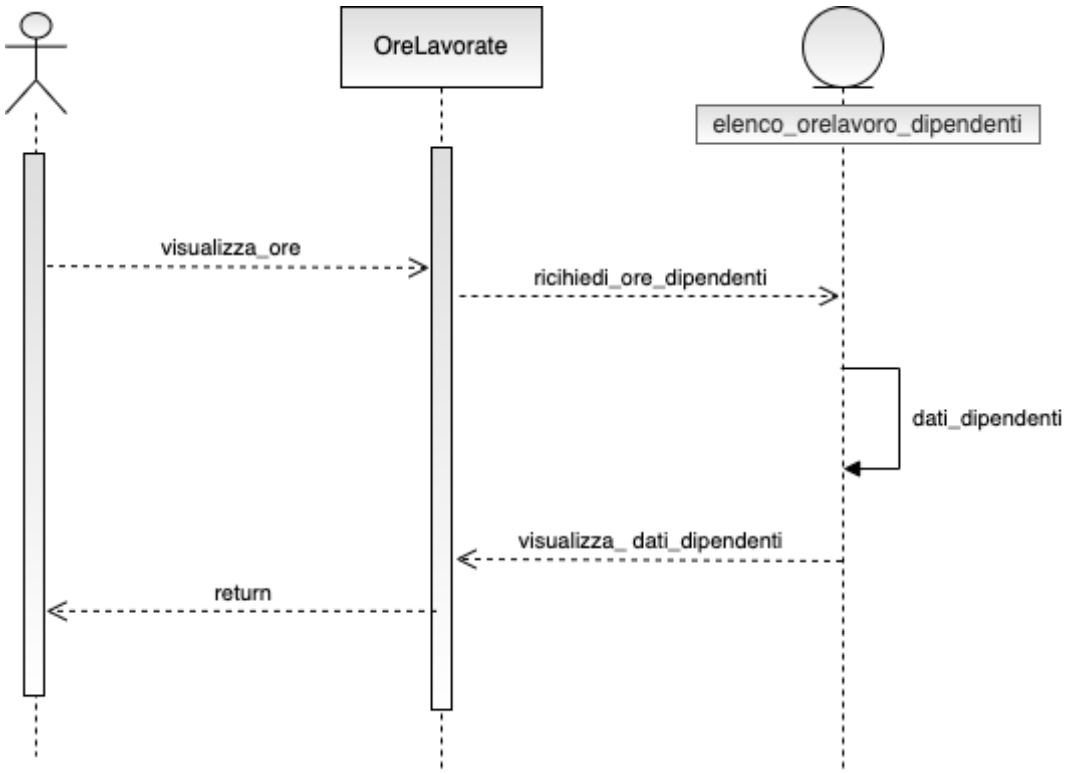
Scarica File Documentoi

3.4.3 Gestione Dipendenti

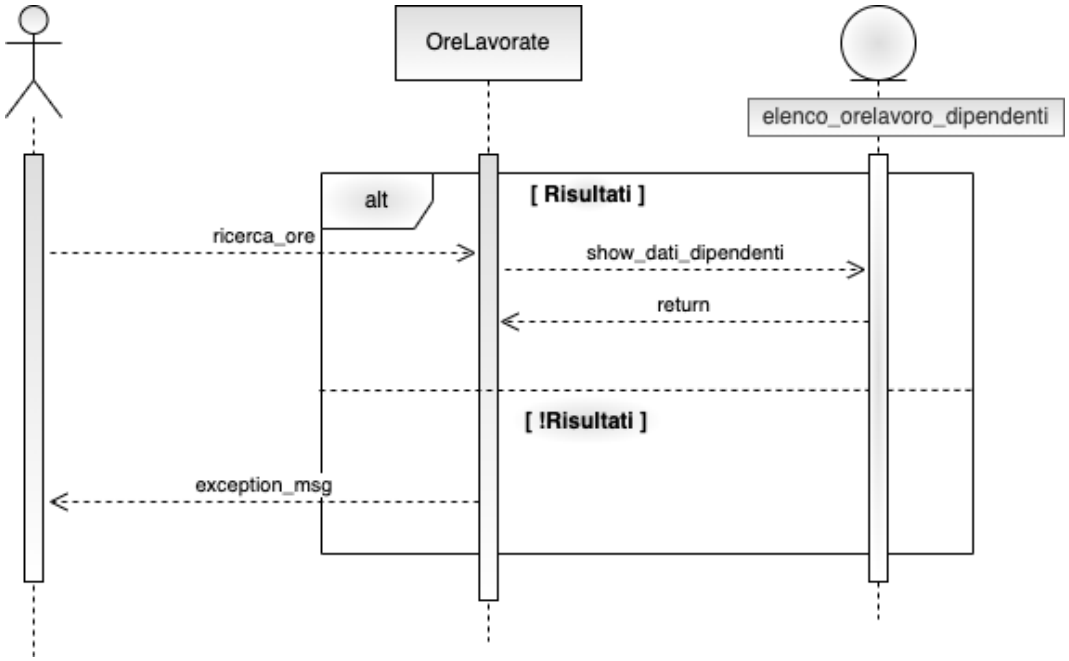
- Visualizza anagrafica
- Visualizza ore lavorate
- Filtra ore lavorate



Visualizza Anagrafica



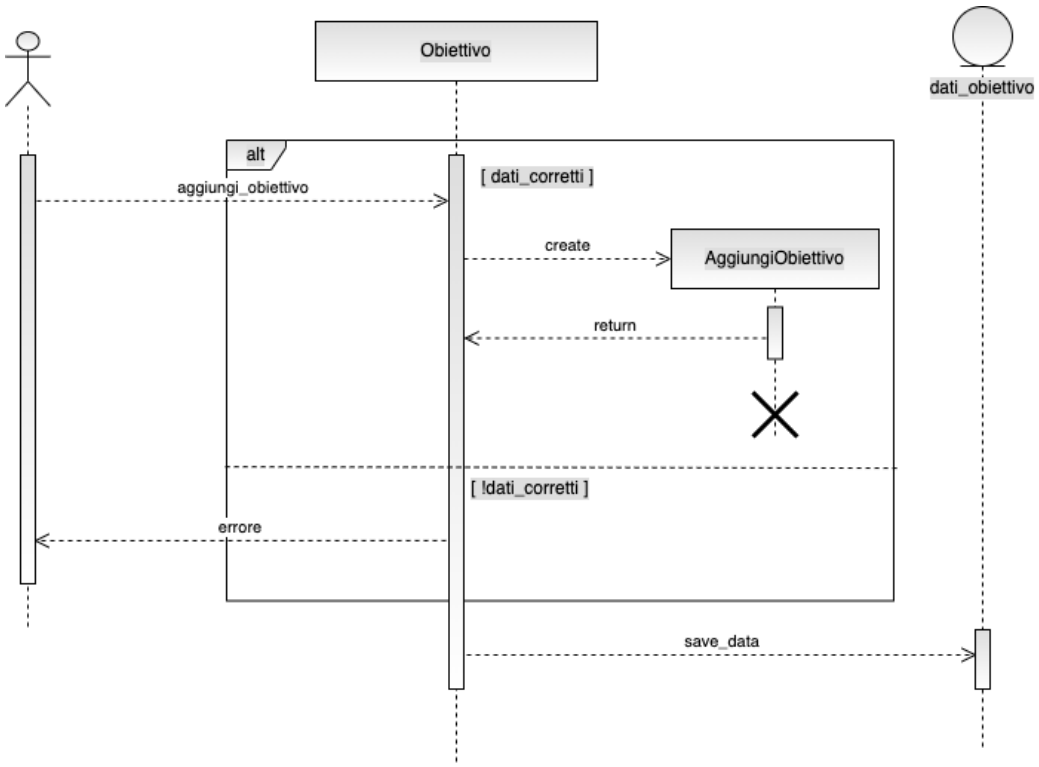
Visualizza Ore Lavorate



Filtra Ore Lavorate

3.4.4 Gestione obiettivi

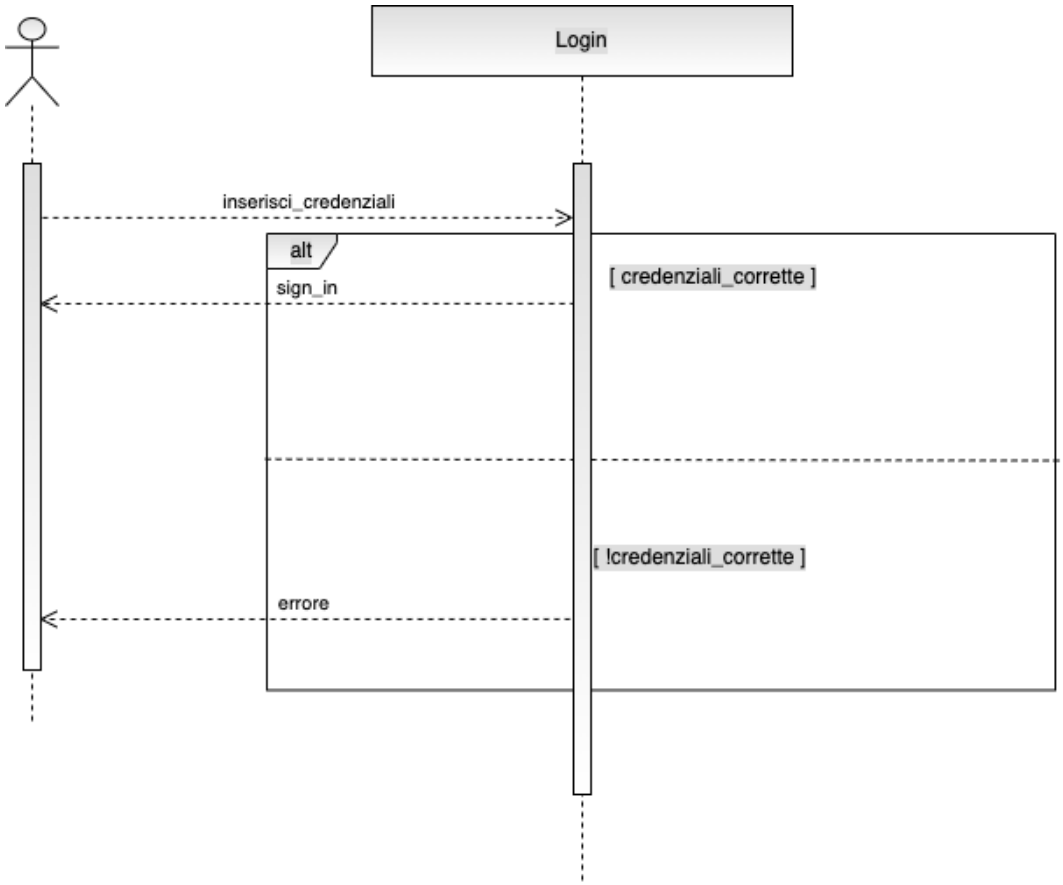
- Inserisci Obiettivo



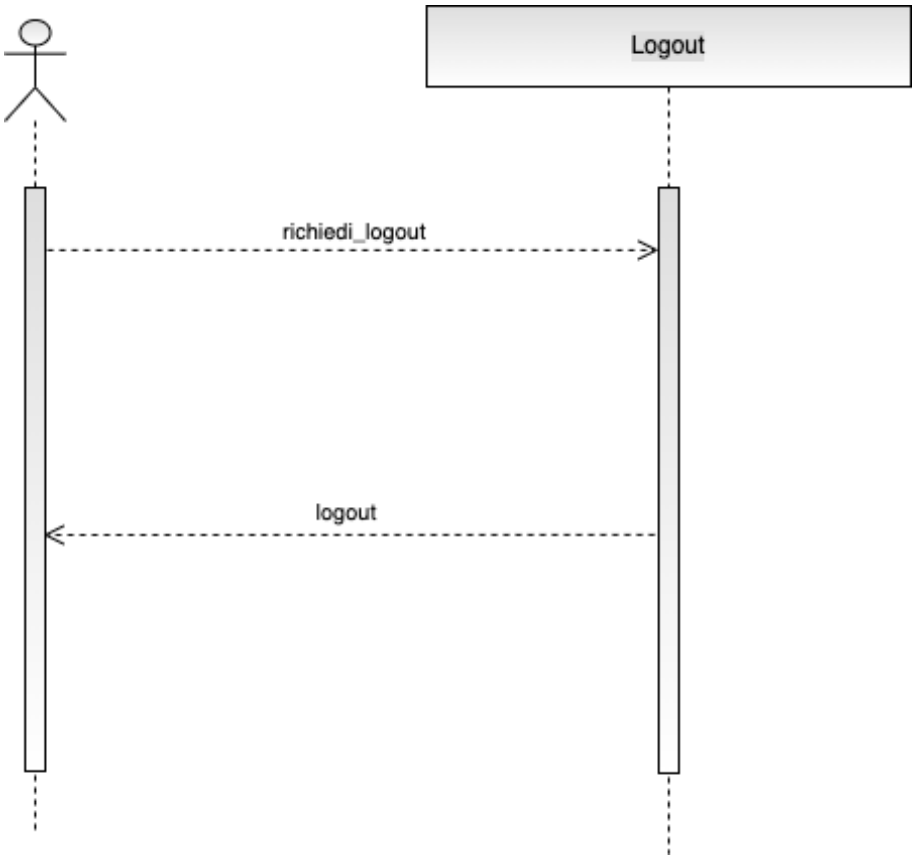
Inserisci Obiettivo

3.4.5 Autenticazione

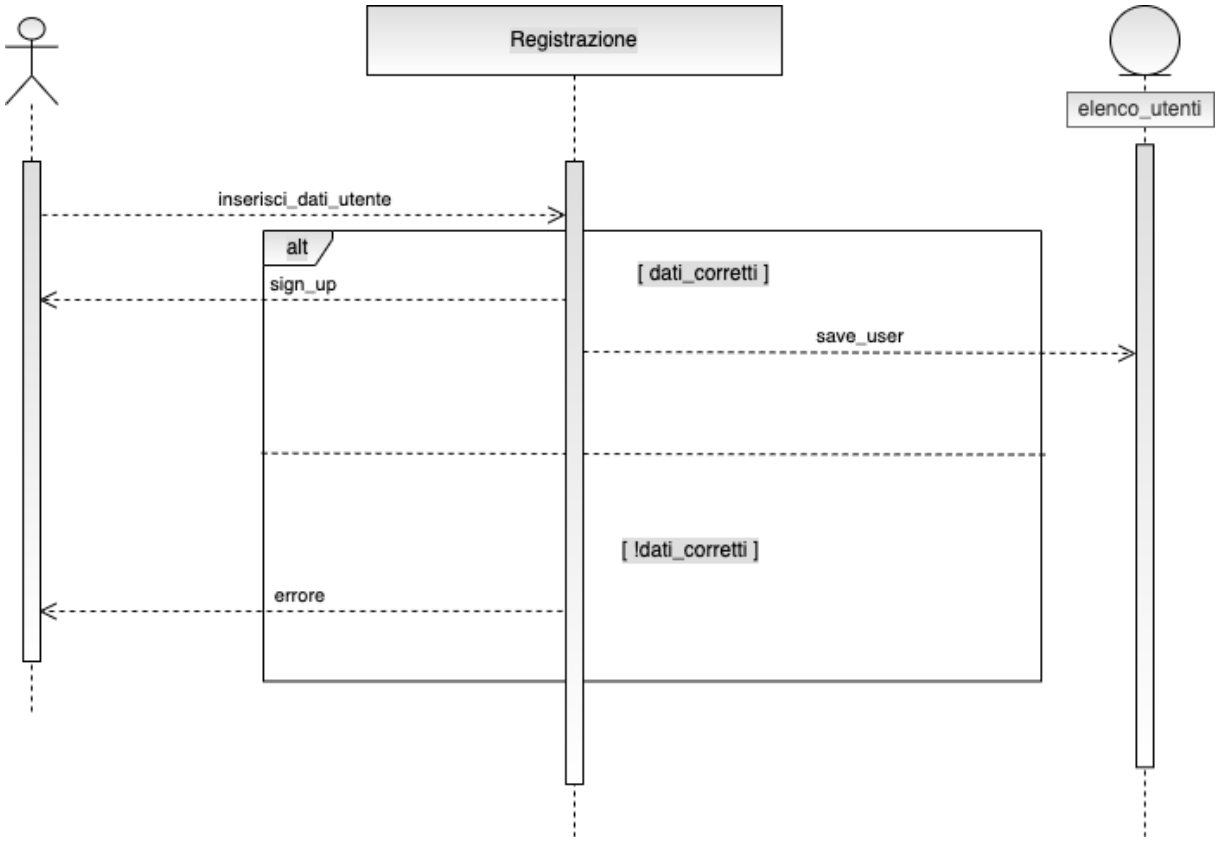
- Login
- Logout
- Registrazione



Logini



Logout



Registrazione

3.5 Imprementazione back-end

3.5.1 Librerie utilizzate

Per lo sviluppo del software lato BE sono state utilizzate le funzionalità di alcune librerie o set di classi, in particolare:

- *PyQT*: è un insieme di classi che permettono l'accesso alle librerie grafiche QT. Per l'implementazione del codice si è scelto di utilizzare la release più recente, ovvero PyQt5.
  Le classi che mette a disposizione sono suddivise in più di 50 moduli, tuttavia per l'implementazione del software ne sono stati usati solo alcuni:
  - *QtCore*: contiene le classi base di PyQt, quali le astrazioni per le animazioni, macchine a stati, thread, files, memoria condivisa, espressioni regolari.
    Le principali classi utilizzate sono:
    * *QtDate*: è un oggetto che rappresenta un giorno specifico, indipendentemente dal calendario, dalle impostazioni locali o da altre impostazioni utilizzate durante la creazione o fornite dal sistema.Un oggetto QDate viene in genere creato fornendo esplicitamente i numeri di anno, mese e giorno.
    * *Qt*: contiene identificatori vari utilizzati in tutta la libreria Qt, come, ad esempio, Qt.GlobalColor, Qt.Orientation, etc.
  - *QWidgets*: si tratta di una vasta gamma di widget (form, pulsanti, label, treeview, tabelle, checkbox e quant'altro) adatti alla creazione di interfacce grafiche desktop. I widget più utilizzati all'interno del software gestionale sono:
    * *QFileDialog*: consente a un utente di accedere al file system per selezionare uno o più file o una directory.
    * *QDialog*: utilizzata per creare una schermata per attività a breve termine e brevi comunicazioni con l'utente.

    * *QMainWindow*: fornisce un framework per la creazione dell'interfaccia utente di un'applicazione. Qt ha QMainWindow e le relative classi per la gestione della schermata principale, inoltre ha un proprio layout a cui si possono aggiungere diversi elementi come QToolBars, QDockWidgets, una QMenuBar e una QStatusBar. Il layout ha un'area centrale che può essere occupata da qualsiasi tipo di widget.
    * *QLineEdit*: consente all'utente di inserire e modificare una singola riga di testo normale con l'utilizzo di un'utile raccolta di funzioni di modifica, tra cui annulla e ripristina, taglia e incolla e trascina e rilascia.
    * *QPushButton*: è forse il widget più comunemente utilizzato in qualsiasi interfaccia utente grafica.
      Un pulsante viene premuto per chiedere al computer di eseguire un'azione o di rispondere a una domanda. I pulsanti tipici di questo oggetto sono OK, Applica, Annulla, Chiudi, Sì, No e Guida. Un pulsante in genere contiene un'etichetta di testo che ne descrive l'azione.
    * *QStackedWidget*: consente di creare più pagine nello stesso layout. Quando viene aggiunta una pagina il widget la inserisce in una propria lista interna.
  - *QtGui*: fornisce un insieme di classi per la gestione delle finestre, immagini, font e testi. In genere, viene utilizzata congiuntamente al modulo "QtWidgets". Gli oggeti utilizzati da questo modulo sono:
    * *QIcon*: consente di creare pixmap (immagini) più piccole, questi oggetti vengono utilizzati per rappresentare una determinata azione.
    * *QPixmap*: consente di gestire immagini all'interno di un oggetto o di un widget.

Le classi, gli oggetti e i moduli sopra citati sono stati i più utilizzati per lo sviluppo del software gestionale.
Gli oggetti utilizzati con minore frequenza non sono stati riportati, ma

sono comunque figli dei moduli sopra elencati.

- *Os*: modulo utilizzato per far interagire il programma con il sistema operativo.
  E' stato utilizzato, in particolare,, per aggiungere,spostare o eliminare file all'interno delle directory.

- *Sys*: fornisce l'accesso ad alcune variabili usate o mantenute dall'interprete nonchè a funzioni che interagiscono fortemente con l'interprete stesso.

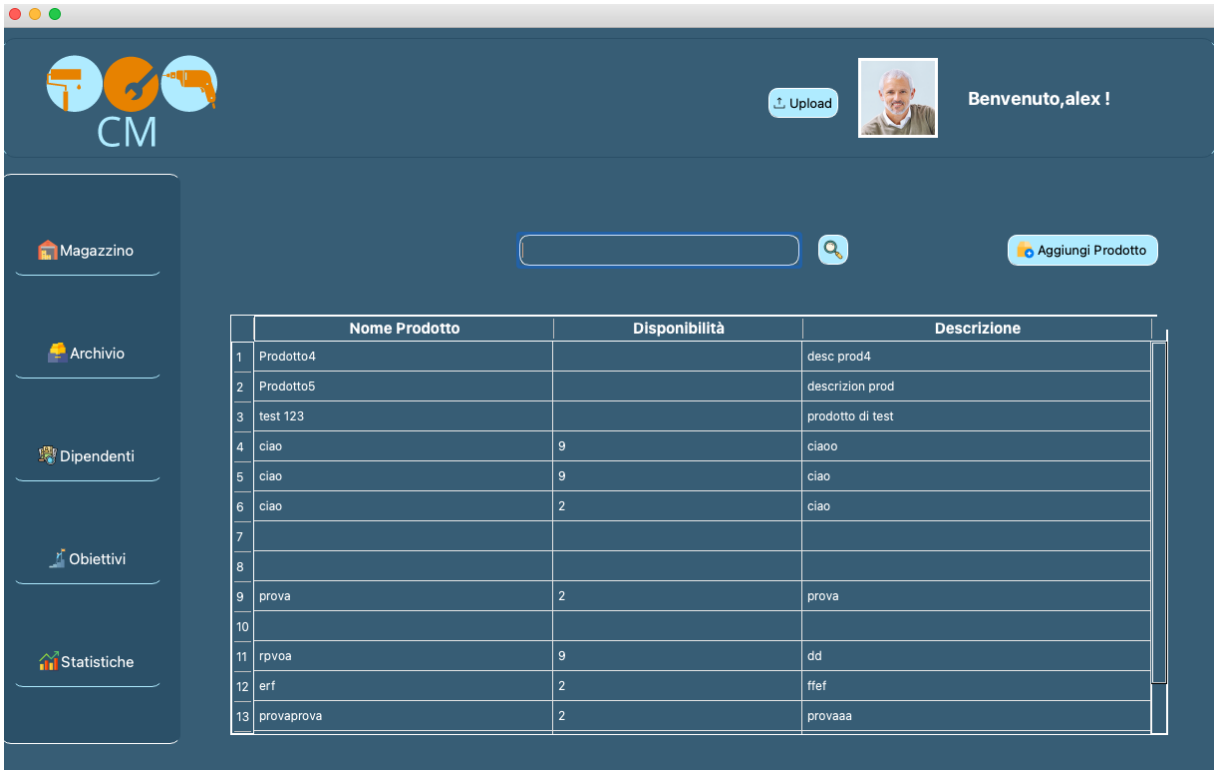
3.6 Implementazione front-end

Per lo sviluppo dell'interfaccia grafica è stato utilizzato il software QT Designer, dal quale sono stati generati i file .ui che rappresentano le schermate del programma con cui interagisce l'utente.

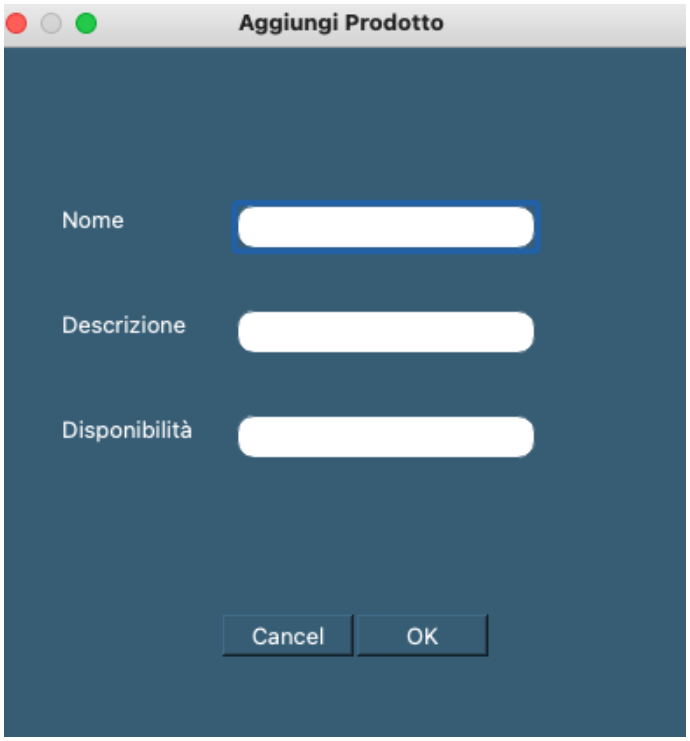
3.6.1 Utilizzo di QT Designer

QT Designer è un software editor che permette la creazione di GUI con strumenti di tipo Drag and Drop.
Con il tool sopra citato è possibile comporre e personalizzare le finestre o le finestre di dialogo in modo WYSIWYG (what-you-see-is-what-you-get) e testarle utilizzando stili e risoluzioni diversi.
Widget e moduli creati con Qt Designer si integrano perfettamente con il codice programmato, utilizzando il meccanismo dei segnali e degli slot di Qt, in modo da poter assegnare facilmente il comportamento agli elementi grafici. Tutte le proprietà impostate in Qt Designer possono essere modificate dinamicamente all'interno del codice.
Le interfacce vengono richiamate nei file di BE con l'utilizzo della funzione loadUI presente nel set di classi PyQt. Inoltre le schermate generate possono essere trasformate in file .py in modo da manipolare direttamente da codice gli oggetti o widget presenti nei file .ui

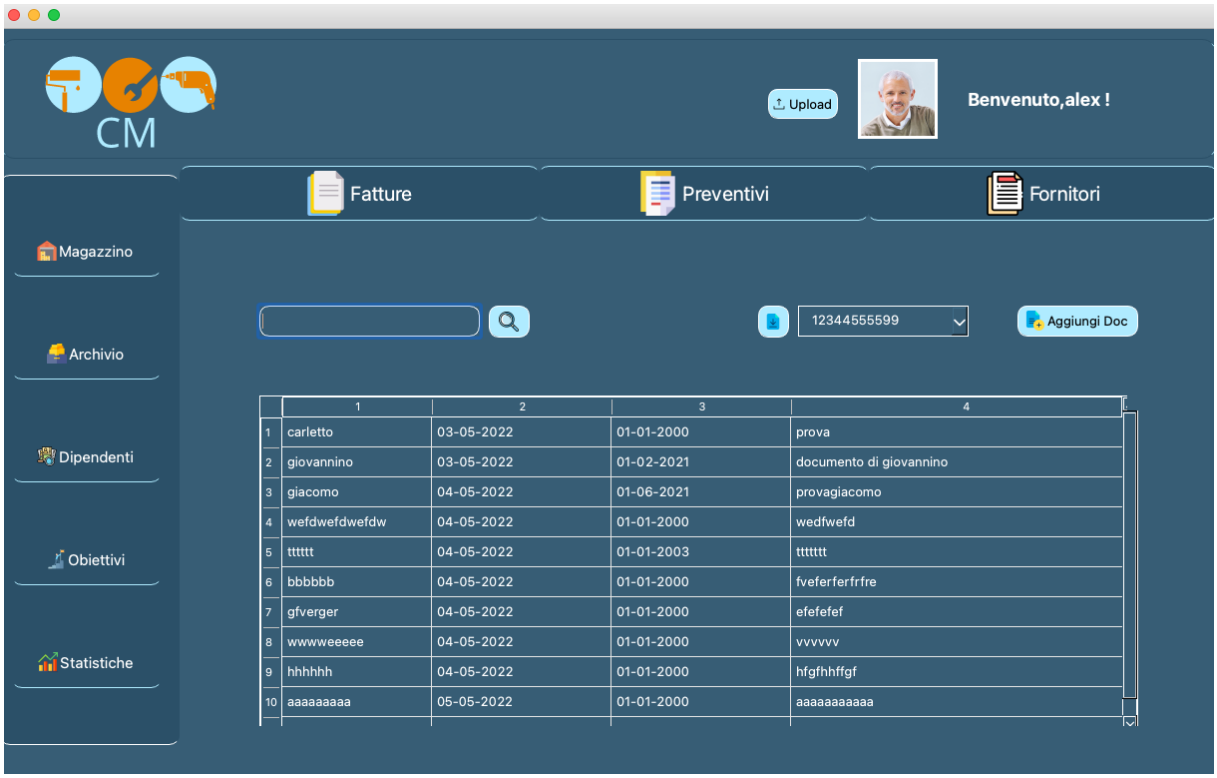
3.6.2 Schermate UI



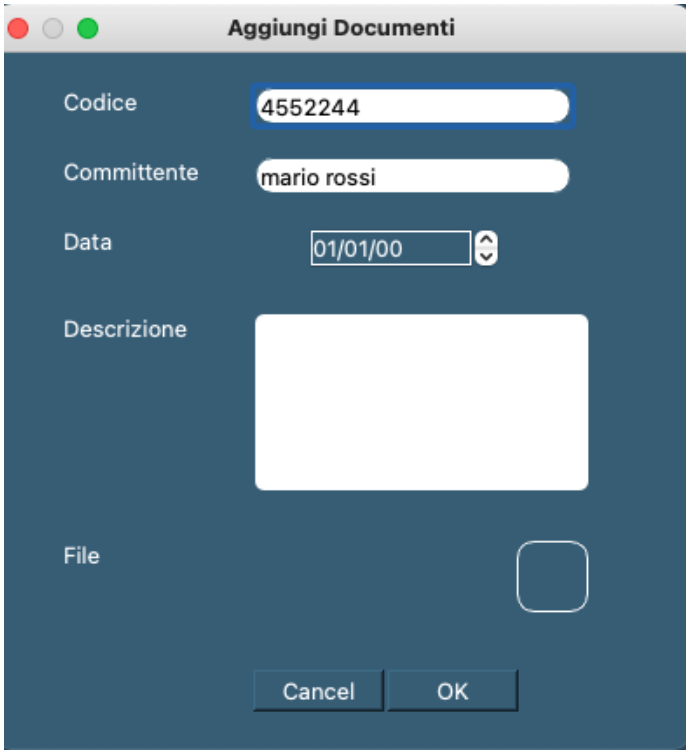
Gestione magazzino



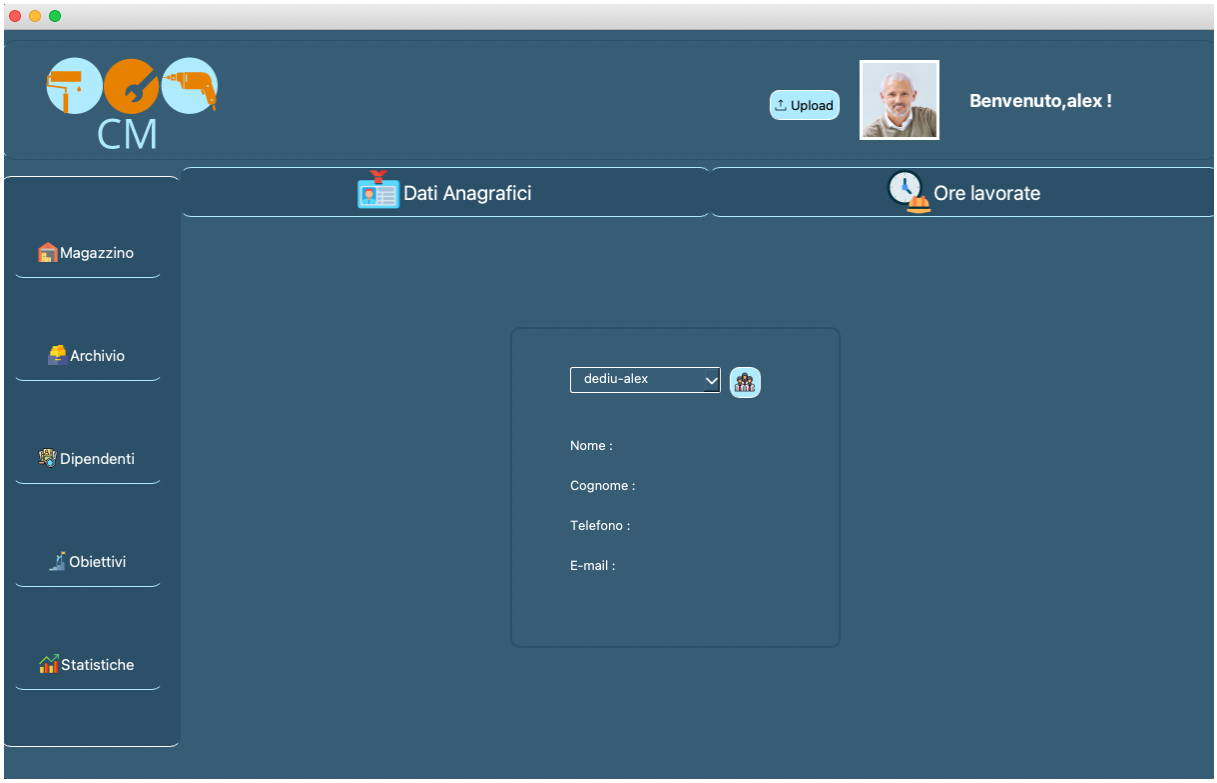
Dialog aggiungi prodotto



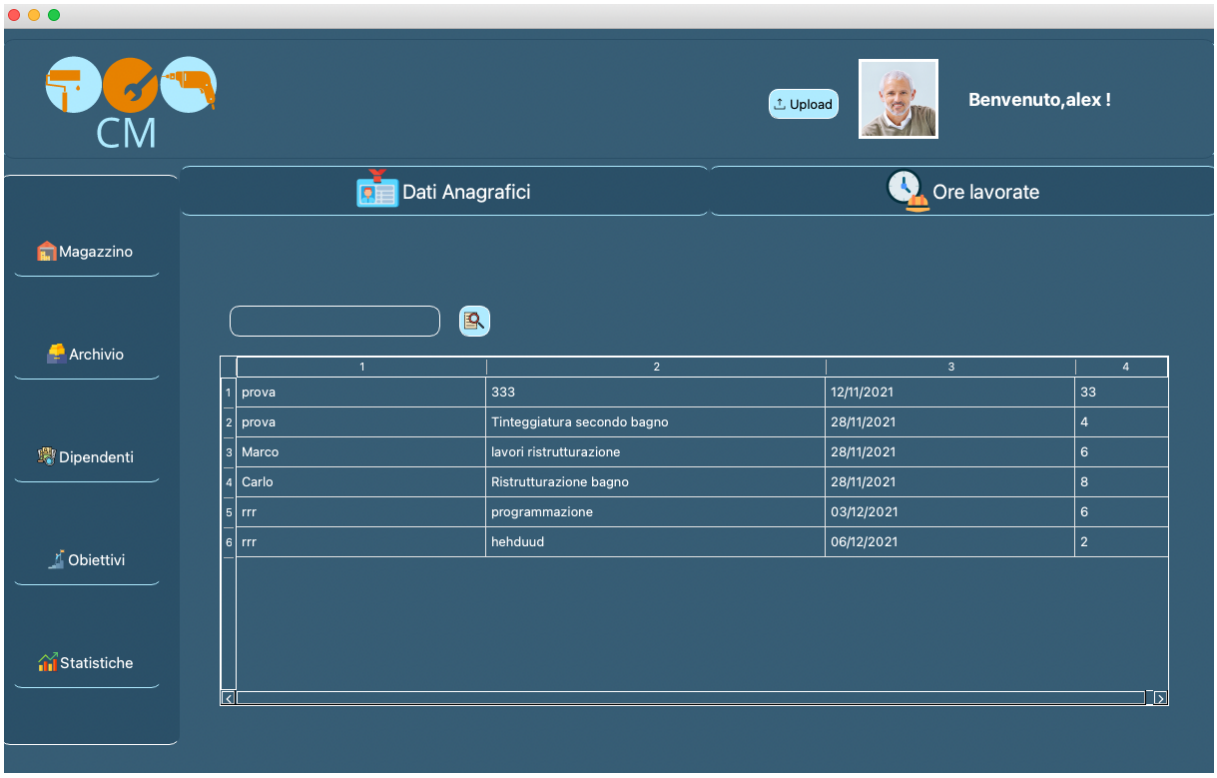
Gestione documenti



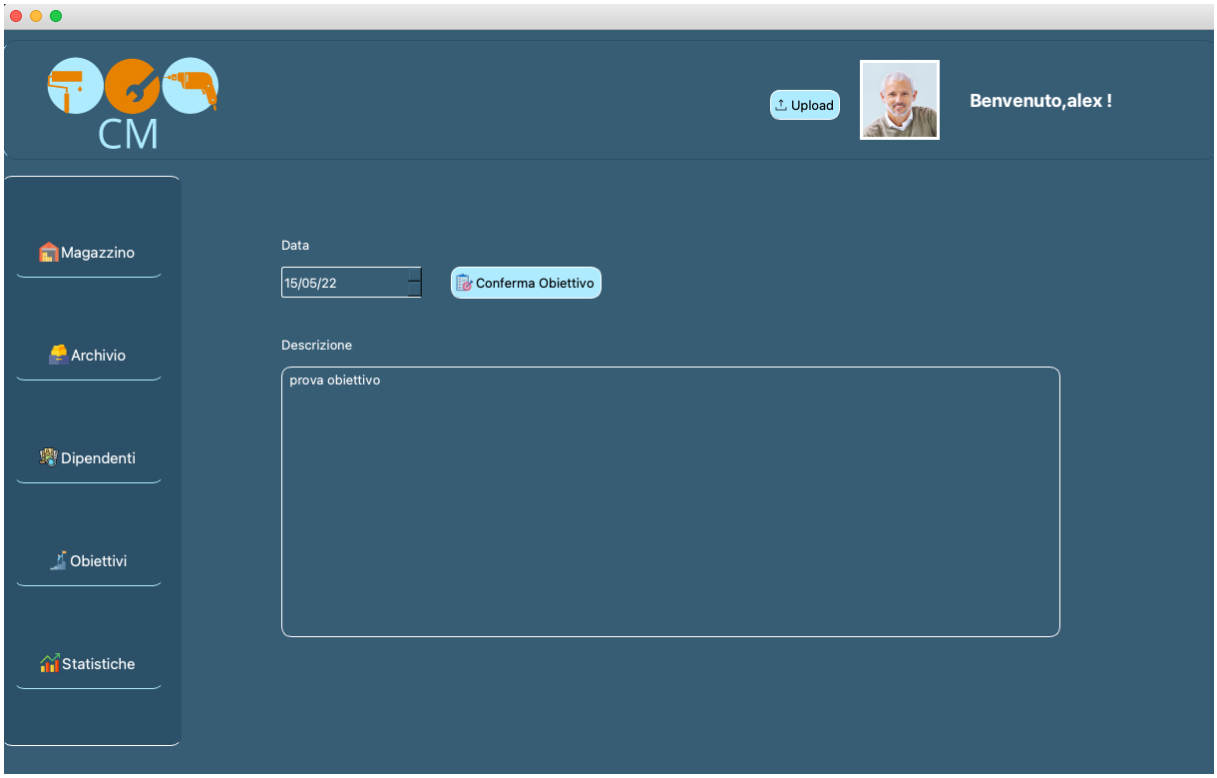
Dialog aggiungi documento



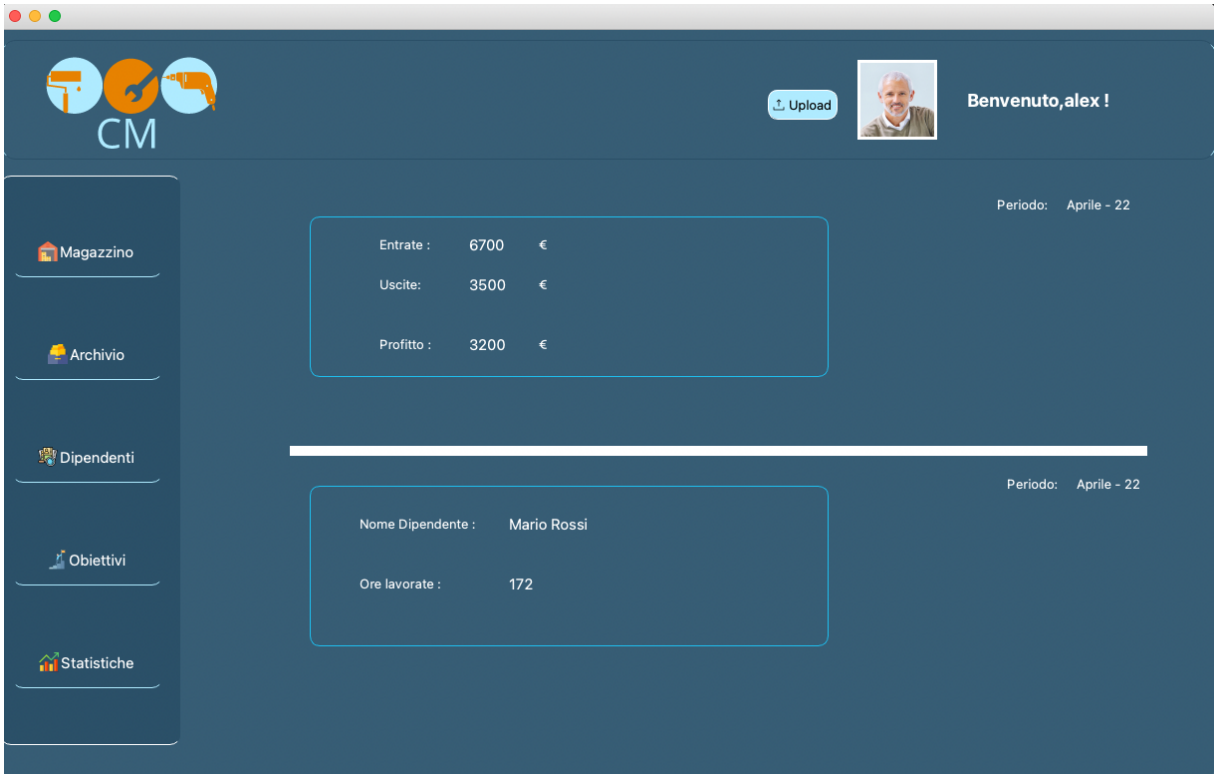
Gestione dati dipendenti



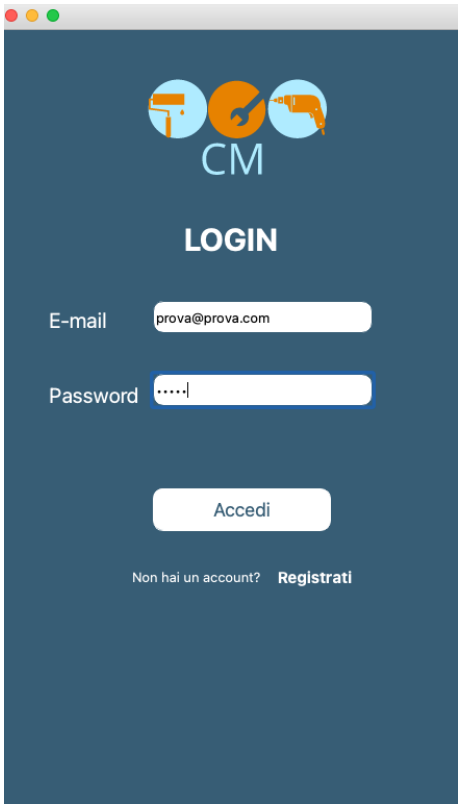
Gestione ore di lavoro dipendenti



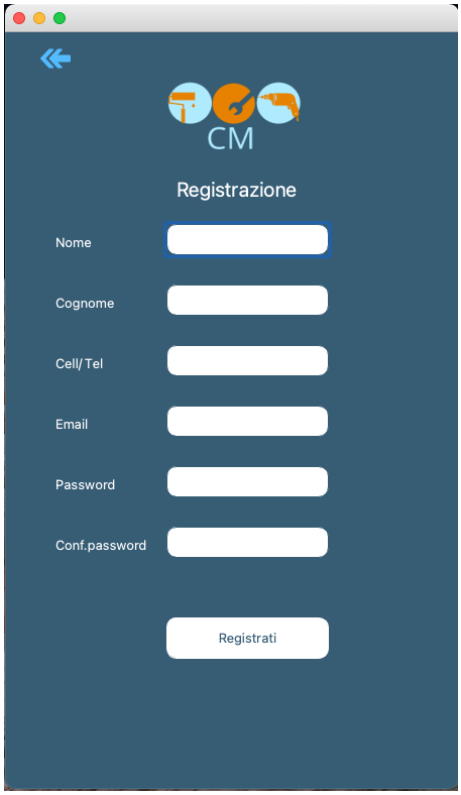
Gestione obiettivo



Statistiche



Login



Registrazione

3.7 Database

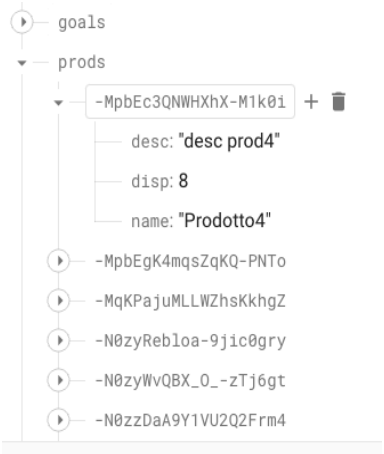
Per la persistenza dei dati e dei file è stata utilizzata la piattaforma Firebase di Google che offre due diversi database e uno Storage.

3.7.1 Pyrebase

Per utilizzare Firebase con il linguaggio Python è stata utilizzata una libreria presente su Github che fa da Wrapper per le API di Firebase. Questa libreria permette di inserire e manipolare dati nei diversi database con maggior facilità rispetto all utilizzo diretto di Firebase. Per il salvataggio dei dati e dei file sono stati utilizzati:

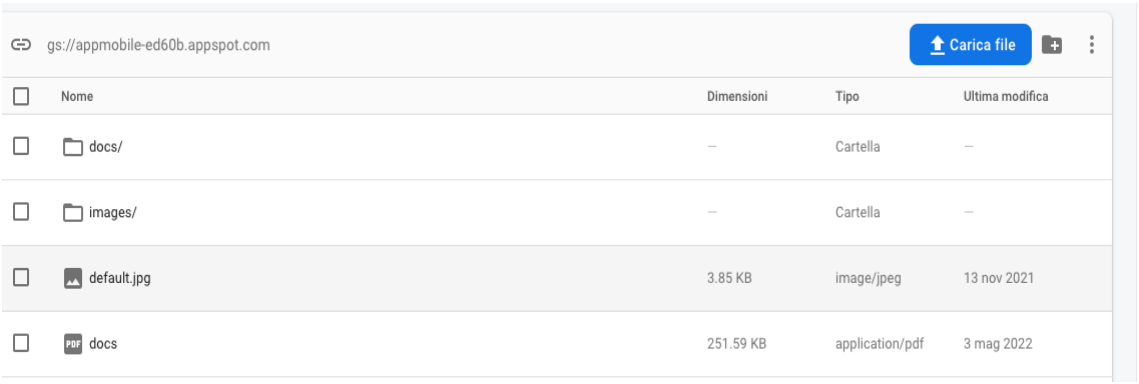
- *Realtime Database*: è un database NoSQL presente nel cloud che consente di archiviare e sincronizzare i dati in tempo reale.
- *Storage*: è uno store presente nel cloud per la memorizzazione di file e immagini.

Nel Realtime database sono presenti tutti i dati visualizzabili nel software. I Dati sono suddivisi in quattro macro-parti: Prods, Docs, Users, Goals. In ogni gruppo di dati sono presenti ramificazioni che indicano il singolo elemento. Questo, a sua volta, contiene altre ramificazioni che rappresentano i dettagli dell'elemento.



Elemento presente nel database

All'interno di Storage sono presenti tutti i documenti .PDF e tutte le immagini di profilo .PNG dell'utente. I file utilizzati sono suddivisi in due directory principali : Images e Docs.



Elemento presente nello storage

Per la manipolazione dei dati nel Database sono stati utilizzati i seguenti metodi:

- *database()*: crea una istanza del database sulla quale applicare i metodi per la manipolare i dati nel database.
- *child()*: consente di ottenere i dati di un determinato elemento, il nome dell'elemento da selezionare viene passato come parametro della funzione.
- *push()*: consente di inserire un elemento all'interno del database, come parametro della funzione viene passato un model contenente i dati dell'elemento da inserire. Il metodo deve essere utilizzato su una funzione child().
- *remove()*: consente di rimuovere un elemento dal database. Il metodo deve essere utilizzato su una funzione child()
- *get()*: consente di ottenere tutti gli elementi del database sotto forma di JSON.Il metodo deve essere utilizzato su una funzione child()
- *each()*: consenti di effettuare un ciclo foreach su un set di elementi. Il metodo devere essere applicato dopo una fuzione get().

- *val()*: consente di ottenere i dati di un elemento del db. Il metodo può essere applicato all'interno di ciclo foreach, inoltre inserendo un parametro tra parentesi quadre è possibile ottenere uno specifico dato di un elemento.

Per lo Storage sono state utilizzate diverse funzioni come :

- *storage()*: crea una istanza dello storage sulla quale applicare i metodi seguenti.
- *child()*: consente di ottenere un file presente nello storage, inoltre è possibile inserire un path per ottenere un documento interno ad una directory.
- *put()*: consente di inserire un file all'interno dello storage, il nome del file deve essere inserito come parametro.
- *download()*: consente di scaricare un file dallo storage, il nome del file deve essere inserito come parametro.

La libreria Pyrebase permette anche di gestire l'autenticazione mettendo a disposizione metodi per la memorizzazione delle credenziali e per il controllo sull'accesso.Inoltre, in mancanza di credenziali, è possibile effettuare la registrazione.
dopo la creazione di una istanza Firebase utilizzando il metodo auth(), possono essere utilizzati i seguenti metodi:

- *signinwithemailandpassword()*: inserendo e-mail e password come parametri della funzione, il metodo effettuerà un controllo sulle credenziali memorizzate sulla piattaforma per consentire o negare l'accesso all'area riservata.
- *createuserwithemailandpassword()*: metodo utilizzato per la registrazione di un nuovo utente, inserendo le credenziali come parametro è possibile memorizzare email e password per effettura l'accesso.

4 Sviluppo App Android

Poichè il software si pone come obiettivo quello di facilitare le attività delle imprese, è stata implementata anche un'applicazione per telefono.
L'applicazione è stata sviluppata per sistemi operativi Android (applicazione nativa) ed è dotata di una interfaccia utente intuitiva, in modo da facilitarne l'utilizzo.

4.1 Linguaggio di Programmazione ed Ambiente di Sviluppo

Questa estensione del software è una Applicazione Nativa per sistemi operativi Android; nelle prossime sezioni verranno descritti sinteticamente il linguaggio di programmazione utilizzato e l'ambiente di sviluppo su cui è stata implementata l'app.

4.1.1 Kotlin

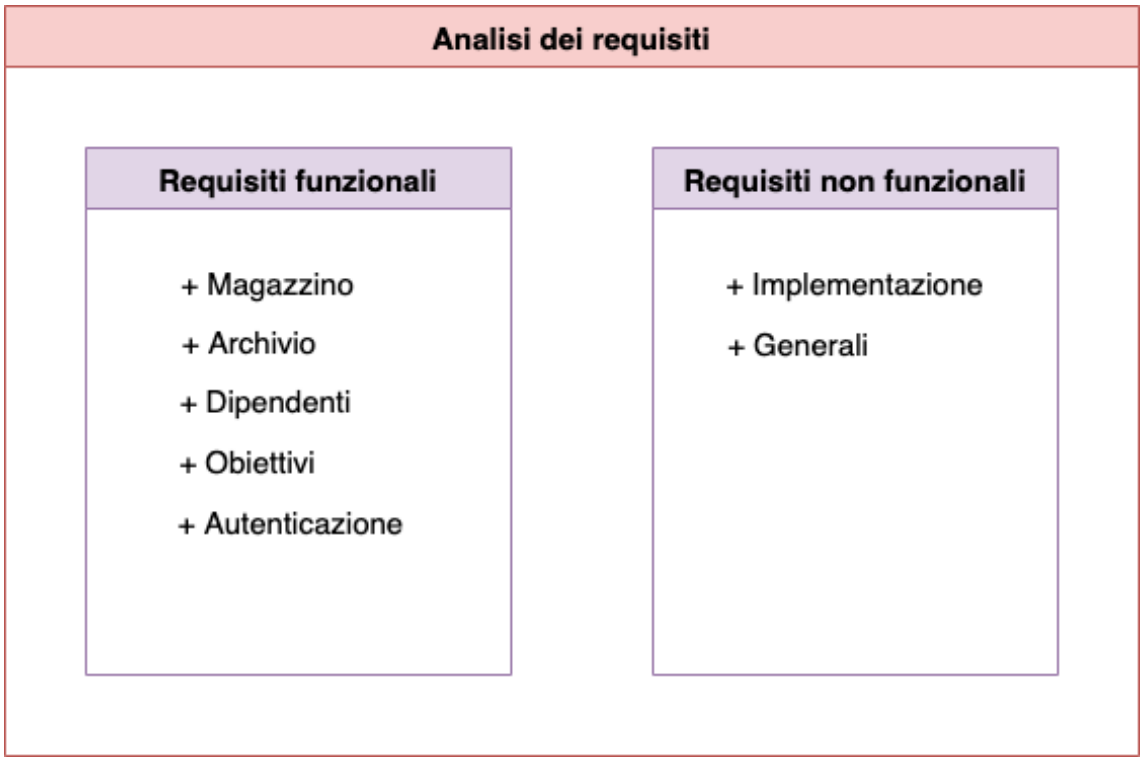
Kotlin è un linguaggio di programmazione open source sviluppato intorno al 2012 dall'azienda JetBrains. Deriva da un insieme di linguaggi di programmazione gia esistenti ed in particolere da Java. Infatti Kotlin si basa sulla JVM (Java Virtual Machine) ed è pienamente operativo con l'intero ecosistema Java (linguaggio principale utilizzato in Android Studio prima di Kotlin). Oltre ad essere general purpose e multi- paradigma, è anche molto sintetico e potente.

4.1.2Android Studio

Android Studio è il principale IDE (Integrated Development Environment) per lo sviluppo di Applicazioni Android, è completamente gratuito e facilita lo sviluppo e la progettazione dei programmi per il sistema operativo omonimo. Consente di sviluppare codice in due linguaggi di programmazione, Java e Kotlin.
Oltre alle principali funzionalità offerte dagli IDE, è possibile anche installare ed utilizzare emulatori per visualizzare l'applicazione durante lo sviluppo.
Come strumento principale interno ad Android Studio citiamo Gradle, che offre un ottimo sistema di build automation. Inoltre citiamo le quattro principali componenti di Android utilizzate per la totale integrazione dell'applicazione con l'ecosistema Android: Activity, Service, Content Provider e BroadcastReceiver.

4.2 Analisi dei Requisiti

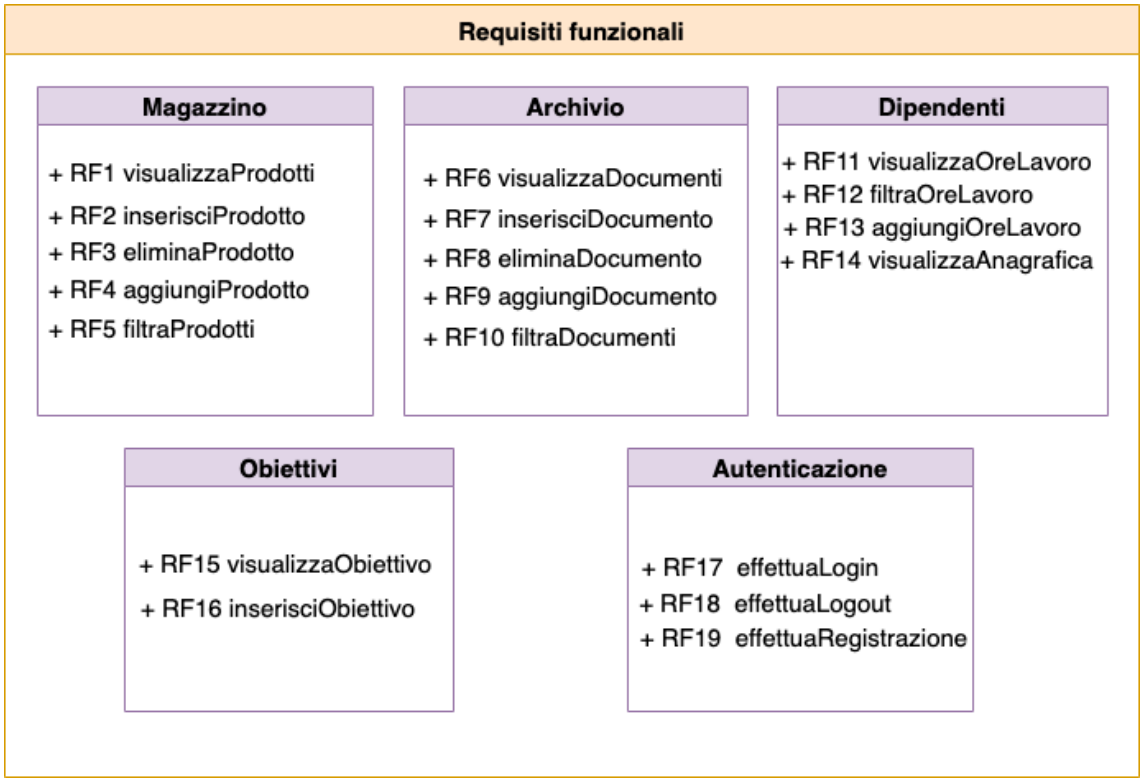
Per la descrizione dell'analisi dei requisiti sono stati descritti i requisiti funzionali, i requisiti non funzionali e l'architettura del progetto.



Analisi dei Requisiti

4.2.1 Requisiti Funzionali

di seguito vengono riportati e descritti i requisiti funzionali dell'applicazione. Ciascuna funzione è descritta nella tabella dei requisiti.



Requisiti Funzionali

	Requisiti	Descrizione
RF1	visualizzaProdotti	l'applicazione dovrà gestire la visualizzazione di tutti i prodotti
RF2	eliminaProdotto	l'applicazione dovrà gestire l'eliminazione di un prodotto selezionato
RF3	inserisciProdotto	l'applicazione dovrà gestire l'inserimento dei dati di un prodotto
RF4	filtraProdotti	l'applicazione dovrà gestire la ricerca di uno o più prodotti. In particolare dovrà filtrare i prodotti per nome
RF5	visualizzaDocumenti	l'applicazione dovrà gestire la visualizzazione di tutti i documenti
RF6	eliminaDocumenti	l'applicazione dovrà gestire l'eliminazione di un documento selezionato
RF7	inserisciDocumento	l'applicazione dovrà gestire l'inserimento degli estremi di un documento
RF8	filtraDocumento	l'applicazione dovrà gestire la ricerca di uno o più documenti. In particolare dovrà filtrare i documenti per nome
RF9	visualizzaOreLavoro	l'applicazione dovrà gestire la visualizzazione di tutte le ore lavorate dai dipendenti
RF10	filtraOreLavoro	l'applicazione dovrà gestire la ricerca delle ore di lavoro. In particolare dovrà filtrare i dati per nome del dipendente
RF11	visualizzaAnagrafica	l'applicazione dovrà gestire la visualizzazione dei dati anagrafici dei dipendenti
RF12	visualizzaObiettivo	l'applicazione dovrà gestire la visualizzazione dell'obiettivo giornaliero
RF13	inserisciObiettivo	l'applicazione dovrà gestire l'inserimento dell'obiettivo giornaliero
RF14	effettuaLogin	l'applicazione dovrà gestire l'accesso degli utenti all'area riservata
RF15	effettuaLogout	l'applicazione dovrà gestire il logout dalla propria area riservata
RF16	effettuaRegistrazione	l'applicazione dovrà gestire la registrazione di un nuovo utente.

Tabella dei Requisiti Funzionali

La schermata iniziale dell'applicazione è la pagina di Login dalla quale è possibile :

- Accedere alla propria area riservata con credenziali
- Registrarsi inserendo email, password ed altri dati personali per creare una utenza con cui accedere.

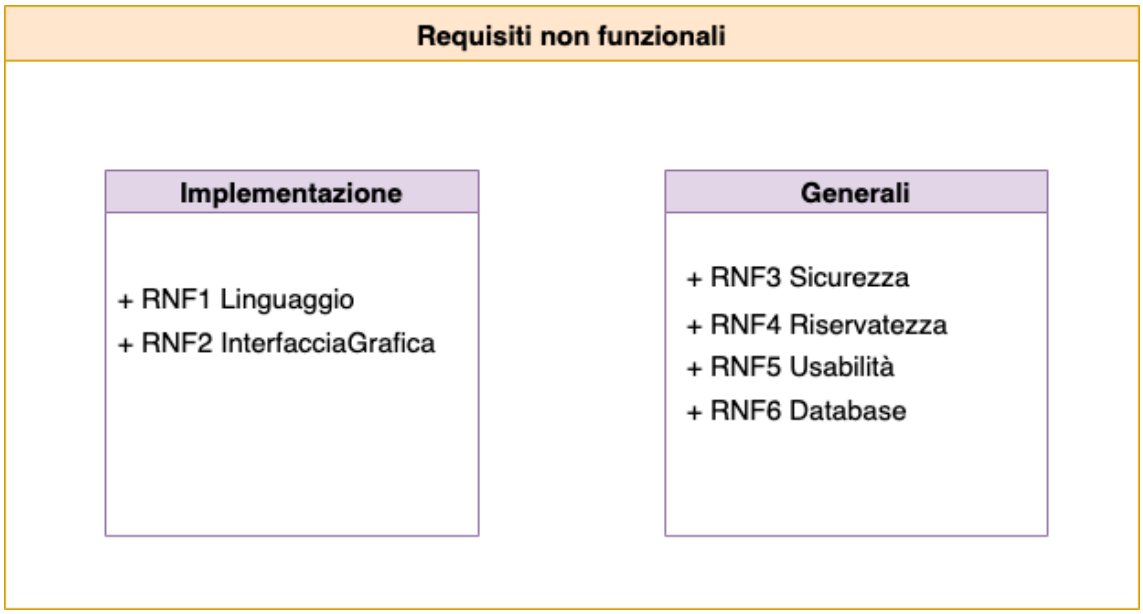
Ad ogni utenza è assegnato un parametro che distingue le credenziali dell'Amministratore da quelle del Dipendente.
Quindi, effettuando l'accesso, l'utente viene reindirizzato in una delle due seguenti dashboard:

- *Area Amministratore*: in questa sezione sono presenti tutte le funzionalità a cui può accedere il datore di lavoro, in particolare le sotto-sezioni sono :
  - *Magazzino*: il magazzino può essere gestito attraverso l'inserimento, l'eliminazione e la ricerca di uno o più prodotti.
  - *Documenti*: all'interno di questa sotto-sezione è possibile inserire o eliminare un documento ed effettuare una ricerca per nome dei documenti.
  - *Dipendenti*: in questa sotto-sezione è possibile visualizzare l'anagrafica e le ore lavorate di ogni dipendente, è possibile inoltre effetturare una ricerca per nome del dipendente.
  - *Obiettivi*: in questa sotto-sezione è possibile inserire un obiettivo giornaliero da comunicare ai dipendenti.
- *Area Dipendente*: in questa dashboard il dipendente può accedere a tre sotto-sezioni :
  - *Magazzino*: in questa sotto-sezione è possibile visualizzare tutti i prodotti presenti nel magazzino ed i dati ad essi associati.Inoltre è possibile effetturare una ricerca filtrando i prodotti per nome.
  - *Orari*: in questa sotto-sezione è possibile inserire l'orario di lavoro ed una breve descrizione del lavoro svolto durante la giornata.

– *Obiettivi*: in questa sotto-sezione è possibile visualizzare la descrizione dell'obiettivo inserito dal datore di lavoro.

4.2.2 Requisiti Non Funzionali

In questa sezione vengono riportati ed in seguito descritti i requisiti non funzionali dell'applicazione.



Requisiti Non Funzionali

	Requisiti	Descrizione
RNF1	Linguaggio	l'applicazione dovrà essere realizzata con il linguaggio di programmazione Kotlin
RNF2	InterfacciaGrafica	l'applicazione dovrà essere dotato di interfaccia grafica
RNF3	Sicurezza	l'applicazione dovrà funzionare senza recare danno a persone o cose
RNF4	Riservatezza	l'applicazione dovrà tutelare le informazioni in esso contenute
RNF5	Usabilità	l'applicazione dovrà essere intuitivo e facile da usare anche per utente poco esperti
RNF6	Database	l'applicazione dovrà essere dotato di database per la persistenza dei dati

Tabella dei Requisiti Non Funzionali

4.2.3 Architettura

L'applicazione sfrutta Activity e Fragment secondo un'architettura modulare. L'intero progetto è stato suddiviso in cinque package:

- *Models*: all'interno di questo package sono presenti tutte le classi Model utilizzate per effettuare il mapping dei dati e per l'inserimento degli stessi nelle Card delle RecycleView.
- *Adapters*: all'interno di questo package sono presenti le classi utilizzate per gestire l'inserimento, eliminazione e la visualizzazione dei dati all'interno dei cataloghi.
- *Fragment*: all'interno di questo package sono presenti i Fragment utilizzati nell'area utente per gestire i dipendenti, magazzino e l'obiettivo.
- *Activity*: all'interno di questo package sono presenti tutte le Activity utilizzate per sviluppare l'applicazione.
- *Auth*: all' interno di questo package sono presenti le classi per l'Autenticazione di un utente e per la Registrazione di una nuova utenza.

Per la realizzazione dei progetto sono state utilizzate diverse risorse inserite all'interno delle seguenti sei sezioni:

- *Drawable*: in questa sezione sono state inserite tutte le icone e le forme geometriche utilizzate per il design dell'applicazione.
- *Layout*: in questa sezione sono presenti tutti i layout dell'applicazione; per ogni layout è stato implementato anche il landscape per rendere l'applicazione responsive.
- *Menu*: in questa sezione è presente il menù utilizzato per visualizzare i fragment all'interno dell'area riservata.
- *MipMap*: in questa sezione sono state inserite le immagini predefinite Andoid e il logo utilizzato nella home page dell'applicazione.
- *Navigation*: in questa sezione sono presenti i file necessari per la gestione e la visualizzazione dei fragment.

4.3 Database

Come per il software gestionale, per la persistenza dei dati e dei file è stata utilizzata la piattaforma Firebase ed, in particolare, le componenti del database RealTime e dello Storage. Durante lo sviluppo dell'applicazione non è stata utilizzata alcuna libreria per il collegamento alle API di Firebase. Le funzioni e gli strumenti offerti dalla piattaforma per lo sviluppo Android sono numerosi. Di seguito saranno riportate alcune funzioni importanti utilizzate per la manipolazione dei dati nel database:

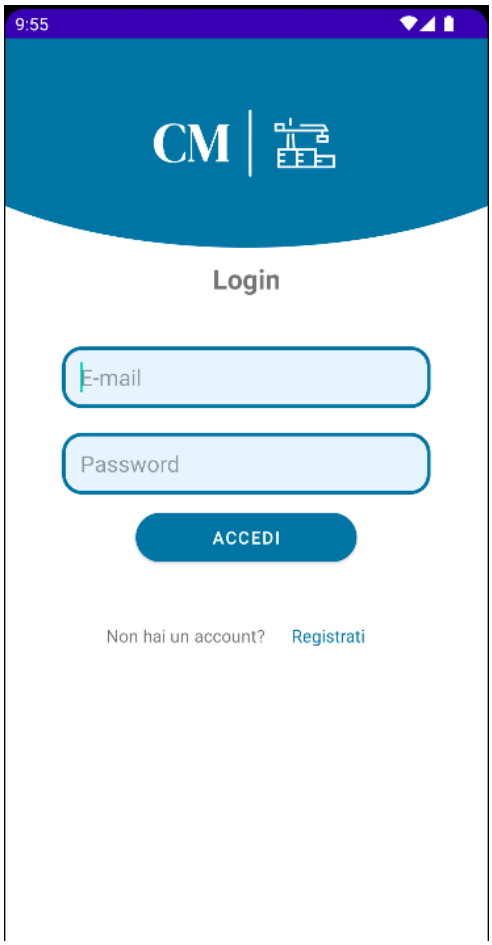
- *createUserWithEmailAndPassword()*: metodo utilizzato per la creazione di una nuova utenza per l'accesso all'area riservata; riceve in ingresso due parametri che rappresentano le nuove credenziali di accesso.
- *signInWithEmailAndPassword()*: metodo utilizzato per effetturare l'accesso all'area utente; riceve in ingresso due parametri che rappresentano le credenziali di accesso.
- *getInstance().getCurrentUser()*: serie di metodi utilizzati per ottenere i dati dell'utente attualmente presenti nella propria area riservata.
- *addValueEventListener()*: metodo utilizzato per effettuare l'aggiornamento dei dati ogni volta che si presenta un cambiamento degli stessi.
- *onDataChange()*: metodo utilizzatto per leggere uno snapshot statico dei contenuti in un determinato percorso; spesso viene adottato all'interno del metodo precedente; esso riceve in ingresso come parametro uno snapshot.
- *addOnSuccessListener()*: metodo utilizzato per segnalare l'esecuzione di un metodo che è andato a buon fine.

4.4 Interfaccia grafica

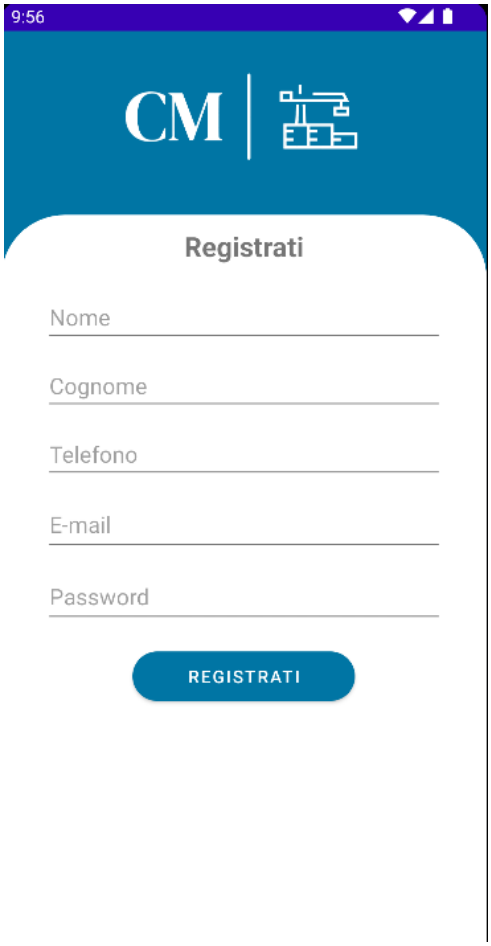
In questa sezione sono riportate le schermate di tutte le pagine presenti nell'applicazione. Esse sono suddivise in autenticazione, area amministratore e area dipendenti.

4.4.1 Schermate Autenticazione

Di seguito sono riportare le schermate delle pagine per effetturare login e registrazione. La pagina di login è la prima pagina con cui l'utente interagisce.



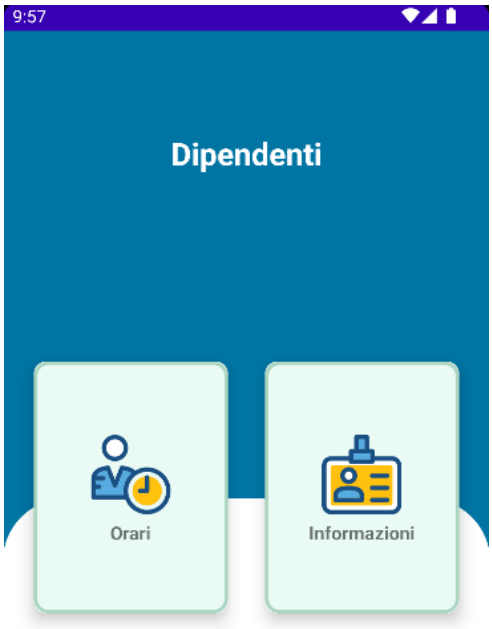
schermata di login



schermata di registrazione

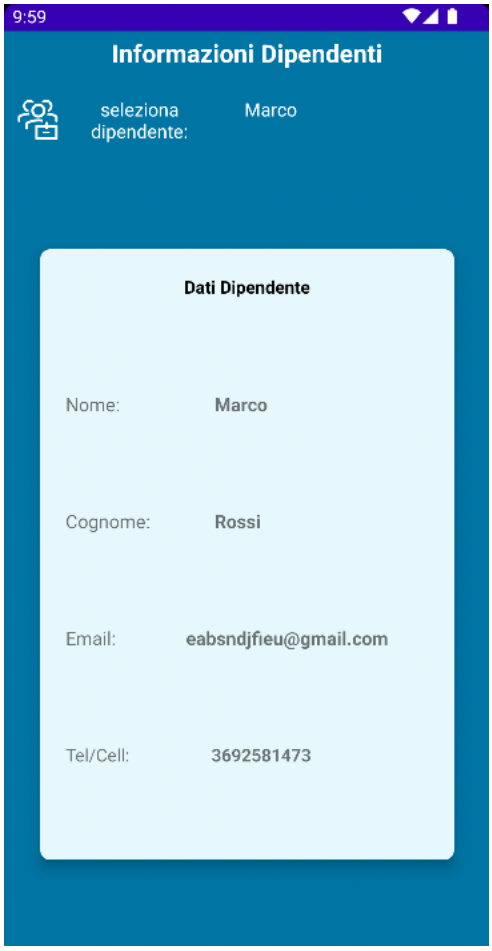
4.4.2 Schermate Area Amministratore

Di seguito sono riportare le schermate dell'area utente riservata all'amministratore.

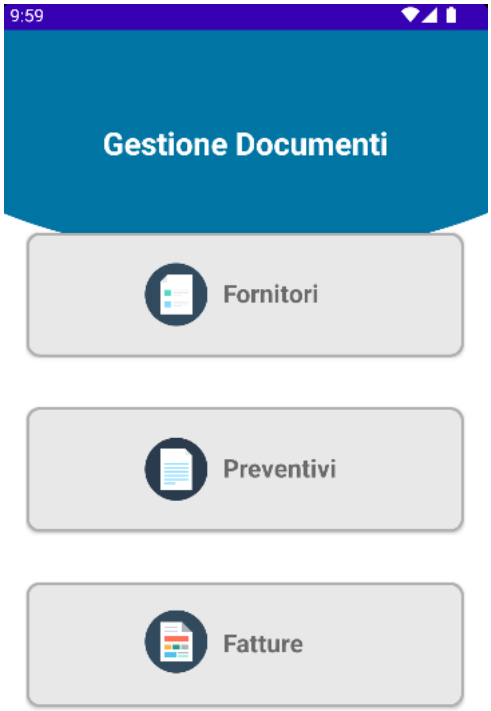


schermata gestione dipendenti

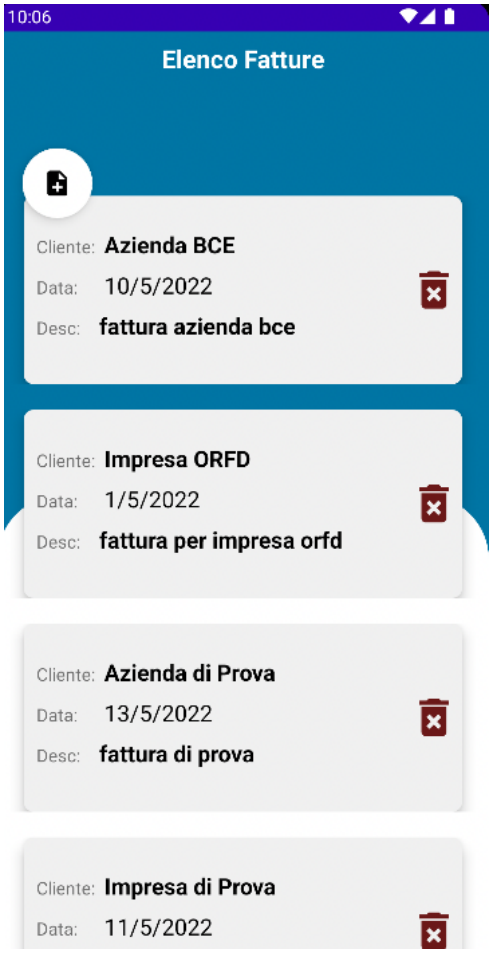
schermata ore di lavoro



schermata anagrafica

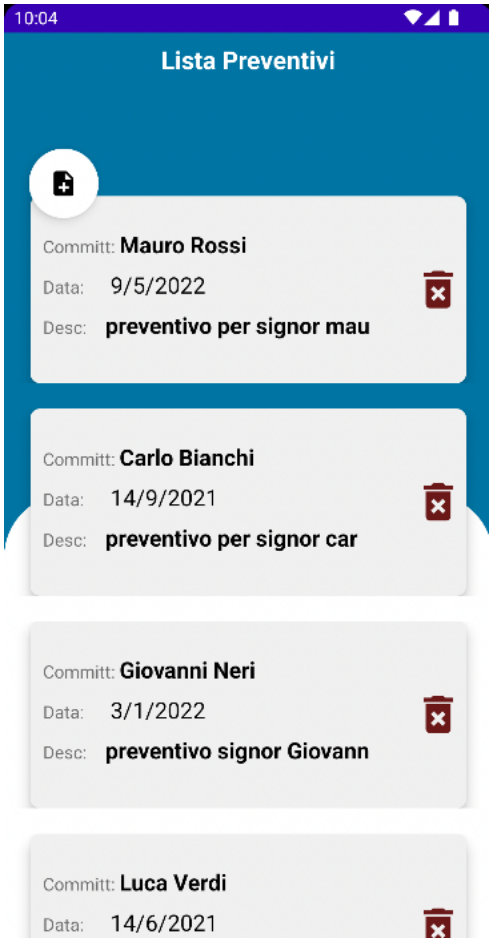


schermata gestione documenti

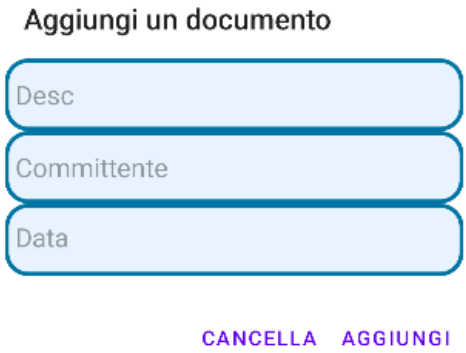


schermata fatture

schermata fornitori



schermata preventivi



schermata aggiungi documento

schermata aggiungi obiettivo

4.4.3 Schermate Area Dipendenti

Di seguito sono riportare le schermate dell'area riservata ai dipendenti.

schermata inserisci ore di lavoro

schermata visualizza obiettivo

4.5 Diagrammi

4.5.1 Diagramma casi d'uso

Nel seguente diagramma sono presenti i possibili casi d'uso dell'applicazione.

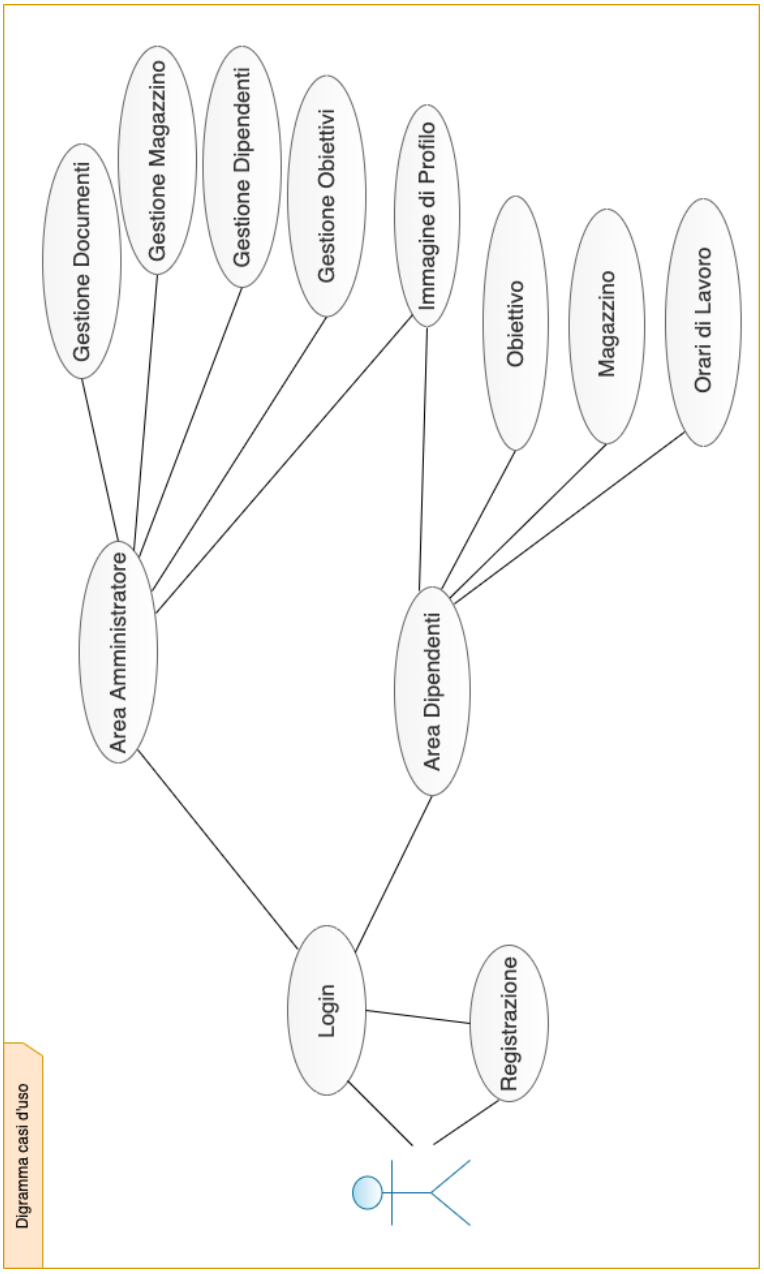
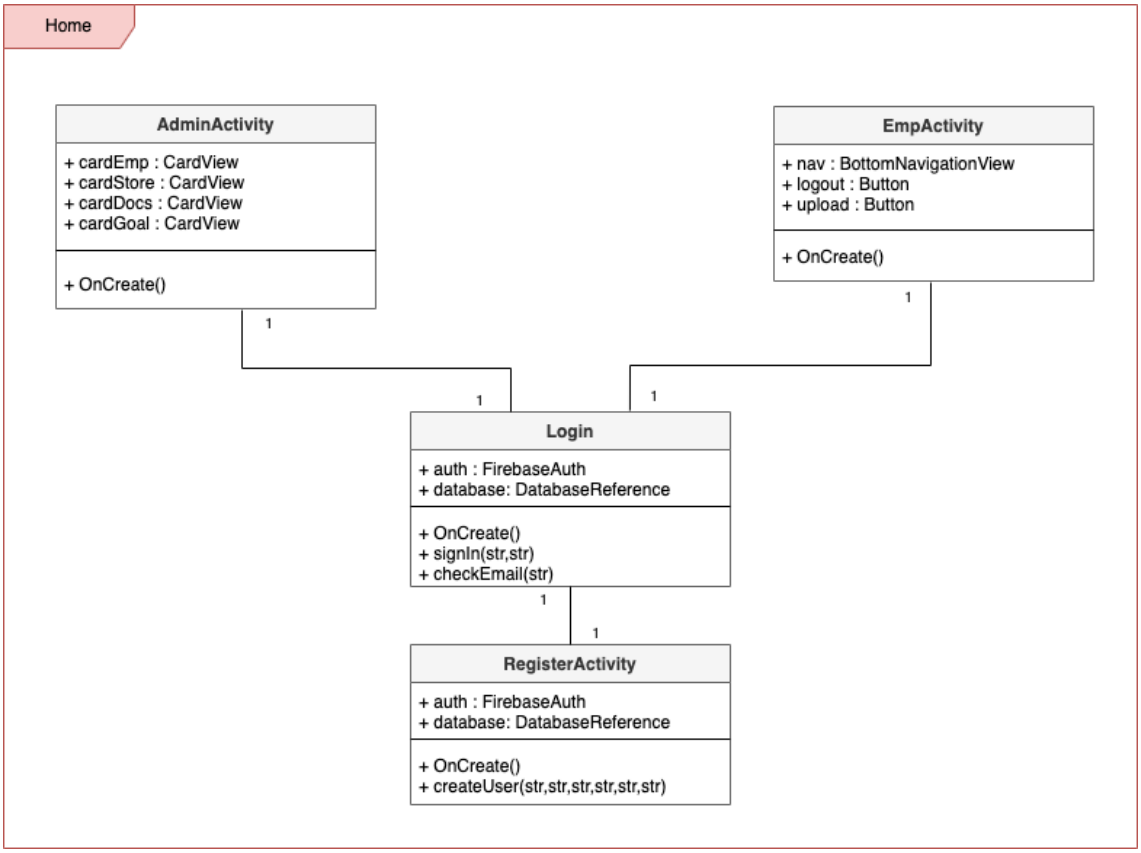


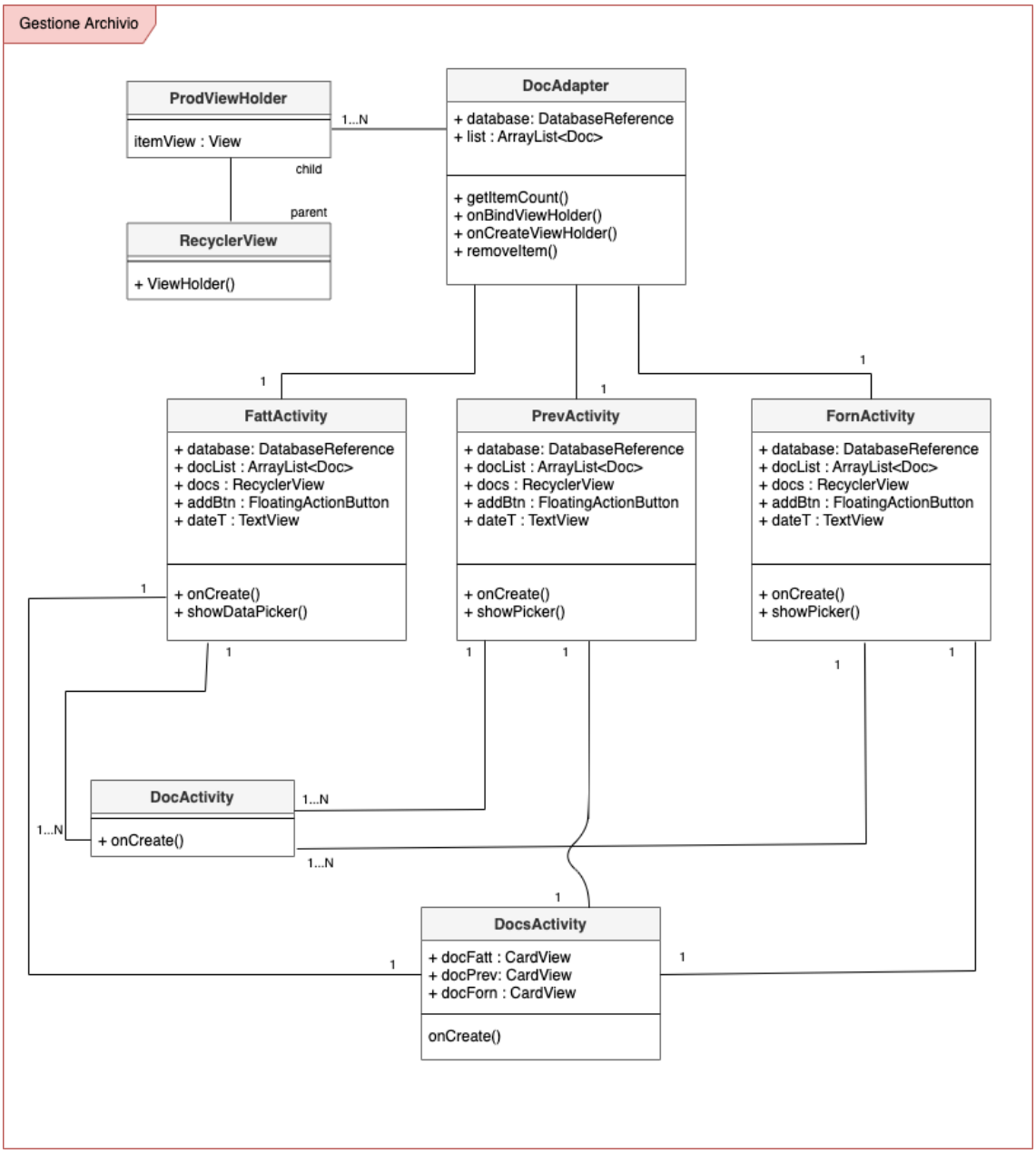
Diagramma dei casi d'uso - App Android

4.5.2 Diagramma delle classi

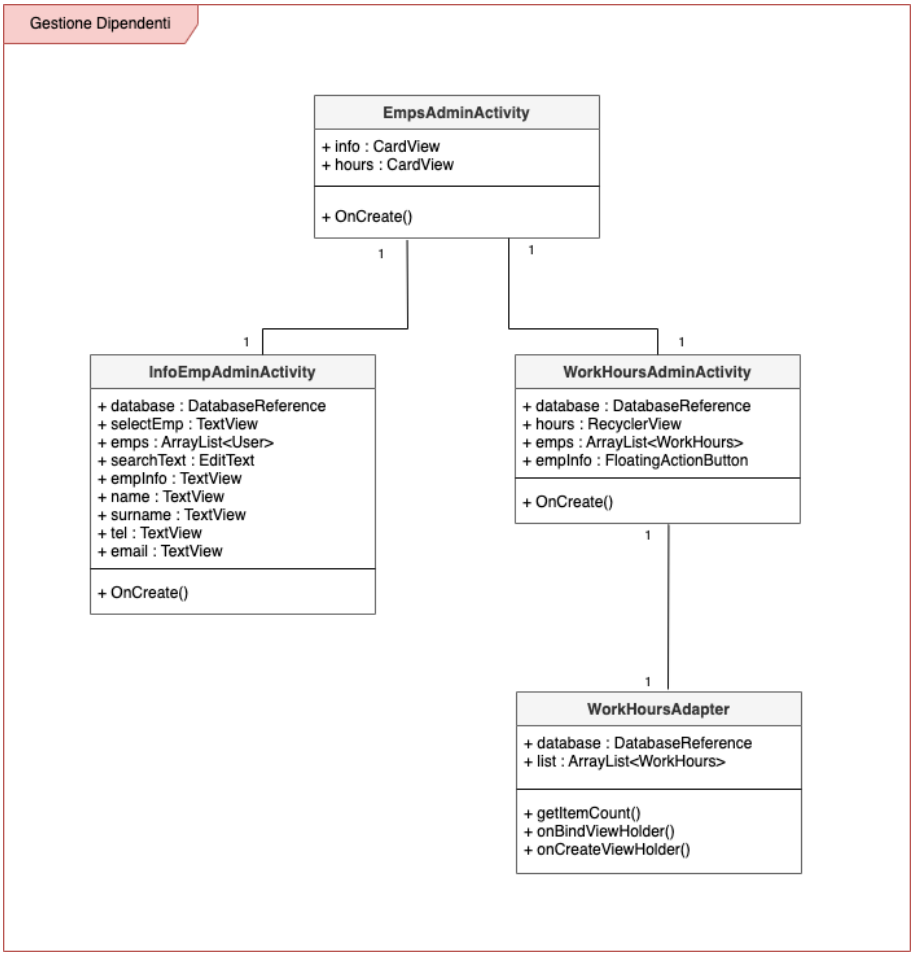
Di seguito vengono riportati i diagrammi di tutte le classi con cui è stata progettata l'applicazione.
Ogni classe rappresentata è suddivisa in tre parti: Nome della classe, proprietà e attributi.



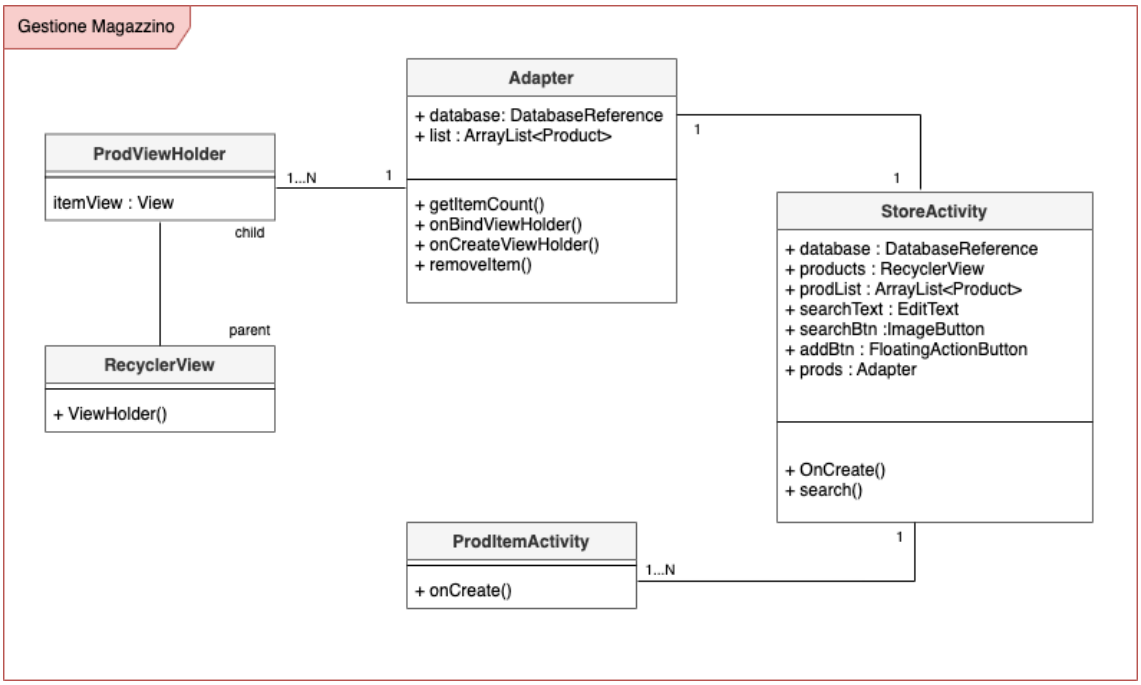
Autenticazione - Homepage



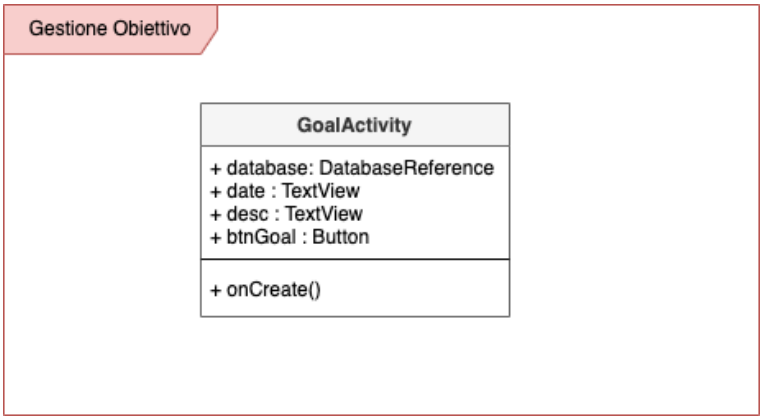
Gestione Archivio



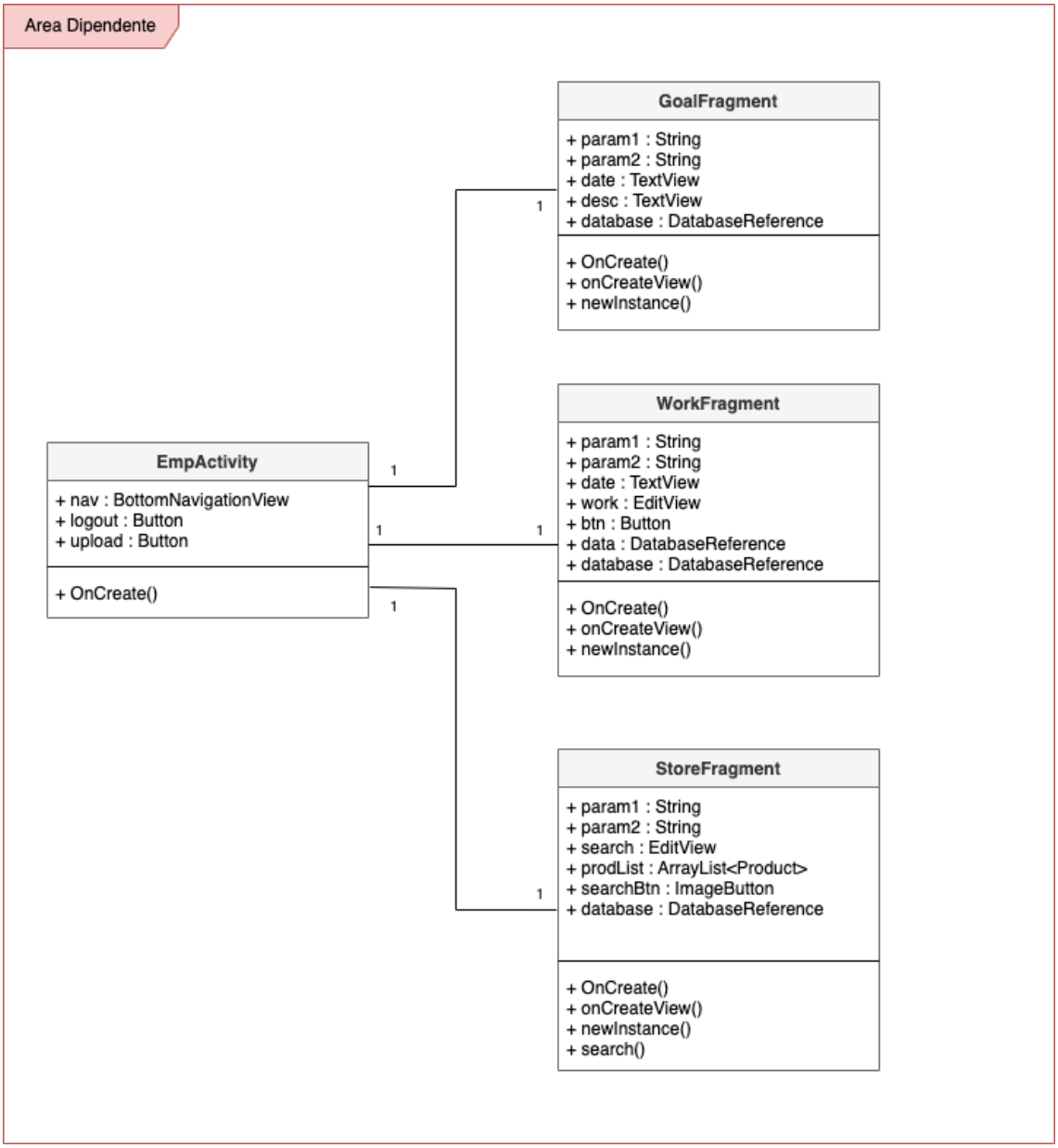
Gestione Dipendenti



Gestione Magazzino



Gestione Obiettivo



Homepage - Dipendente

5 Conclusioni

Nei precedenti capitoli è stata descritta la progettazione e la realizzazione di un software gestionale in grado di semplificare la gestione delle principali attività di una azienda.
Dopo una descrizione accurata dei requisiti e degli strumenti utilizzati per lo sviluppo del gestionale è stata descritta la progettazione di una applicazione mobile associata a tale software, in modo da facilitare ulteriormente la gestione e la memorizzazione dei dati.

5.1 Punti di forza e di debolezza

In questa sezione saranno messi in evidenza i punti di forza e di debolezza del software e dell'applicazione mobile. Tra i punti di forza principali ci sono :

- la completezza con cui il lavoro è stato realizzato e documentato, poichè, in primo luogo, consente di avere una visione completa di ciò che è stato fatto e, in secondo luogo, ne facilita una potenziale modifica futura.
- la semplicità con cui l'utilizzatore può usufruire dei servizi offerti, le interfaccie sono state realizzate in modo da rendere intuitivo l'utilizzo delle componenti.
- la sicurezza nella memorizzazione dei dati sensibili, l'utilizzo di una piattaforma esterna e delle funzionalità offerte garantisce la sicurezza durante l'inserimento di dati.

Alcuni aspetti negativi riscontrati anche durante la realizzazione sono:

- la mancanza di una area riservata per i dipendenti nel software gestionale che avrebbe reso completo il sistema.
- la mancanza di una suddivisione dei dipendenti in ruoli in modo da ottenere più livelli di autenticazione.

5.2 Sviluppi futuri

In questa sezione saranno messi in risalto i possibili sviluppi futuri, in particolare:

- la possibilità di realizzare ulteriori statistiche utilizzando dati reali e impostando grafici per facilitarne la comprensione.
- la possibilità di sviluppare una area riservata per i dipendenti anche sul software gestionale per desktop.
- la possibilità di implementare una applicazione cross-platform in modo da rendere completo il sistema.

Alcuni potenziali sviluppi futuri potrebbero nascere anche dalle esigenze delle imprese che necessitano di funzionalità specifiche, quindi gli sviluppi vanno sempre progettati e implementati in base alle aziende ed alle loro necessità.

Bibliografia

BOSCAINI, M. (2020), *Imparare a programmare in Python*, Apogeo.

BUTTU, M. (2014), *Programmare con Python, Guida completa*, Edizioni Lswr.

C. GHEZZI, D. M., M. JAZAYERI (2021), *Ingegneria del software, Fondamenti e principi*, Pearson.

GASSTON, P. (2011), *CSS3, Guida completa per lo sviluppatore*, Hoepli.

GIGLIOTTI, G. (2011), *HTML5 e CSS3*, Apogeo.

MOORE, A. D. (2019), *Mastering GUI Programming with Python*, Packt.

P. ATZENI, S. C. (2018), *Basi di dati*, McGraw-Hill.

P. DEITEL, H. D. (2020), *Introduzione a Python, Per l'informatica e la data science*, Pearson.

P. FOGGIA, M. V. (2011), *Algoritmi e strutture dati*, McGraw-Hill.

PRESSMAN, R. S. (2008), *Principi di Ingegneria del software*, McGraw-Hill Education.

SHVETS, A. (2019), *Dive into Design Patterns*, Ebook.

SOMMERVILLE, I. (2017), *Ingegneria del Software*, Pearson Education.

SUMMERFIELD, M. (2007), *Rapid GUI Programming with Python and Qt The Definitive Guide to PyQt Programming*, Prentice Hall.

Websites consulted

- Stack Overflow – www.stackoverflow.con
- Github: Where the world builds software – www.github.com
- HTML.it – www.html.it
- Python Documentation– www.docs.python.org
- QT – www.qt.io
- Real Python – www.realpython.com
- Python GUIs– www.pythonguis.com
- CSS: Cascading Style Sheets – www.developer.mozilla.org
- Wikipedia – www.wikipedia.org

Ringraziamenti

Il primo ringraziamento va a mia madre e a mio padre, che mi hanno dato la possibilità, per nulla scontata, di intraprendere e portare a termine questo percorso universitario, sostenendomi nei momenti di difficoltà.
Vorrei ringraziare il Prof. Domenico Ursino, che mi ha accompagnato nel lavoro di tesi seguendomi costantemente con estrema disponibilità. Grazie per le competenze che mi ha trasmesso e per i preziosi consigli.
Ringrazio di cuore la mia fidanzata Sissi per avermi sempre sopportato e supportato durante questa fantastica esperienza.
Ringrazio i miei colleghi e amici Michele, Diego e Gianmatteo per aver condiviso con me questa esperienza.Grazie per il supporto e per essere stati sempre incredibilmente pazienti.
Infine, vorrei ringraziare il mio amico,collega,coinquilino e compagno di avventure Lorenzo. Grazie per avermi contagiato con la tua infinita curiosità, grazie per avermi sempre aiutato nel momento del bisogno.