

Progetto di MODULO 2: Laboratorio di Sistemi Informativi
Anno Accademico 2024/2025

Sistema Informativo per l'Organizzazione
Postale Calabrese

Docente
prof. Francesco Parisi

Studenti
Umberto Frega 239527
Leonardo Napoli 234364

Link alla repository: <https://github.com/ZiOLEO/SistemiInformativi>

Contents

| | | |
|----------|---------------------------------------|-----------|
| 1 | Introduzione | 2 |
| 1.1 | Sede | 2 |
| 1.2 | Organizzativo | 2 |
| 1.3 | Posizione nel Mercato | 2 |
| 1.4 | Prospettive Future | 2 |
| 1.5 | Benefici Attesi | 2 |
| 1.6 | Funzionalità | 3 |
| 2 | Analisi dei Requisiti | 5 |
| 2.1 | Analisi dello scenario | 5 |
| 2.1.1 | Registrazione cliente | 6 |
| 2.1.2 | Invio corrispondenza | 9 |
| 2.1.3 | Attori e archivi | 11 |
| 2.2 | Specifica dei requisiti | 13 |
| 2.2.1 | Gestione clienti | 13 |
| 2.2.2 | Gestione recapiti | 16 |
| 3 | Progettazione | 21 |
| 3.1 | Modellazione della base di dati | 21 |
| 3.1.1 | Modello concettuale | 22 |
| 3.1.2 | Modello relazionale | 24 |
| 3.2 | Modellazione della logica di business | 26 |
| 3.3 | Modellazione delle interfacce | 28 |
| 4 | Implementazione | 31 |
| 4.1 | Base di dati | 31 |
| 4.2 | UI | 31 |
| 4.3 | Design Pattern | 31 |
| 4.3.1 | Model View Controller | 31 |
| 4.3.2 | Data Transfer Object | 31 |



1 Introduzione

L'Organizzazione Postale Calabrese (OPCal) affiliato a Poste Italiane, con sede legale a Cosenza e filiale a Rende(CS), consiste in un Ufficio atto alla spedizione e ricezione di corrispondenze.

1.1 Sede

La sede di Rende(CS) in via Marconi, 11 è una piccola struttura avente 3 sportelli e un magazzino. La sua clientela è composta prevalentemente da studenti della vicina Università della Calabria che effettuano operazioni di ricevimento pacchi. La fondazione dell'ufficio risale al 2017, pertanto l'attuale sistema informativo è datato.

1.2 Organizzativo

Il direttore dal 2020 è il sig.Lucio Dalla. L'ufficio è diviso in 3 sezioni, la sezione Sportello con 3 dipendenti è gestita dal sig.Francesco de Gregori, la sezione Magazzino con 2 dipendenti è gestita dal sig.Adriano Celentano e la sezione Recapiti è gestita dal sig.Francesco Guccini con 2 dipendenti al suo seguito. In totale l'organizzazione ammonta a 11 dipendenti.



Figure 1: Organigramma

1.3 Posizione nel Mercato

L'OPCal detiene ad oggi gran parte del palcoscenico postale cosentino, i competitor sono per la maggioranza servizi privati in rapida ascesa.

1.4 Prospettive Future

Nel breve termine l'obiettivo della OPCal rimane il mantenere le quote di mercato nell'area metropolitana cosentina con uno sguardo verso l'esterno, con la possibilità a medio/lungo termine di espandere il proprio mercato all'intera area calabrese, continuando a garantire una politica di serietà e velocità nel servizio e disponibilità del personale.

1.5 Benefici Attesi

L'implementazione del sistema all'interno dell'organizzazione aziendale porterà diversi benefici, come:



- Rinnovamento del sistema attuale;
- Centralizzazione delle funzionalità;
- Centralizzazione dei dati;
- Rimozione della dipendenza da documenti cartacei;
- Maggiore possibilità di estensione dell'organizzazione;
- Semplificazione e deburocratizzazione della user experience;
- Possibilità di avere un sistema pubblicitario più esteso ed efficiente tramite il sito web;

1.6 Funzionalità

Le macro-funzionalità fornite dal sistema informativo sono le seguenti:

1. Gestione dei Clienti

- Il sistema avrà la funzionalità di mantenere, organizzare e visualizzare le informazioni riguardanti i clienti, quali *spedizioni a carico*, *saldo corrente* e *pacchi in arrivo*.
- Ogni cliente avrà la possibilità di visualizzare i suoi dati, prenotarsi allo sportello e prenotare una spedizione tramite le interfacce offerte.
- L'utente potrà gestire il proprio saldo corrente e decidere come utilizzarlo.
- Il sistema sarà inizialmente disponibile esclusivamente come applicazione per pc.

2. Gestione dei Dipendenti

- Il sistema terrà traccia dei vari dipendenti e delle operazioni che svolgono durante il giorno, nonché dei loro turni e delle loro buste paga.
- Il responsabile di ciascun settore potrà visualizzare le prestazioni dei propri subordinati, allo scopo di ammonire e/o premiare i dipendenti meritevoli tramite un sistema di punti.
- Ogni dipendente potrà accedere alle informazioni riguardanti la propria busta paga e i turni che dovrà rispettare.

3. Gestione dei Recapiti

- Il sistema terrà traccia dello stato delle spedizioni prenotate, in corso ed effettuate in una base di dati.
- La base di dati interagirà con l'interfaccia fornita ai dipendenti che si occuperanno di aggiornare lo stato delle spedizioni.
- Per ogni missiva presa in carico il sistema sarà responsabile di associarle un codice identificativo univoco a 6 cifre che contraddistinguerà l'oggetto dall'inizio alla fine della sua lavorazione,

4. Gestione del Magazzino

- Il magazzino interagirà con il sistema tramite una base di dati.



- La gestione del magazzino sarà strettamente legata alla gestione dei recapiti, in quanto il magazzino contiene le corrispondenze in entrata e in uscita.
- Sarà possibile gestire ogni missiva tramite il codice e trovarla facilmente nonché catalogarla in base ai suoi dati.
- Per ogni oggetto nel magazzino sarà anche memorizzata la sua posizione negli scaffali.

5. Gestione Pagamenti

- Il sistema avrà la funzione di monitoraggio dei pagamenti verso l'azienda OPCal.
- Il cliente potrà visualizzare lo stato dei propri pagamenti.

6. Interfaccia

- Il sistema provvede un'interfaccia per utenti e dipendenti.
- Tale interfaccia avrà una duplice implementazione, un sito web e un applicativo per dispositivi mobili (Android o IOS).
- Le funzionalità esposte al pubblico saranno tutte disponibili tramite le interfacce specificate.



2 Analisi dei Requisiti

2.1 Analisi dello scenario



| Logistica in entrata (LE) | Attività operative (AO) | Logistica in uscita (LU) | Marketing e vendita (MV) | Servizi post-vendita (SPV) |
|--|---|--------------------------|-------------------------------|----------------------------|
| LE1: registrazione cliente | | | | |
| LE2: aggiornamento dati cliente | AO1: invio corrispondenza | LU1: notifica spedizione | MV1: incasso allo sportello | SPV1: gestione resi |
| LE3: registrazione operazioni dipendenti | AO2: monitoraggio spedizione | LU2: consegna a cliente | MV2: incasso con contrassegno | SPV2: gestione reclami |
| LE4: registrazione spedizioni in entrata | AO3: smistamento corrispondenza in arrivo | LU3: consegna ricevuta | MV3: comunicazione promozioni | SPV3: raccolta feedback |
| LE5: registrazione pagamento | AO4: selezione spedizioni in uscita | LU4: pagamento corriere | | |
| LE6: controllo magazzino | | | | |
| Approvvigionamenti (AP) | | | | |
| AP1: acquisto consumabili | | | | |
| AP2: acquisto nuova attrezzatura | | | | |
| Gestione risorse umane (GRU) | | | | |
| GRU1: gestione turni di lavoro | | | | |
| GRU2: gestione buste paga | | | | |
| GRU3: valutazione dipendenti | | | | |
| Gestione infrastrutture (GI) | | | | |
| GI1: manutenzione | | | | |



2.1.1 Registrazione cliente

Nome processo (identificativo): Registrazione cliente (LE1)

Attori coinvolti: Cliente, Sportellista, Portalettere

Archivi coinvolti: Lista clienti, Rubrica degli indirizzi

Descrizione processo: Un **cliente** può registrarsi recandosi fisicamente nella sede dell'ufficio e richiedendo l'apposito modulo di registrazione, da compilare con: nome, cognome, data di nascita e indirizzo (nel formato Via, CAP, Città, Provincia) e presentando un documento d'identità come patente o carta d'identità. Dopo di ciò lo **sportellista** dovrà premurarsi di controllare la coerenza delle informazioni all'interno del modulo confrontate con quelle del documento d'identità. Lo **sportellista** inserirà il documento all'interno della lista degli utenti, dove all'occorrenza inserirà anche i dati in merito alle spedizioni del **cliente**, (vedi LE4: registrazioni spedizioni in entrata) e all'interno della rubrica degli indirizzi dove un **portalettere** (vedi LU2: consegna a cliente) può attingere per avere informazioni sul cliente a cui deve consegnare. Nel caso in cui il **cliente** volesse accedere ai suoi dati si deve recare in sede e richiederli allo **sportellista**, che attingerà alla lista degli utenti, i dati presenti nella lista degli utenti sono costantemente aggiornati dagli **sportellisti** per garantire la loro correttezza e completezza. Questi aggiornamenti possono includere modifiche ai dati personali del cliente, come variazioni di indirizzo o recapiti (vedi LE2: aggiornamento dati cliente).

Processi correlati:

LE2, LE4, LU2.

Cosiderazioni dopo l'implementazione del nuovo sistema informativo:

Queste operazioni saranno gestite in automatico tramite l'apposita piattaforma, senza che il cliente vada in sede e senza che lo sportellista lo inserisca manualmente nell'archivio.





Figure 2: Activity Diagram di LE1



Figure 3: Data Flow Diagram LVL 0 di LE1



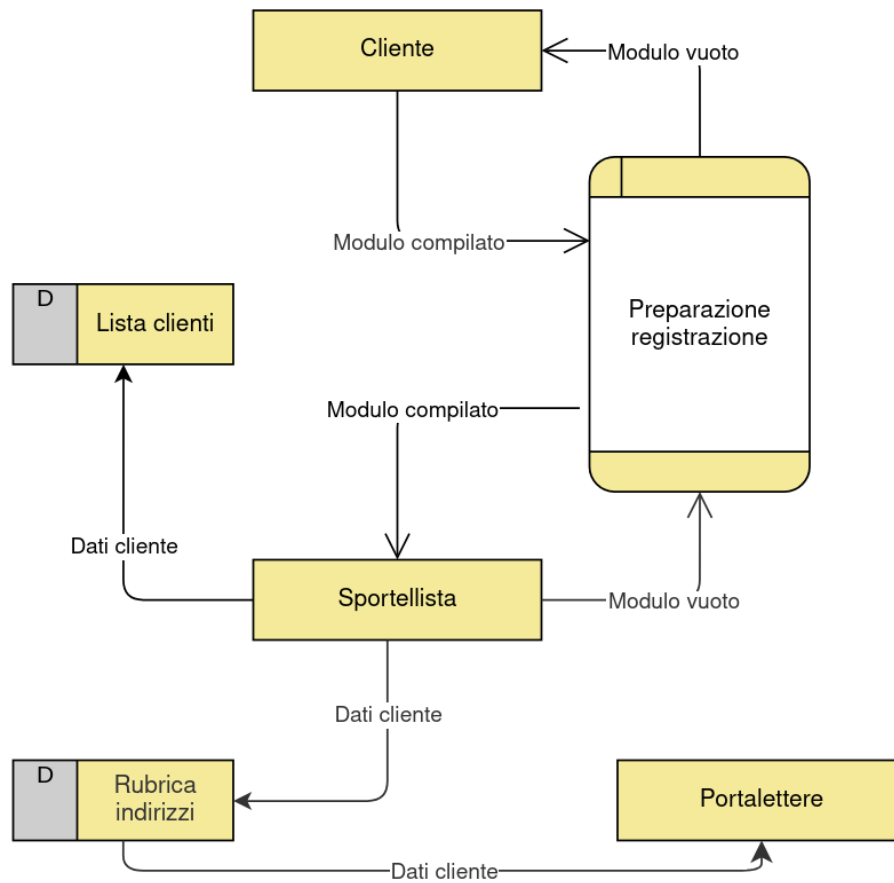


Figure 4: Data Flow Diagram LVL 1 di LE1



2.1.2 Invio corrispondenza

Nome processo (identificativo): Invio corrispondenza (AO1)

Attori coinvolti: Responsabile recapito, Magazziniere, Corriere

Archivi coinvolti: Registro spedizioni, Inventario, Rubrica corrieri, Lista spedizioni odierne, Registro pagamenti

Descrizione del processo: Quando notificato da LE6, il **responsabile recapito** controlla nel registro spedizioni il numero di articoli da spedire verso l'esterno. Sceglie un sottoinsieme di articoli (si veda AO4) e compila la lista spedizioni odierne, nella quale va ad inserire gli articoli da spedire in giornata. Una volta compilata la lista, consulta la rubrica corrieri al fine di trovare quello più conveniente alle condizioni specifiche. Una volta effettuata una stima, inizia a contattare le sedi di corrieri, a partire dalla più conveniente, fin quando trova un corriere disponibile in giornata, con il quale concorda un orario per il ritiro ed un prezzo. Una volta trovato l'accordo con il corriere, compila il documento da mandare al **direttore**, il quale provvederà al pagamento del **corriere** (LU4) e conserverà il documento nel registro pagamenti. Il direttore si occuperà inoltre di stampare la ricevuta di pagamento e consegnarla ai **magazzinieri**. Ricevuta la lista spedizioni odierne i **magazzinieri** consultano l'inventario, in cui è riportata la posizione dell'articolo all'interno del magazzino, e lo mettono da parte, in attesa del **corriere**. All'arrivo di questo, consegnano la ricevuta e caricano sul furgone gli articoli messi da parte.

Processi correlati:

AO4, LE6, LU4



Figure 5: Activity Diagram per AO1

Si riporta di seguito il dfd di livello 0:





Figure 6: DFD per AO1

Si riporta di seguito il dfd di livello 1:

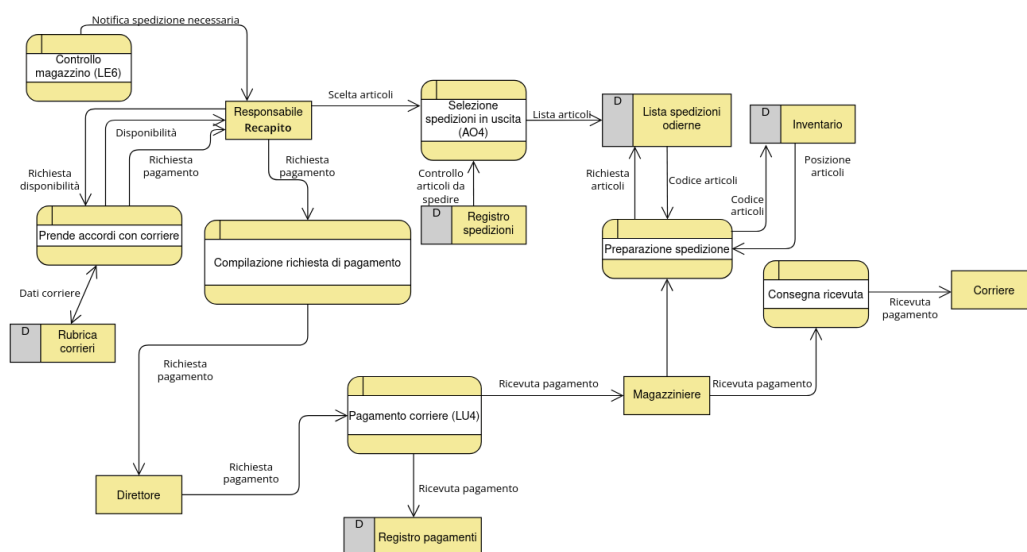


Figure 7: DFD livello 1 per AO1



2.1.3 Attori e archivi

| Attore | Descrizione | Processi in cui è coinvolto | Archivi a cui accede |
|-----------------------|--|------------------------------------|---|
| Cliente | Uno degli utenti dell'OPCAL | - LE1 | |
| Sportellista | Uno dei dipendenti che lavora nella sezione sportello | -LE1 | <ul style="list-style-type: none"> • Lista utenti • Rubrica degli indirizzi |
| Portalettere | Uno dei dipendenti che lavora nella sezione corrispondenze | -LE1 | <ul style="list-style-type: none"> • Rubrica degli indirizzi |
| Responsabile recapito | Il dipendente a capo del recapito | -AO1 | <ul style="list-style-type: none"> • Registro spedizioni • Lista spedizioni odierne • Rubrica corrieri |
| Direttore | Il responsabile generale, si occupa principalmente di contabilità | -AO1 | <ul style="list-style-type: none"> • Registro pagamenti |
| Magazziniere | Uno dei dipendenti addetto alla sezione magazzino | -AO1 | <ul style="list-style-type: none"> • Lista spedizioni odierne • Inventario |
| Corriere | Un ente esterno che si occupa di gestire la corrispondenza verso zone non sotto la competenza di OPCal | -AO1 | |



| Archivio | Descrizione | Processi in cui è coinvolto | Attori che vi accedono |
|-------------------------|--|------------------------------------|---|
| Lista degli utenti | Archivio in cui sono scritte le informazioni su ogni utente, quali nome, cognome, codice fiscale, e-mail | -LE1 | <ul style="list-style-type: none"> • Sportellista |
| Rubrica degli indirizzi | Archivio in cui sono scritte informazioni sugli utenti specifiche per i portalettere, queste sono nome, cognome, numero di telefono, indirizzo | -LE1 | <ul style="list-style-type: none"> • Portalettere |
| Registro spedizioni | Archivio in cui sono conservate le informazioni delle spedizioni accettate | -AO1 | <ul style="list-style-type: none"> • Sportellista • Responsabile recapito |
| Inventario | Archivio che testimonia lo stato del magazzino, conservando informazioni sugli articoli in esso presenti | -AO1 | <ul style="list-style-type: none"> • Magazziniere |



2.2 Specifica dei requisiti

Requisiti funzionali

I gruppi funzionali che si è deciso di implementare sono quelli riguardanti la **gestione dei clienti** e la **gestione dei recapiti**.

2.2.1 Gestione clienti

1. (MUST) Implementare una schermata di sign-in [Cliente];
2. (MUST) Inserimento dei propri dati anagrafici[Cliente];
3. (MUST) Modifica dei propri dati anagrafici[Cliente];
4. (MUST) Implementare la possibilità di poter tracciare le spedizioni[Cliente];
5. (MUST) Introdurre un ordinamento ascendente e discendente alla lista delle spedizioni[Cliente];
 - 5.1. Per data;
 - 5.2. Per stato (tranne consegnata);
6. (MUST) Aggiungere una funzione per iniziare una procedura di reso[Cliente];
7. (MUST) Scaricare tutti i documenti da compilare per procedere con il reso[Cliente];
8. (MUST) Visualizzare la lista dei resi richiesti [Cliente];
9. (MUST) Introdurre un ordinamento ascendente e discendente alla lista dei resi[Cliente];
 - 9.1. Per stato del reso (Default);
 - 9.2. Per data del reso;
10. (MUST) Poter annullare una procedura di reso[Cliente];
11. (MUST) Introdurre di un sistema di sicurezza per il login nella piattaforma[Cliente, Dipendente];
12. (MUST) Dare la possibilità di avere uno storico delle consegne[Cliente];
13. (MUST) Introdurre un ordinamento ascendente e discendente per lo storico delle consegne[Cliente]:
 - 13.1. Per data, dalla più alla meno recente (Default);
 - 13.2. Per numero di ordine;
14. (MUST) Visualizzare le ricevute a proprio carico[Cliente];
15. (MUST) Ordinare la lista delle ricevute a proprio carico[Cliente]:
 - 15.1. Per stato del pagamento (Default);
 - 15.2. Per data del pagamento;
 - 15.3. In ordine alfabetico;



16. (MUST) Visualizzare la lista dei clienti[Dipendente];
17. (MUST) Introdurre un ordinamento ascendente e discendente alla lista dei clienti [Dipendente];
 - 17.1. Per ordine alfabetico del cognome (Default);
18. (MUST) Visualizzare i dati anagrafici dei clienti [Dipendente];
19. (MUST) Modificare i dati anagrafici dei clienti [Dipendente];
20. (MUST) Implementare un sistema di mailing [Dipendente];

Per il **Requisito 20, Implementare un sistema di mailing**, si riporta di seguito una descrizione dettagliata:

Attore Principale: Sportellista

Descrizione dello scenario principale:

1. Un cliente si registra all'interno della piattaforma;
2. Inserisce nel campo e-mail il suo indirizzo;
3. Viene creata una *mailing list* e viene inserito all'interno questo cliente;
4. Comincia una pratica di reso, richiede una spedizione oppure un'appuntamento in sede;
5. Quando ci sono aggiornamenti su un qualcosa che concerne l'utente viene inviata una e-mail in materia;
6. Nel caso in cui ci siano informazioni sui servizi dell'organizzazione possono essere mandati aggiornamenti tramite le *newsletter*.

Descrizione di scenari alternativi:

SA1 (Non arriva l'email di aggiornamento):

1. Viene mandata una seconda e-mail;
2. Se il problema persiste si prova a sostituire l'indirizzo dell'utente o a creare una nuova mailing list.

Di seguito è riportato il diagramma dei casi d'uso per **l'Area Funzionale 1: Gestione clienti**, per semplificare la lettura è diviso in due :





Figure 8: Diagramma Casi d'Uso 1

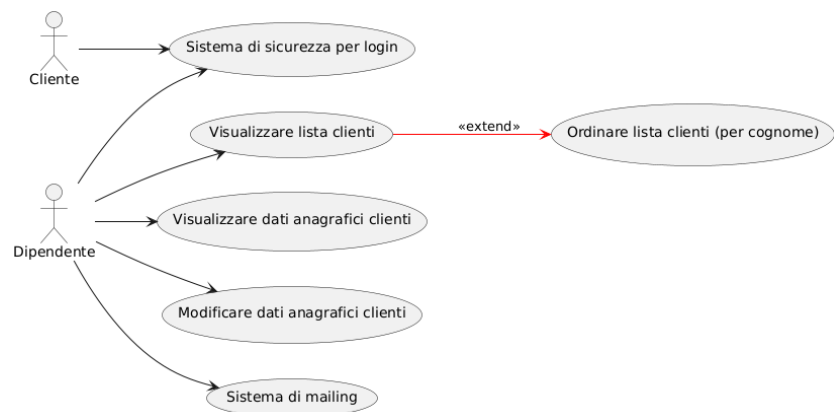


Figure 9: Diagramma Casi d'Uso 2



2.2.2 Gestione recapiti

1. (MUST) Associare un cliente ai suoi dati rilevanti [dipendente];
2. (MUST) Visualizzare gli indirizzi dei clienti [dipendente];
3. (MUST) Visualizzare le spedizioni in arrivo ad ogni un indirizzo [dipendente];
4. (MUST) Visualizzare le spedizioni da ritirare [dipendente];
5. (MUST) Creare una spedizione [dipendente];
6. (MUST) Visualizzare le spedizioni in arrivo al proprio indirizzo [cliente];
7. (MUST) Prenotare un ritiro [cliente];
8. (MUST) Ottenere un preventivo per una spedizione interna [cliente];
9. (MUST) Ottenere un preventivo per una spedizione esterna [cliente];
10. (MUST) Modificare il proprio orario di consegna preferito [cliente];
11. (MUST) Visualizzare l'orario di consegna preferito associato ad un indirizzo [dipendente];
12. (MUST) Tenere traccia delle spedizioni prenotate, in corso ed effettuate [dipendente];
13. (MUST) Modificare il ciclo di vita di una spedizione(prenotata, in corso, effettuata) [dipendente];
14. (MUST) Prendere in carico una spedizione [dipendente];
15. (MUST) Aggiornare lo stato di una spedizione [dipendente]:
 - 15.1. Da "prenotata" a "presa in carico" ;
 - 15.2. Da "presa in carico" a "spedita";
 - 15.3. Da "spedita" a "in consegna";
 - 15.4. Da "in consegna" a "consegnata";
 - 15.5. Da "in consegna" a "tentato recapito";
16. (MUST) Filtrare la lista degli indirizzi, la quale di default mostra tutti gli indirizzi, secondo i seguenti criteri[dipendente]:
 - 16.1. Nome del cliente;
 - 16.2. Comune di appartenenza;
 - 16.3. Via di residenza
 - 16.4. Orario preferito per la consegna;
17. (MUST) Ordinare la lista degli indirizzi, di default in ordine alfabetico, secondo i seguenti criteri[dipendente]:
 - 17.1. Nome del cliente;
 - 17.2. Comune di appartenenza;
 - 17.3. Via di residenza;



- 17.4. Orario preferito per la consegna;
- 18. (MUST) Filtrare la lista delle spedizioni, la quale di default mostra tutte le spedizioni, secondo i seguenti criteri[dipendente, cliente]:
 - 18.1. Per indirizzo di consegna;
 - 18.2. Per indirizzo di partenza;
 - 18.3. Per stato della spedizione;
- 19. (MUST) Ordinare la lista delle spedizioni, di default ordinata in base al codice, secondo i seguenti criteri[dipendente, cliente]:
 - 19.1. Prezzo crescente o decrescente;
 - 19.2. Peso crescente o decrescente;
 - 19.3. Data crescente o decrescente;
 - 19.4. Codice identificativo crescente o decrescente;
- 20. (MUST) Identificare una spedizione solo tramite il suo codice [dipendente, cliente];
- 21. (MUST) Assegnare ad ogni spedizione un codice univoco [dipendente];
- 22. (MUST) Visualizzare la lista dei corrieri esterni [dipendente];
- 23. (MUST) Calcolare un preventivo per il corriere esterno selezionato [dipendente];

N.B.

Tenere a mente la distinzione tra **ciclo di vita** e **stato** di una spedizione. Il **ciclo di vita** si riferisce alla situazione della spedizione(prenotata, in corso, effettuata). Per ognuna di queste, la spedizione verrà catalogata in basi di dati differenti. Lo **stato** è proprio solo delle spedizioni *in corso*(presa in carico, consegnata, ecc.).

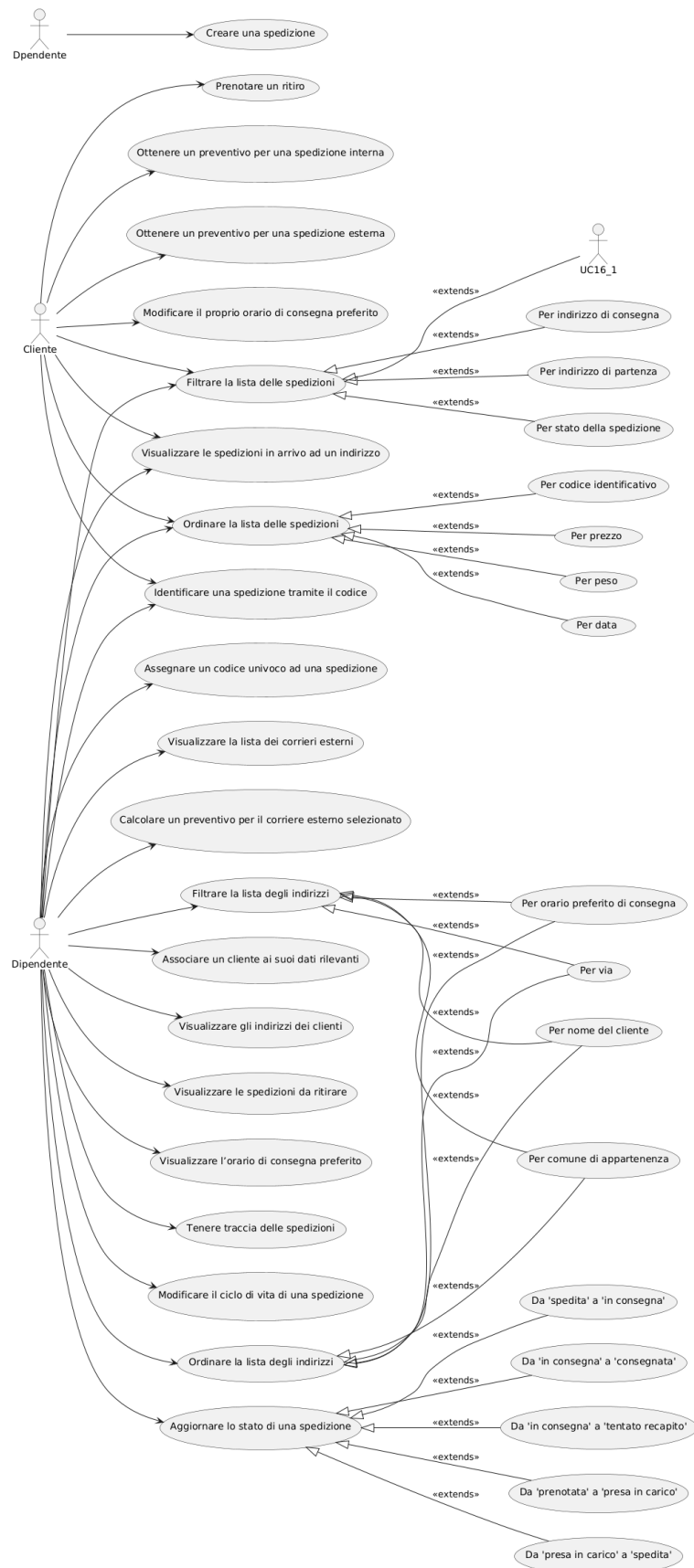
Per i **casi d'uso 5 (Creare una spedizione)** e **6 (Prenotare un ritiro)** segue descrizione dettagliata:

Una spedizione può essere creata da due attori separati:

- 1. Dipendente: nel caso in cui un cliente si rechi in sede per effettuare una spedizione, l'intera operazione viene effettuata dallo sportellista;
- 2. Cliente: il cliente può creare la spedizione tramite l'interfaccia offerta.

Di seguito è riportato il diagramma dei casi d'uso per **l'Area Funzionale 2: Gestione recapito** :





Requisiti non funzionali

Il sistema sarà inizialmente implementato tramite una propria applicazione per pc, che potrà essere utilizzata sia dal personale che dalla clientela, differenziando i due casi tramite un sistema di autorizzazioni.

Il sistema dovrà avere le seguenti caratteristiche:

- **Multiutente:** Diversi utenti, con permessi e livelli di autorità differenti dovranno potersi connettere e visualizzare dati contemporaneamente;
- **Multi postazione:** Il sistema dovrà consentire l'accesso a tutte le postazioni presenti in sede contemporaneamente;
- **Indipendenza:** Il sistema dovrà essere pronto all'utilizzo dopo un'unica installazione, soprattutto nel caso delle funzionalità offerte ai clienti;
- **Piattaforma:** Il sistema deve essere disponibile per il sistema operativo utilizzato in sede (Windows);
- **Backup:** Il sistema eseguirà un backup dei dati a cadenza settimanale;
- **Sicurezza:** Il sistema renderà sicura la gestione degli utenti, rendendo le alcune aree del programma accessibili solo dagli utenti con i permessi.

| Utente/Ruolo | Funzioni a cui ha accesso | Requisito funzionale(numero) |
|----------------|--|--|
| Amministratore | Tutte, inclusa la possibilità di aggiungere e/o rimuovere permessi | tutti |
| Dipendente | Può accedere a tutti i dati e ha la possibilità di modificarli, ad eccezione dei dati dei dipendenti | Da 2.2.1.16 a 2.2.1.20, da 2.2.2.1 a 2.2.2.5, da 2.2.1.11 a 2.2.2.23 |
| Cliente | Può accedere solo ad un set prestabilito di funzioni | Da 2.2.1.1 a 2.2.1.15, da 2.2.2.6 a 2.2.2.10, da 2.2.2.18 a 2.2.2.20 |

Di seguito è riportato un diagramma che raffigura la gerarchia tra gli utenti del sistema:

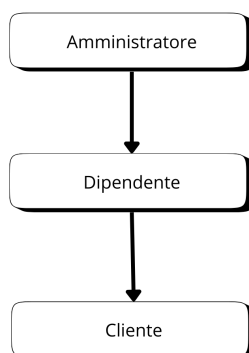


Figure 11: Diagramma dei privilegi



3 Progettazione

3.1 Modellazione della base di dati

Dai **requisiti 2.2.1.1-3, 2.2.1.16-19** si può dedurre che all'interno della base di dati dovranno essere rappresentati i dati riguardanti i *clienti*, specificatamente ogni utente sarà caratterizzato da nome, cognome, e-mail e password.

Dai **requisiti 2.2.1.4-5, 2.2.1.18-19** e praticamente l'intero **requisito funzionale 2.2.2** si evince che è importante descrivere i dati relativi alle *spedizioni*, caratterizzate da peso, da prezzo e codice, che funge da identificativo. Inoltre specificatamente dai **requisiti 2.2.2.13** si evince la necessità di rappresentare ogni ciclo di vita in una tabella a se stante:

- Quando *prenotata*, una spedizione sarà caratterizzata da codice, un cliente emittente, data di prenotazione, indirizzo di ritiro, data prevista di ritiro, cliente destinatario, peso e prezzo.
- Una volta ritirata, la spedizione verrà riferita come *in corso* e sarà caratterizzata da emittente, codice identificativo, data di spedizione, peso, prezzo, destinatario, data prevista di consegna e stato.
- Una volta consegnata, la spedizione sarà *effettuata* e sarà caratterizzata da emittente, codice identificativo, data di spedizione, data di consegna, destinatario, prezzo, peso.

Dai **requisiti 2.2.1.6-10** il cliente può far partire delle procedure di *reso*, caratterizzate dalla data in cui sono iniziate e la spedizione per cui è stata fatta partire. Un cliente può iniziare più resi, ma solo per le spedizioni da esso ricevute.

Dai **requisiti 2.2.1.12-13** si evince che il cliente deve avere associato la lista delle *consegne* a proprio carico, esse sono un tipo di *spedizione*, ogni cliente può avere molte consegne o nessuna, ma una consegna può essere assegnata a soli due clienti, un emittente ed un destinatario.

I **requisiti 2.2.1.14-15** ci dicono che ogni cliente deve avere a se relazionato la lista delle *ricevute* a proprio carico, identificate dal numero della spedizione a cui si riferiscono e caratterizzata dalla data di emissione e dallo stato del pagamento. Ogni ricevuta può avere un cliente, ma un cliente può averne molte, allo stesso tempo, una spedizione può avere una ricevuta e una ricevuta si riferisce ad una sola spedizione.

I **requisiti 2.2.2.2, 16 e 17** ci guidano verso la rappresentazione dei dati relativi agli indirizzi, che dovranno quindi includere comune, via e civico, che insieme formano l'identificativo, l'email del cliente a cui si riferiscono, e l'orario di consegna preferito. Dai **requisiti 2.2.2.22, 23** si evince la necessità di una rubrica corrieri. Ogni corriere sarà caratterizzato da nome, partita iva(identificativa), numero di telefono, sito web, prezzo per 1kg, prezzo per 10kg, prezzo per 100kg.



3.1.1 Modello concettuale

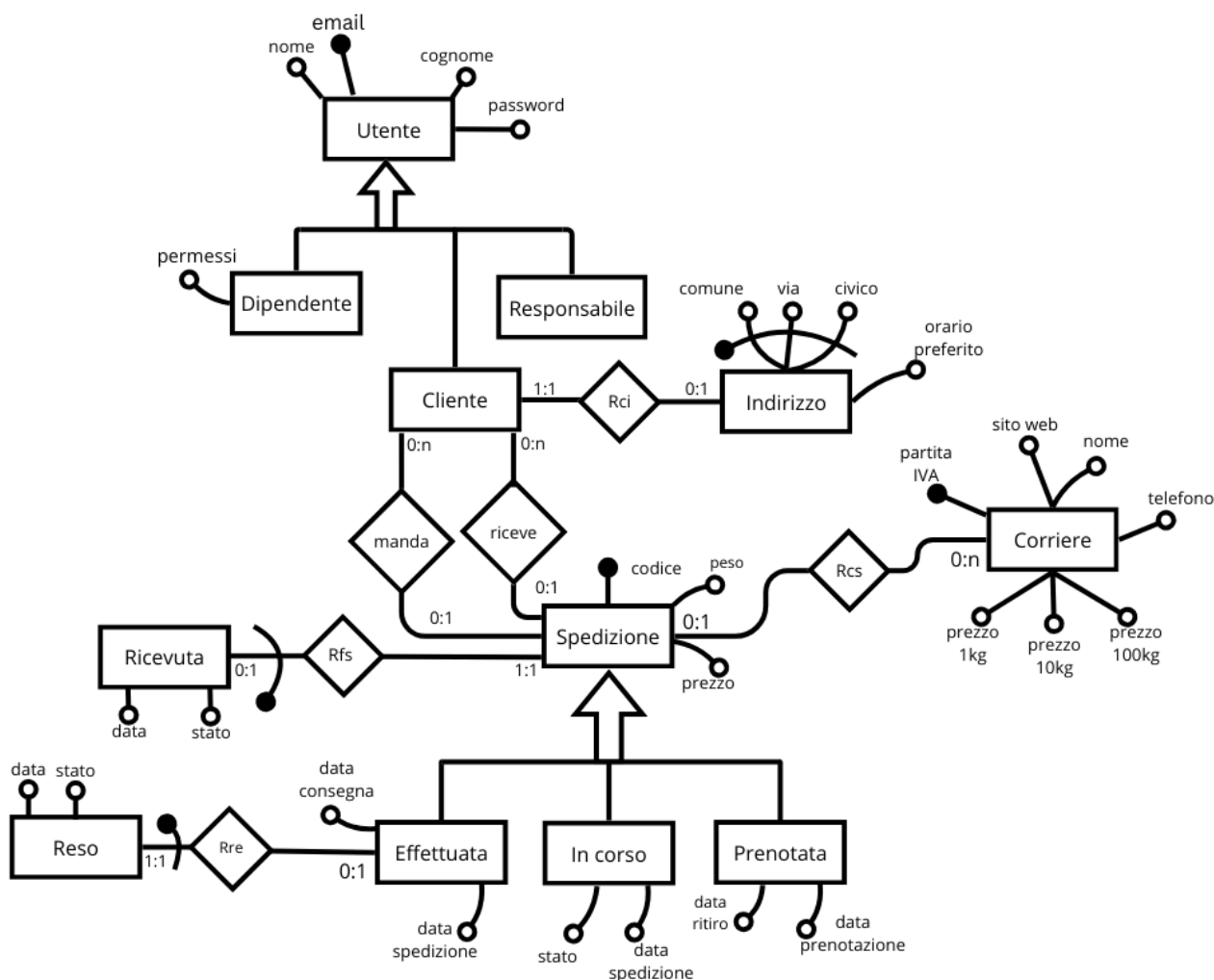


Figure 12: Diagramma E-R

Vincoli d'integrità

1. Gli attributi *data ritiro* e *data consegna* devono sempre essere successivi rispettivamente a *data prenotazione* e *data spedizione*;
2. Il valore dell'attributo *stato* all'interno dell'entità *reso* potrà essere solo: richiesto, processato, terminato;
3. Il valore dell'attributo *data* all'interno dell'entità *reso* deve essere posteriore al valore *data consegna* all'interno dell'entità *Effettuata*;
4. Il valore dell'attributo *stato* dell'entità *ricevuta* può essere: pagamento confermato, non regolarizzata.
5. Il valore dell'attributo *stato* all'interno dell'entità *in corso* potrà essere solo: presa in carico, spedita, arrivata alla filiale, in consegna, in attesa di ritiro;

Regole di derivazione

Alcuni attributi devono essere derivati automaticamente dal sistema:



- L'attributo *prezzo* dell'entità **Spedizione** è calcolato tramite una tabella di prezzi basata ai pesi.
- L'attributo identificativo *codice* dell'entità **Spedizione** viene generato automaticamente dal sistema

Dizionario dei concetti per entità

| Entità | Descrizione | Attributi | Identificatore |
|----------------|--|--|---------------------|
| Utente | Rappresenta ogni utente del sistema | Nome, cognome, e-mail, password | E-mail |
| Amministratore | Rappresenta gli utenti con ogni permesso | Vedi utente | E-mail entità madre |
| Dipendente | Rappresenta ogni dipendente con accesso al sistema | Vedi utente, permessi | E-mail entità madre |
| Cliente | Rappresenta i clienti dell'organizzazione | Vedi utente | E-mail entità madre |
| Reso | Rappresenta i resi iniziati dai clienti | Data, stato, codice | Spedizione |
| Indirizzo | Rappresenta l'indirizzo di spedizione dei clienti | Comune, Via, Civico, Orario, email associata | Comune, Via, Civico |
| Ricevuta | Rappresenta le ricevute delle spedizioni dei clienti | Codice, stato, importo, data | Codice |
| Spedizione | Rappresenta le spedizioni | Codice, peso, prezzo, email destinatario, email mittente, corriere | Codice |
| Effettuata | Rappresenta le spedizioni effettuate | Vedi spedizione, data di consegna, data di spedizione | Codice entità madre |
| In corso | Rappresenta le spedizioni in corso | Vedi spedizione, data di spedizione, stato | Codice entità madre |
| Prenotata | Rappresenta le spedizioni prenotate | Vedi spedizione, data di prenotazione, data di ritiro | Codice entità madre |
| Corriere | Rappresenta i corrieri associati all'organizzazione | Partita IVA, sito web, nome, telefono, prezzo 1kg, prezzo 10kg, prezzo 100kg | Partita IVA |



Dizionario dei concetti per relazione

| Relazione | Descrizione | Entità coinvolte | Attributi |
|-----------|--|----------------------|-----------|
| R_{ci} | Associa i clienti ai loro indirizzi | Cliente, Indirizzo | |
| R_{fs} | Associa le spedizioni alle loro ricevute | Ricevuta, Spedizione | |
| Manda | Associa il cliente alla spedizione che fa spedire | Cliente, Spedizione | |
| Riceve | Associa il cliente alla spedizione che riceve | Cliente, Spedizione | |
| R_{re} | Associa il reso alla spedizione di tipo effettuata su cui è stato effettuato | Reso, Effettuata | |
| R_{cs} | Associa la spedizione al corriere col quale è stata inviata | Spedizione, Corriere | |

3.1.2 Modello relazionale

La sintassi utilizzata nel modello relazionale è la seguente:

- Per ogni schema di relazione, il nome viene indicato in grassetto, mentre i suoi attributi sono espressi tra parentesi, ad esempio **Relazione**(attributo1,..., attributoN)
- Negli schemi di relazione, vengono indicati i vincoli di chiave primaria sottolineando gli attributi parte della chiave, ad esempio **Relazione**(attributo1, attributo2,..., attributoN)
- Eventuali altre chiavi verranno specificate al di sotto della relazione con la keyword "Unique" in corsivo, ad esempio
Relazione(chiave, attributo)
Unique(attributo)
- Eventuali vincoli di integrità referenziale saranno descritti al di sotto della relazione a cui appartengono, nella forma **R1**(a1) \subseteq_{FK} **R2**(a1);
- Per indicare che un attributo può essere NULL, viene indicato al pedice dell'attributo la dicitura NULL, ad esempio **R**(k, A1_{NULL}, A2) indica che l'attributo A1 potrebbe essere NULL;

I tipi di dato utilizzati per gli attributi sono i seguenti:

- INT: quantità intere;
- FLOAT: prezzi, pesi;
- DATE: rappresenta le date nel formato gg/mm/aaaa;
- VAR_CHAR: stringhe di testo, codici alfanumerici;



Il modello relazionale sarà il seguente:

- **Utente**(email, nome, cognome, password);
- **Amministratore**(email);
- **Dipendente**(email, permessi);
- **Cliente**(email);
- **Indirizzo**(comune, via, civico, orario_preferito, email_cliente)
Indirizzo(email_cliente) \subseteq_{FK} **Cliente**(email);
- **Ricevuta**(codice, stato, data)
Ricevuta(codice) \subseteq_{FK} **Spedizione**(codice);
- **Spedizione**(codice, peso, prezzo, email_mittente, email_destinatario, corriere)
Spedizione(email_mittente) \subseteq_{FK} **Cliente**(email)
Spedizione(email_destinatario) \subseteq_{FK} **Cliente**(email)
Spedizione(corriere) \subseteq_{FK} **Corriere**(iva)
- **Effettuata**(codice, data_consegna, data_spedizione)
Effettuata(codice) \subseteq_{FK} **Spedizione**(codice)
 Lo schema di dati **Effettuata** di fatto funge da storico delle spedizioni;
- **In_corso**(codice, data_spedizione, stato)
In_corso(codice) \subseteq_{FK} **Spedizione**(codice);
- **Prenotata**(codice, data_ritiro, data_prenotazione)
Prenotata(codice) \subseteq_{FK} **Spedizione**(codice);
- **Corriere**(iva, nome, sito, telefono, prezzo1, prezzo10, prezzo100);
- **Reso**(codice, data, stato)
Reso(codice) \subseteq_{FK} **Effettuata**(codice);

Triggers

In merito al *vincolo di integrità* 4 che tratta lo schema di relazione **Reso**(Codice, Data, Stato) ed **Effettuata**(Codice, DataConsegna, DataSpedizione), nel momento in cui all'interno dell'attributo data dell'entità **Reso** si vuole inserire una nuova tupla bisogna controllare che la condizione per cui la *Data* che si sta cercando di inserire nel **Reso** sia maggiore (posteriore) a quella che è attualmente all'interno di *DataConsegna* di **Effettuata**.

```

1 DELIMITER //
2 CREATE TRIGGER 'vincoli_date_insert'
3 BEFORE INSERT ON Reso
4 FOR EACH ROW BEGIN
5 IF (EXISTS
6     (SELECT *
7      FROM Effettuata
8      WHERE NEW.data > dataCosegna))
9 THEN SIGNAL SQLSTATE '45000'
```



```

10      SET MESSAGE_TEXT = 'Vincolo di integrità su Reso non rispettato';
11  END IF;
12  END //
13  DELIMITER;

```

Di seguito è anche riportata la versione analoga per gli aggiornamenti nella tupla.

```

1  DELIMITER //
2  CREATE TRIGGER 'vincoli_date_update'
3  BEFORE UPDATE ON Reso
4  FOR EACH ROW BEGIN
5  IF (EXISTS
6      (SELECT *
7       FROM Effettuata
8       WHERE NEW.data > dataCosegna))
9  THEN SIGNAL SQLSTATE '45000'
10     SET MESSAGE_TEXT = 'Vincolo di integrità su Reso non rispettato';
11  END IF;
12  END //
13  DELIMITER;

```

Un altro vincolo d'integrità è quello per valore dell'attributo *Stato* all'interno di **Ricevuta**, quando si va ad inserire bisogna infatti controllare che sia all'interno dell'insieme degli stati ammissibili.

```

1  DELIMITER //
2  CREATE TRIGGER 'vincoli_ricevute_insert'
3  BEFORE INSERT ON Ricevuta
4  FOR EACH ROW BEGIN
5  IF (NEW.stato NOT IN ('pagamento confermato', 'non regolarizzata'))
6  THEN SIGNAL SQLSTATE '4500'
7      SET MESSAGE_TEXT = 'Vincolo di integrità su Ricevuta non rispettato';
8  END IF;
9  END //
10 DELIMITER;

```

Segue la versione analoga per gli update.

```

1  DELIMITER //
2  CREATE TRIGGER 'vincoli_ricevute_insert'
3  BEFORE UPDATE ON Ricevuta
4  FOR EACH ROW BEGIN
5  IF (NEW.stato NOT IN ('pagamento confermato', 'non regolarizzata'))
6  THEN SIGNAL SQLSTATE '4500'
7      SET MESSAGE_TEXT = 'Vincolo di integrità su Ricevuta non rispettato';
8  END IF;
9  END //
10 DELIMITER;

```



3.2 Modellazione della logica di business

Viene riportato il primo diagramma che rappresenta la logica di business, è importante annotare che NON sono raffigurate le classi, e quindi i metodi, che sono all'interno della parte del Controller e della View.

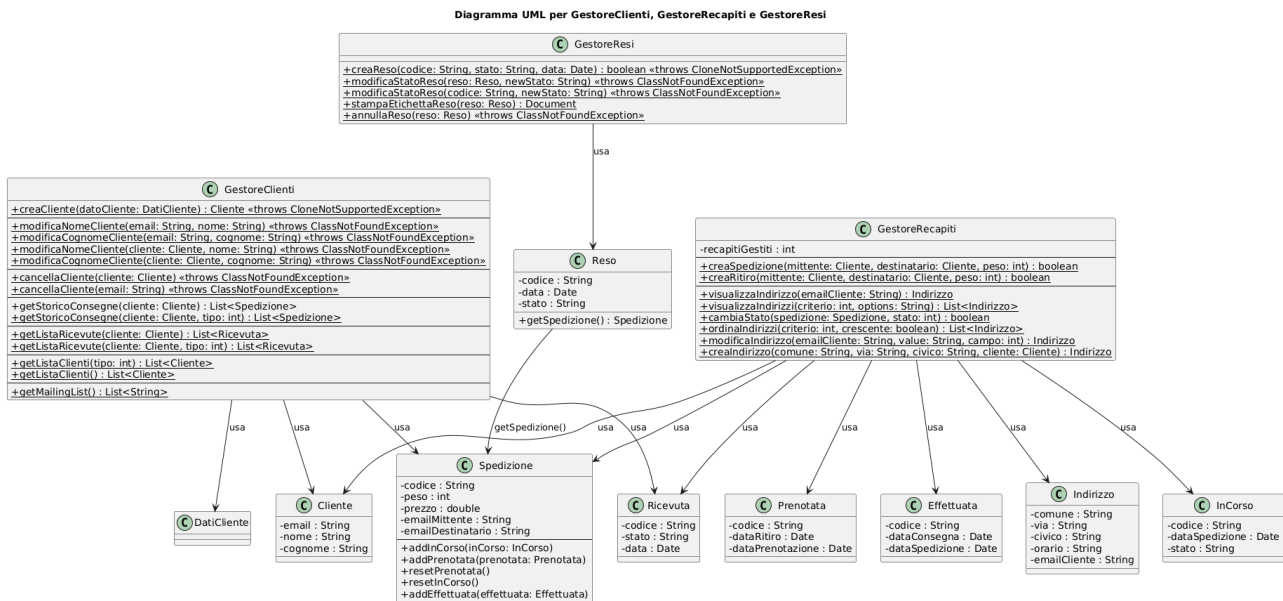


Figure 13: Diagramma

Tutti i metodi di ricerca sono implementati tramite la funzione **criteria** di Torque, nello specifico il metodo *getStoricoConsegne(Cliente cliente, int tipo)* restituisce la lista delle consegne del cliente in input, l'ordinamento in questo metodo, come in tutti gli altri è dettato dalla variabile *tipo* che è inserita in input, come dal *requisito funzionale 2.2.1.5* il numero 0 da un'ordinamento ascendente per data, 1 discendente per data, 2 ascendente per stato e 3 discendente per stato. Il metodo è riportato dopo come è implementato in java e SQL per semplicità è mostrato solo il caso di tipo 0.

```

1 Criteria criteria = Criteria().addAscendingOrderByColumn(EffettuataPeer.DATA_CONSEGNA));
2 criteria.addJoin(SpedizionePeer.CODICE, EffettuataPeer.CODICE);
3 criteria.where(SpedizionePeer.EMAIL_DESTINATARIO, cliente.getEmail());
4 criteria.addSelectColumn(SpedizionePeer.CODICE);
5
6 List<Spedizione> ris = new ArrayList<>();
7
8 try {
9     ris = SpedizionePeer.doSelect(criteria);
10 } catch (TorqueException e) {
11     System.out.println("Errore nella query");
12 }
13
14 return ris;
  
```

```

1 SELECT Spedizione.CODICE
2 FROM Spedizione
  
```



```
3 JOIN Effettuata ON Spedizione.CODICE = Effettuata.CODICE
4 WHERE Spedizione.EMAIL_DESTINATARIO = "test@example.com"
5 ORDER BY Effettuata.DATA_CONSEGNA ASC;
```

Torque nella creazione e modifica delle singole tabelle si assicura che le operazioni eseguite siano parte della stessa transazione. Questo però non avviene nel caso in cui si debbano andare a modificare allo stesso tempo più tabelle collegate da una foreign key. Per rispettare le proprietà **ACID** quindi, nel momento in cui si è andato ad effettuare operazioni simili, si è provveduto a creare manualmente una connessione ed effettuare le operazioni all'interno di una singola operazione. Ne è un esempio il metodo **creaSpedizione** in **Gestore Recapiti**:

```
1 public static boolean creaSpedizione(Cliente mittente,
2     Cliente destinatario, int peso) {
3     String codice = generaCodice();
4     Connection connection = null;
5     InCorso inCorso = new InCorso(codice);
6     Spedizione spedizione = new Spedizione(codice, mittente.getEmail(),
7         destinatario.getEmail(), peso, calcolaPrezzo(peso));
8     try {
9         connection = Transaction.begin();
10        inCorso.save();
11        spedizione.addInCorso(inCorso);
12        spedizione.save();
13        creaRicevuta(spedizione);
14        Transaction.commit(connection);
15    } catch (SQLException e) {
16        Transaction.safeRollback(connection);
17        return false;
18    }
19    return true;
20 }
```

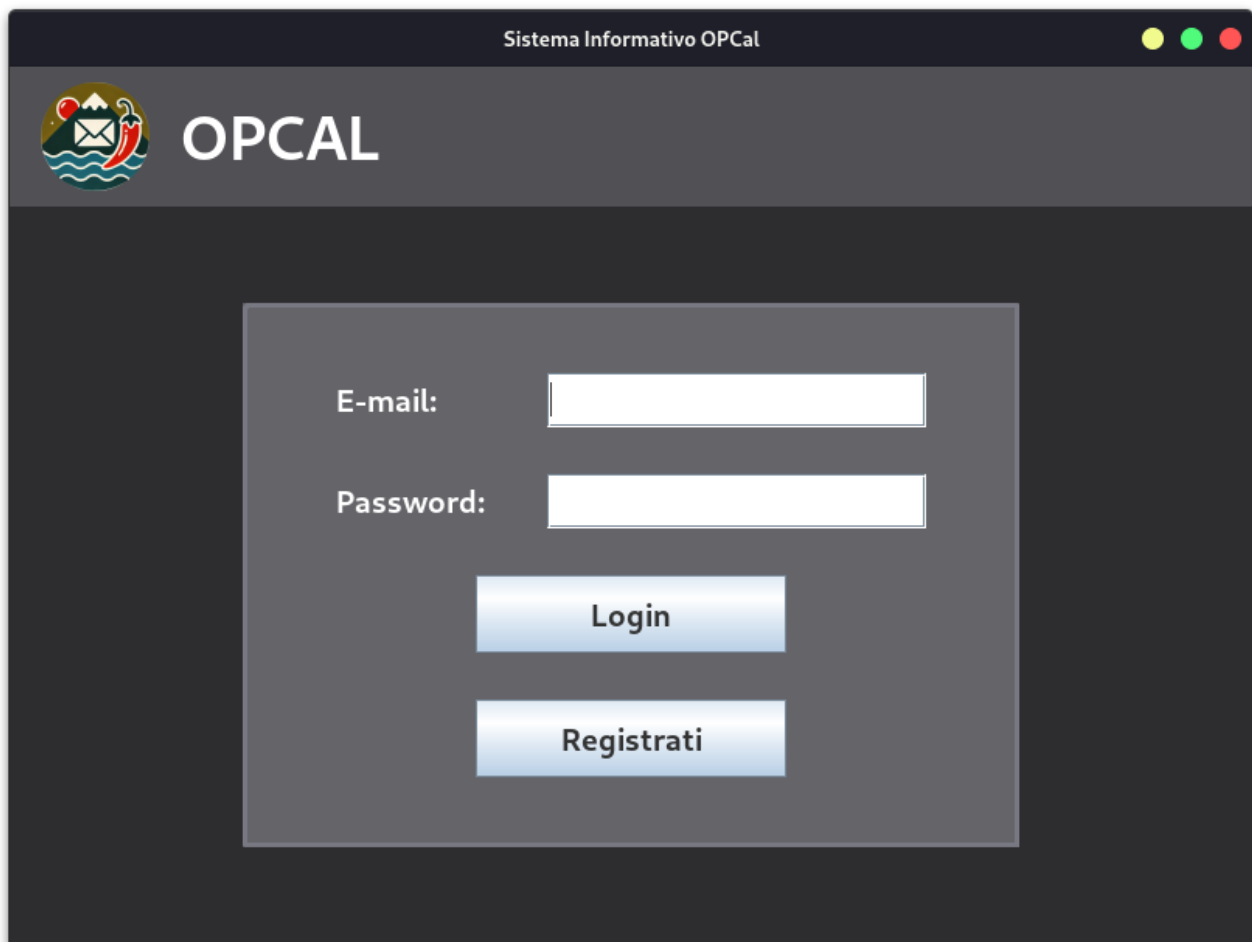
Grazie a queste considerazioni possiamo essere sicuri che anche in caso di errori la base di dati sia lasciata in uno stato consistente.

I metodi di tipo update, invece, soprattutto quelli della sezione Gestione Recapiti e quelli della Gestione Resi devono controllare che gli stati inseriti all'interno della funzione cambiaStato e modifica Stato siano ammessi.

3.3 Modellazione delle interfacce

All'accensione del sistema informatico si è accolti dalla schermata di login



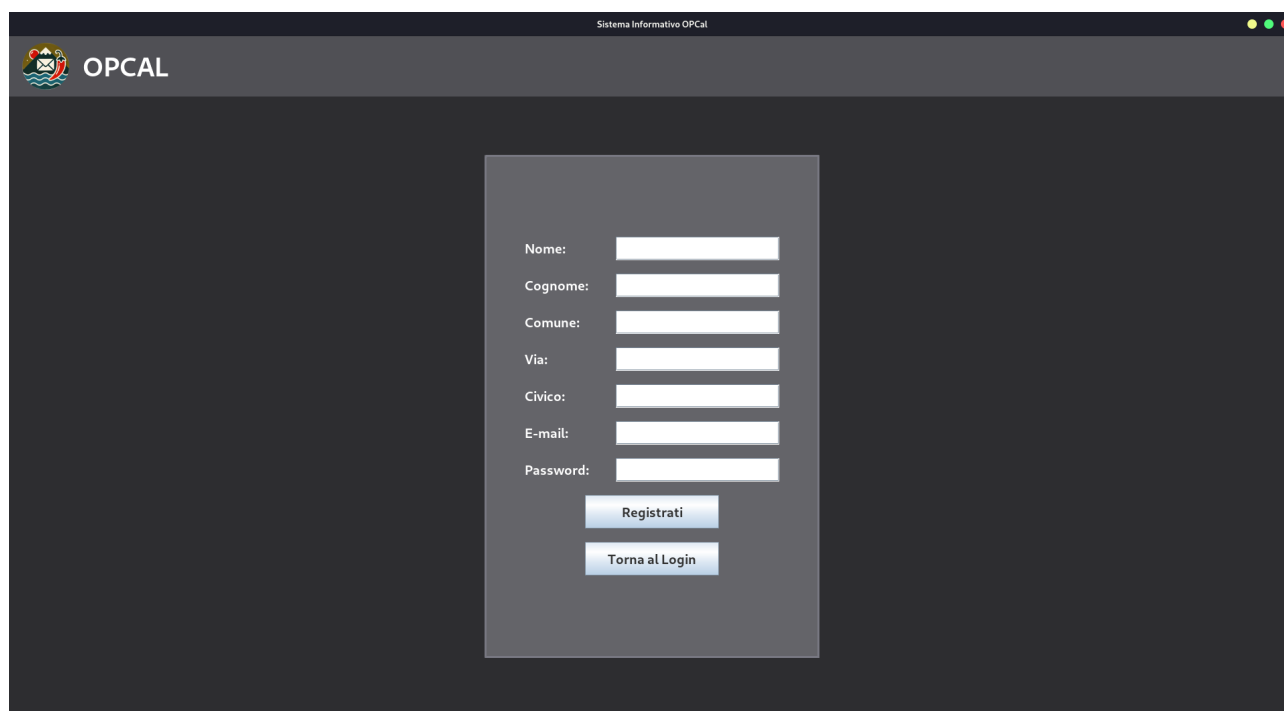


The image shows a web application window titled "Sistema Informativo OPCal". The header features a logo on the left and the text "OPCAL" in large white letters. The main content area is a dark gray rectangle containing a lighter gray box with the login form. The form has two input fields: "E-mail:" and "Password:". Below these fields are two buttons: "Login" and "Registrati".

Figure 14: Schermata di login

Qui ci sono due opzioni, nel caso si è già in possesso di un account ci si può muovere alla main page, altrimenti un utente si può registrare premendo il tasto *Registrati* e quindi muovendosi nella pagina di registrazione





Sistema Informativo OPCAL

OPCAL

Nome:

Cognome:

Comune:

Via:

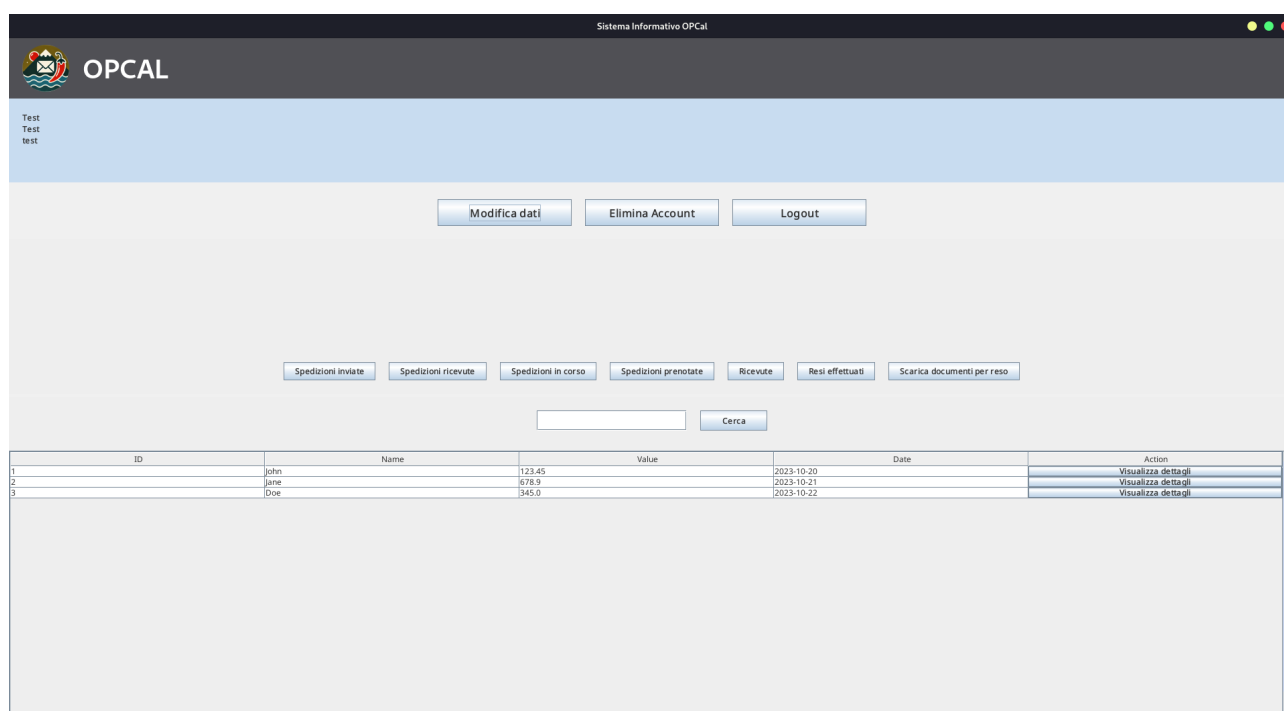
Civico:

E-mail:

Password:

Figure 15: Schermata di registrazione

Riempendo i campi all'interno di questa schermata quindi si può creare una entry nel database con i propri dati.



Sistema Informativo OPCAL

OPCAL

Test
Test
test

| ID | Name | Value | Date | Action |
|----|------|--------|------------|--|
| 1 | John | 123.45 | 2023-10-20 | <input type="button" value="Visualizza dettagli"/> |
| 2 | Jane | 678.9 | 2023-10-21 | <input type="button" value="Visualizza dettagli"/> |
| 3 | Doe | 345.0 | 2023-10-22 | <input type="button" value="Visualizza dettagli"/> |

Figure 16: Schermata principale

Qui ci sono diverse opzioni, partendo dall'alto, il pannello azzurro ci mostrerà i nostri dati, quali nome, cognome, email ed indirizzo. A seguire abbiamo i pulsanti tramite i quali possiamo gestire il nostro account, modificando dati, eliminando l'account e tutti i dati associati, oppure facendo il logout. A seguire sono presenti i tasti che ci permettono di eseguire delle query nel database, in base al comando scritto sul pulsante. Sotto a questo troviamo una barra di ricerca,



che ci permette di filtrare i risultati ottenuti in base ai criteri descritti nella **Sezione 2**. Infine, in fondo alla pagina possiamo trovare la tabella con i risultati delle query richieste, che sarà possibile ordinare cliccando sul nome del campo.

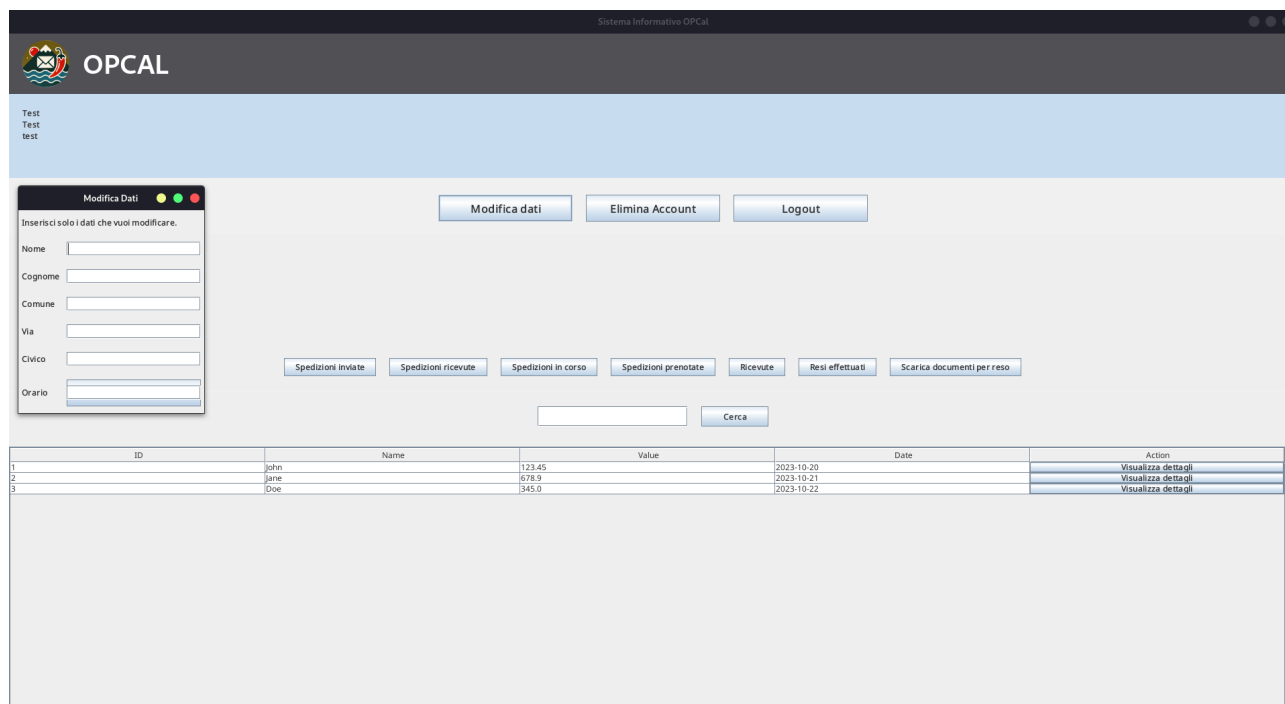


Figure 17: Schermata principale con modifica

Per modificare i propri dati, nel caso del cliente, oppure i dati relativi a spedizioni, indirizzi o clienti, nel caso dei dipendenti, si aprirà un dialog "Modifica Dati", il cui contenuto cambierà dinamicamente in base alla entry selezionata. Sarà possibile andare ad inserire soltanto i valori che necessitano di essere aggiornati, e il resto dei valori rimarranno immutati.

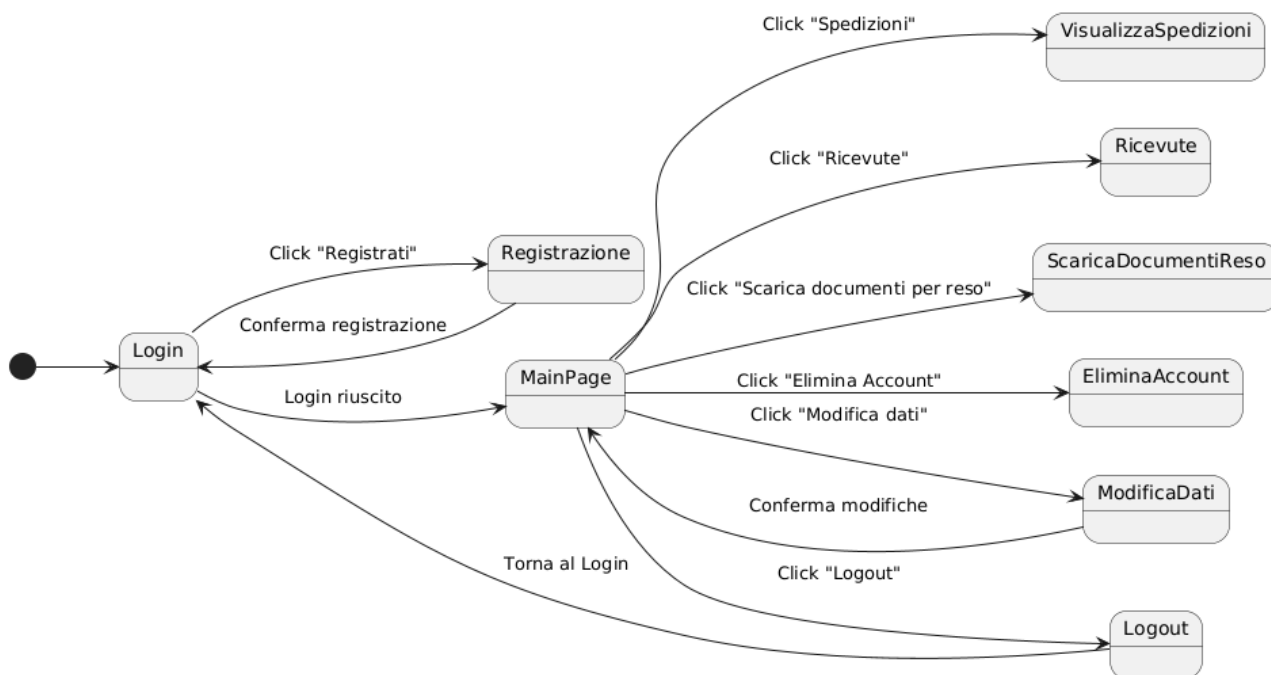


Figure 18: Diagramma di Navigazione



4 Implementazione

4.1 Base di dati

La base di dati è appoggiata su MySQL, ma in Java è implementata tramite *Torque*, quindi tramite un'approccio di tipo ORM, le classi da lui generate sono quindi degli accessi diretti alle entità. Per ogni tabella dello schema specificato saranno generate numerose classi.

Quelle che interessano e sono state utilizzate e modificate sono `<tabella>.java` e `<tabella>Peer.java`, che estendono rispettivamente `Base<tabella>.java` e `Base<tabella>Peer.java`. La differenza tra queste due classi è che la classe `<tabella>.java` ci permette di operare su singole tuple, ad esempio tramite operazioni di **update**, mentre la classe `<tabella>Peer.java` si riferisce all'intera tabella e ci permette di operare su di essa, è il caso di operazioni di tipo **select**.

Ad esempio per la tabella *Spedizioni* si è andato a modificare le classi *Spedizione.java* e *SpedizionePeer.java*, per adattarle alle esigenze del sistema. Ognuna di queste classi viene acceduta esclusivamente dalle classi *Gestore* tramite dei metodi statici.

4.2 UI

La gui è stata implementata tramite il pacchetto Java Swing.

4.3 Design Pattern

All'interno dell'implementazione sono stati utilizzati 2 design pattern:

4.3.1 Model View Controller

Il pattern MVC è utilizzato per far comunicare le 3 parti principali della base di dati, all'interno del package *model* quindi sono presenti tutti i metodi della logica di business, all'interno del package *view* invece sono presenti quelli per la modellazione dell'interfaccia grafica, infine all'interno del package *controller* ci sono tutte quelle classi atte a regolare le comunicazioni tra queste due parti.

4.3.2 Data Transfer Object

La classe astratta *Dati*, con le sue due implementazioni *Dati Cliente* e *Dati Sportellista* sono delle classi utilizzate per comprimere i dati in un singolo oggetto e comodamente trasferirle dal Model (*Gestione Clienti*) al Controller e quindi alla View, di seguito quindi è riportato il codice della classe *Dati*.

```
1 public abstract class Dati {
2     private final String nome;
3     private final String cognome;
4     private final String email;
5     private final String passwd;
6
7     public Dati(String nome, String cognome, String email, String passwd) {
8         this.nome = nome;
9         this.cognome = cognome;
10        this.email = email;
```




```
11         this.passwd = passwd;
12     }
13     public String getNome() {
14         return nome;
15     }
16     public String getCognome() {
17         return cognome;
18     }
19     public String getEmail() {
20         return email;
21     }
22     public String getPassword() {
23         return passwd;
24     }
25 }
```

4.3.3 State

La distinzione tra interfaccia per cliente e interfaccia dipendente viene effettuata silenziosamente in base alle credenziali di login, tramite un pattern state che collega in automatico le implementazioni necessarie.



5 Disclaimer

Il codice finale, così come le implementazioni delle interfacce, potrebbe differire leggermente da quello riportato all'interno di questo report, ma i requisiti funzionali descritti rimarranno garantiti anche in versioni successive. Inoltre, ogni cambiamento avrà come obiettivo l'aggiunta di funzionalità ed il miglioramento della manutenibilità dell'applicativo.

Per visualizzare la versione più aggiornata visitare la [repository github](#).

