Università degli Studi della Calabria

Facoltà di Scienze Matematiche Fisiche e Naturali

Corso di Laurea in Informatica

Progetto per il corso di
Basi di Dati Evolute
a.a. 2009/2010

Applicazione aziendale per la gestione di un inventario di beni

Docente del corso: **Prof. P. Rullo**
Docenti di Laboratorio: **Ing. G. Laboccetta,** Ing. C. Cumbo.
Coadiutore: Dott.ssa V. Policicchio

116342 Andrea Martire
114654 Loria Salvatore
115301 Umberto Agosto

Descrizione

Il progetto realizza un sistema informativo sul web per la gestione di beni aziendali.

E' stato realizzato con due obiettivi principali:

- realizzare un sistema semplice ed intuitivo da utilizzare, senza troppi fronzoli grafici, puntando più sulle funzionalità e sull'affidabilità.
- rendere l'applicazione indipendente dal DBMS utilizzato, con il grosso vantaggio di poter cambiare sistema senza dover riscrivere l'applicazione.

In particolare i DBMS sui quali l'applicazione è stata testata sono PostgreSQL 8.4 e Oracle 10g XE, ma non dovrebbero sorgere problemi con DBMS diversi. Il garantire questa compatibilità ha introdotto diverse problematiche: gestione delle sequenze da applicazione e non da DBMS; aggiornamenti e cancellazioni potenzialmente pericolosi gestiti direttamente da applicazione e non con trigger; estrema attenzione ai tipi di dato utilizzati, in particolar modo alle date; invio mail da applicazione; Report e controlli vari da applicazione.

Sequence

Non era possibile usare le sequenze messe a disposizione dai vari DBMS in quanto differivano tra loro su diversi aspetti, primo fra tutti la sintassi; abbiamo così deciso di creare una classe Java/JDBC Sequencer che gestisse le sequenze in maniera indipendente dal BDMS.

Il punto principale nello sviluppo di questa classe, era il fatto che bisognava mantenere le sequenze nel tempo, anche dopo la chiusura del server web; inoltre essendo un ambiente fortemente multithread, c'era anche la necessità di gestire in maniera corretta le varie richieste di “nextval” su una certa sequenza. Abbiamo quindi sviluppato due versioni del nostro “Sequencer”. Entrambe prevedono un metodo “nextval(String sequenza)” che ritorna il prossimo valore della sequenza specificata; in particolare se la sequenza specificata non esiste, viene creata al volo ed inizializzata di default con partenza da 0 ed incremento di 1. Il metodo nextval è ovviamente un metodo sincronizzato per evitare anomalie in esecuzioni multithread.

Le due versioni di Sequencer differiscono invece per il salvataggio delle sequenze:

la prima classe salva le sequenze, mantenute a livello logico in una HashMap, su un file, che viene letto quando viene richiamato il metodo nextval, e viene scritto prima di ritornare il valore (eventuali eccezioni in fase di lettura/scrittura del file vengono propagate alla classe chiamante, in modo da gestirle opportunamente).

La seconda classe si serve invece di una tabella del database. All'invocazione del metodo nextval,

viene letta la tabella, ed eventualmente creata al volo se non esiste, si fa un aggiornamento in cui viene incrementato il valore corrente nella sequenza specificata, e infine si ritorna questo stesso valore. Queste operazioni vengono gestite dalla classe chiamante come una transazione JDBC, in quanto una eccezione nell'esecuzione non deve lasciare incrementato il valore della sequenza.
Questa seconda scelta è stata preferita rispetto alla prima in quanto più sicura ed affidabile, proprio per il fatto che i dati delle sequenze vengono scritti direttamente sul database, e quindi si ha una grossa affidabilità, sicuramente maggiore di quella di un semplice file gestito dal filesystem e sul webserver.

Tipi di Dato

I vari DBMS offrono all'utente una miriade di tipi di dato, per ogni esigenza e gusto, ma ovviamente questi ultimi differiscono nettamente da un sistema all'altro. Volendo rendere l'applicazione indipendente dal DBMS utilizzato, è stato quindi necessario utilizzare soltanto tipi di dato standard. Nonostante ciò, anche i tipi standard differiscono su alcuni aspetti, e sicuramente il tipo di dato più "personalizzato" dai vari DBMS è il tipo Date.
Per rendere quindi l'applicazione indipendente dal sistema database, abbiamo cercato di sfruttare al meglio l'API JDBC, utilizzando i metodi specifici per ciascun tipo di dato nelle query parametriche, lasciando il tutto trasparente all'utente. In particolare per l'inserimento di date da form, è stato utilizzato il formato standard italiano, giorno-mese-anno (dd-MM-yyyy).
Le stringhe inserite dall'utente (la cui corretta formattazione viene controllata lato client tramite javascript e regular expression) vengono processate nell'applicazione tramite Formatter Java, che producono oggetti di tipo java.util.Date, poi convertiti in oggetti di tipo java.sql.Date per essere utilizzati nei metodi JDBC specifici.

Invio e-Mail

Non potendo utilizzare package specifici dei singoli DBMS per l'invio delle email, è stata scritta un'apposita classe "MailSender" che si mette a disposizione il metodo statico

```
sendmail( String from, String to, String subject, String body )
```

In particolare la classe dialoga usando il protocollo SMTP con un mail server, fino a ricevere la conferma di mail spedita.

Import / Export da/su file XML

Per completare il quadro dell'indipendenza dell'applicazione dal DBMS si è optato per la realizzazione di un meccanismo di import/export dei dati del database in un file XML con una

grammatica DTD semplice ed immediata scritta appositamente, così da poter spostare con semplicità i dati da un database all'altro oltre che effettuare eventuali copie di backup ed avere i dati in un formato facilmente leggile.

A questo scopo è stata creata la classe Java/JDBC ImportExport, che mette a disposizione i metodi statici exportData(String file) ed importData(String file).

Figura 1: Struttura file di esportazione – type è il valore intero del tipo di dato, secondo la classe java.sql.Types

```
<db_import_export>
  <table name="tablename">
      <row>
            <data name="attributo1" type="1">valoreattributo1</data>
            <data name="attributo2" type="2">valoreattributo2</data>
      </row>
  </table>
</db_import_export>
```

Script SQL

Script per la creazione della Base di dati

In seguito è riportato lo script SQL con le istruzioni DDL necessarie per la definizione dello schema. Da notare l'uso di SQL standard, per la compatibilità con i vari sistemi DBMS.

```
CREATE TABLE Personale (
        matricola INTEGER not null,
        codice_fiscale VARCHAR(16),
        nome VARCHAR(25),
        cognome VARCHAR(25),

        PRIMARY KEY (matricola)
);

CREATE TABLE Account (
        username VARCHAR(32) not null,
        password VARCHAR(32) not null,
        email VARCHAR(50) not null,
        data_registrazione DATE,
        tipologia VARCHAR(30),
                CHECK (tipologia IN ('amministratore', 'addetto amministrativo', 'dipendente')),
        personale INTEGER,

        PRIMARY KEY (username),
        CONSTRAINT account_fk_personale
                FOREIGN KEY (personale) REFERENCES Personale(matricola)
);

CREATE TABLE  Fornitore (
        partita_iva VARCHAR(11) not null,
        nome_organizzazione VARCHAR(20) not null,
        tipologia VARCHAR(20),
                CHECK (tipologia IN ('banca', 'azienda', 'ente pubblico', 'persona fisica')),
        telefono VARCHAR(20),
        email VARCHAR(40),
        indirizzo VARCHAR(50),

        PRIMARY KEY ( partita_iva )
);

CREATE TABLE CategoriaBene (
        sigla VARCHAR(2) not null,
                CHECK (sigla IN ('MA', 'MB', 'SA')),
        nome VARCHAR(50),

        PRIMARY KEY ( sigla )
);
```

```
CREATE TABLE SottoCategoriaBene (
        codice VARCHAR(10) not null,
        nome VARCHAR(20),
        categoria_bene VARCHAR(2),

        PRIMARY KEY ( codice ),
        CONSTRAINT sottocategoria_fk_categoria
                FOREIGN KEY ( categoria_bene ) REFERENCES CategoriaBene ( sigla )
);

CREATE TABLE Bene (
        numero_inventario_generico INTEGER NOT NULL,
        numero_inventario_seriale VARCHAR(10),
        importo INTEGER,
        data_acquisto DATE,
        garanzia VARCHAR(50),
        data_attivazione DATE,
        data_scadenza DATE,
        targhetta VARCHAR(30),
        descrizione VARCHAR(100),
        conforme CHAR(1),
                CHECK (conforme IN ('S', 'N')),
        obsoleto CHAR(1),
                CHECK (obsoleto IN ('S', 'N')),
        sotto_categoria_bene VARCHAR(10),
        fornitore VARCHAR(11),

        PRIMARY KEY (numero_inventario_generico),
        CONSTRAINT bene_fk_sottocategoria
                FOREIGN KEY (sotto_categoria_bene) REFERENCES SottoCategoriaBene(codice),
        CONSTRAINT bene_fk_fornitore
                FOREIGN KEY (fornitore) REFERENCES Fornitore (partita_iva)
);

CREATE TABLE Bando (
        codice VARCHAR(10) NOT NULL,
        legge VARCHAR(100),
        denominazione VARCHAR(50),
        data_bando DATE,
        percentuale_finanziamento VARCHAR(10),

        PRIMARY KEY (codice)
);

CREATE TABLE Finanziamento(
        bene INTEGER,
        bando VARCHAR(10) NOT NULL,
        numero_progressivo INTEGER NOT NULL,

        PRIMARY KEY (bando, numero_progressivo),
```

```
        CONSTRAINT finanziamento_fk_bando
                FOREIGN KEY ( bando ) REFERENCES Bando(codice),
        CONSTRAINT finanziamento_fk_bene
                FOREIGN KEY ( bene ) REFERENCES Bene ( numero_inventario_generico )
);

CREATE TABLE Dotazione (
        codice VARCHAR(10) NOT NULL,
        personale INTEGER,
        bene INTEGER,
        data_inizio DATE,
        data_fine DATE,
        note VARCHAR(100),

        PRIMARY KEY ( codice ),
        CONSTRAINT dotazione_fk_personale
                FOREIGN KEY ( personale ) REFERENCES Personale ( matricola ),
        CONSTRAINT dotazione_fk_bene
                FOREIGN KEY ( bene ) REFERENCES Bene ( numero_inventario_generico ),
        CONSTRAINT dotazione_check_data
                CHECK (data_inizio < data_fine)
);

CREATE TABLE GruppoDiLavoro (
        codice VARCHAR(10) NOT NULL,
        denominazione VARCHAR(20),

        PRIMARY KEY ( codice )
);

CREATE TABLE Assegnazione (
        codice VARCHAR(10) NOT NULL,
        gruppo_di_lavoro VARCHAR(10),
        bene INTEGER,
        data_inizio DATE,
        data_fine DATE,
        note VARCHAR(100),

        PRIMARY KEY ( codice ),
        CONSTRAINT assegnazione_fk_gruppo
                FOREIGN KEY ( gruppo_di_lavoro ) REFERENCES GruppoDiLavoro ( codice ),
        CONSTRAINT assegnazione_fk_bene
                FOREIGN KEY ( bene ) REFERENCES Bene ( numero_inventario_generico ),
        CONSTRAINT assegnazione_check_data
                CHECK (data_inizio < data_fine)
);

CREATE TABLE Richiesta(
        codice INTEGER NOT NULL,
        personale INTEGER,
        sotto_categoria_bene VARCHAR(10),
```

```
        data DATE,
        motivazione VARCHAR(400),
        esito CHAR(1),
                CHECK (esito IN ('S', 'N', '-')),

        PRIMARY KEY ( codice ),
        CONSTRAINT richiesta_fk_personale
                FOREIGN KEY ( personale ) REFERENCES Personale(matricola),
        CONSTRAINT richiesta_fk_sottocategoria
                FOREIGN KEY ( sotto_categoria_bene ) REFERENCES SottoCategoriaBene( codice )
);

CREATE TABLE Allocazione(
        codice VARCHAR(10) NOT NULL,
        personale INTEGER,
        gruppo_di_lavoro VARCHAR(11),
        data_inizio DATE,
        data_fine DATE,

        PRIMARY KEY(codice),
        CONSTRAINT allocazione_fk_personale
                FOREIGN KEY ( personale ) REFERENCES Personale(matricola),
        CONSTRAINT allocazione_fk_gruppo
                FOREIGN KEY ( gruppo_di_lavoro ) REFERENCES GruppoDiLavoro( codice ),
        CONSTRAINT allocazione_check_data
                CHECK (data_inizio < data_fine)
);

CREATE TABLE Stanza(
        codice VARCHAR(10) NOT NULL,
        denominazione VARCHAR(11),
        posizione VARCHAR(10),
        note VARCHAR(50),

        PRIMARY KEY (codice)
);

CREATE TABLE Ubicazione(
        codice VARCHAR(10) NOT NULL,
        bene INTEGER,
        stanza VARCHAR(10),
        data_inizio DATE,
        data_fine DATE,

        PRIMARY KEY (codice),
        CONSTRAINT ubicazione_fk_bene
                FOREIGN KEY ( bene ) REFERENCES Bene( numero_inventario_generico ),
        CONSTRAINT ubicazione_fk_stanza
                FOREIGN KEY ( stanza ) REFERENCES Stanza( codice ),
        CONSTRAINT ubicazione_check_data
                CHECK (data_inizio < data_fine)
```

```
);

CREATE TABLE Postazione(
        codice VARCHAR(10) NOT NULL,
        personale INTEGER,
        stanza VARCHAR(10),
        data_inizio DATE,
        data_fine DATE,

        PRIMARY KEY(codice),
        CONSTRAINT postazione_fk_personale
                FOREIGN KEY ( personale ) REFERENCES Personale( matricola ),
        CONSTRAINT postazione_fk_stanza
                FOREIGN KEY ( stanza ) REFERENCES Stanza( codice ),
        CONSTRAINT postazione_check_data
                CHECK (data_inizio < data_fine)
);

-- Tabella per la gestione delle sequenze --
CREATE TABLE sequences (
        name VARCHAR(20) PRIMARY KEY,
        startval INTEGER NOT NULL,
        incr INTEGER,
        minval INTEGER,
        maxval INTEGER,
        currval INTEGER,
        cycle CHAR(1) CHECK (cycle IN ('T', 'F'))
);
```

Script per l'eliminazione dello schema.

```
ALTER TABLE Account drop CONSTRAINT account_fk_personale;
ALTER TABLE Finanziamento drop CONSTRAINT finanziamento_fk_bando;
ALTER TABLE Finanziamento drop CONSTRAINT finanziamento_fk_bene;
ALTER TABLE Dotazione drop CONSTRAINT dotazione_fk_personale;
ALTER TABLE Dotazione drop CONSTRAINT dotazione_fk_bene;
ALTER TABLE Assegnazione drop CONSTRAINT assegnazione_fk_gruppo;
ALTER TABLE Assegnazione drop CONSTRAINT assegnazione_fk_bene;
ALTER TABLE Richiesta drop CONSTRAINT richiesta_fk_personale;
ALTER TABLE Richiesta drop CONSTRAINT richiesta_fk_sottocategoria;
ALTER TABLE Allocazione drop CONSTRAINT allocazione_fk_personale;
ALTER TABLE Allocazione drop CONSTRAINT allocazione_fk_gruppo;
ALTER TABLE Ubicazione drop CONSTRAINT ubicazione_fk_bene;
ALTER TABLE Ubicazione drop CONSTRAINT ubicazione_fk_stanza;
ALTER TABLE Postazione drop CONSTRAINT postazione_fk_personale;
ALTER TABLE Postazione drop CONSTRAINT postazione_fk_stanza;
ALTER TABLE SottoCategoriaBene drop CONSTRAINT sottocategoria_fk_categoria;
ALTER TABLE Bene drop CONSTRAINT bene_fk_sottocategoria;
ALTER TABLE Bene drop CONSTRAINT bene_fk_fornitore;

DROP TABLE Account;
DROP TABLE Finanziamento;
DROP TABLE Dotazione;
DROP TABLE Assegnazione;
DROP TABLE Richiesta;
DROP TABLE Allocazione;
DROP TABLE Ubicazione;
DROP TABLE Postazione;
DROP TABLE Personale;
DROP TABLE Fornitore;
DROP TABLE CategoriaBene;
DROP TABLE Bando;
DROP TABLE GruppoDiLavoro;
DROP TABLE Stanza;
DROP TABLE SottoCategoriaBene;
DROP TABLE Bene;
DROP TABLE sequences;
```

Script per le statistiche

Le statistiche sono state realizzate attraverso l'implementazione di stored function. Questa è l'unica parte del progetto dipendente dal DBMS, prima di tutto in quanto non tutti i DBMS supportano le funzioni e procedure, e secondariamente in quanto era richiesto esplicitamente l'utilizzo di stored procedure o function (le statistiche seguenti sarebbero potute essere realizzate anche direttamente in java, senza utilizzare il DBMS). In particolare ci siamo preoccupati di creare le funzioni per il DBMS PostgreSQL (nel linguaggio PL/pgSQL) e per il sistema Oracle (nel linguaggio PL/SQL).
Nota: le differenze tra i due linguaggi sono poche e riguardanti soltanto la parte sintattica.

```
-- statistica 1 - dato un bando, ritorna la somma degli importi dei beni finanziati attraverso
questo bando --

-- PL/SQL --
CREATE OR REPLACE FUNCTION statistica1( bando IN VARCHAR )
RETURN NUMBER IS
somma NUMBER;

BEGIN
	SELECT sum(importo) INTO somma
	FROM bene B JOIN finanziamento F
		ON B.numero_inventario_generico = F.bene
	WHERE F.bando = bando;

	RETURN somma;
END;

-- PL/pgSQL --
CREATE OR REPLACE FUNCTION statistica1( bando IN VARCHAR )
RETURNS INTEGER AS $$
DECLARE somma INTEGER;

BEGIN
	SELECT sum(importo) INTO somma
	FROM bene B JOIN finanziamento F
		ON B.numero_inventario_generico = F.bene
	WHERE F.bando = bando;

	RETURN somma;
END;
$$ LANGUAGE plpgsql;


-- statistica 2 - data una sotto categoria ritorna il fornitore che ha fornito più beni di questa
sotto categoria --
```

```
-- PL/SQL --
CREATE OR REPLACE FUNCTION statistica2( sottocategoria IN VARCHAR )
RETURN VARCHAR IS
forn VARCHAR(11);

BEGIN
        SELECT fornitore INTO forn
        FROM bene
        WHERE sotto_categoria_bene = sottocategoria
        GROUP BY fornitore
        HAVING count(numero_inventario_generico) >= ALL (
                SELECT count(numero_inventario_generico)
                FROM bene
                WHERE sotto_categoria_bene = sottocategoria
                GROUP BY fornitore
        );

        RETURN forn;
END;

-- PL/pgSQL --
CREATE OR REPLACE FUNCTION statistica2( sottocategoria IN VARCHAR )
RETURNS VARCHAR AS $$
DECLARE
        forn VARCHAR(11);

BEGIN
        SELECT fornitore INTO forn
        FROM bene
        WHERE sotto_categoria_bene = sottocategoria
        GROUP BY fornitore
        HAVING count(numero_inventario_generico) >= ALL (
                SELECT count(numero_inventario_generico)
                FROM bene
                WHERE sotto_categoria_bene = sottocategoria
                GROUP BY fornitore
        );

        RETURN forn;
END;
$$ LANGUAGE plpgsql;

-- alternativa con cursore: --
-- PL/SQL --
CREATE OR REPLACE FUNCTION statistica2( sottocategoria IN VARCHAR )
RETURN VARCHAR IS
forn VARCHAR(11);
nummax NUMBER;
CURSOR c1 IS
        SELECT count(numero_inventario_generico) AS numbeni, fornitore
        FROM bene
```

```
        WHERE sotto_categoria_bene = sottocategoria
        GROUP BY fornitore;

BEGIN
        nummax := 0;
        forn := 'nessuno';
        FOR rec IN c1
        LOOP
                IF rec.numbeni > nummax THEN
                        nummax := rec.numbeni;
                        forn := rec.fornitore;
                END IF;
        END LOOP;

        RETURN forn;
END;
-- PL/pgSQL --
CREATE OR REPLACE FUNCTION statistica2( sottocategoria IN VARCHAR )
RETURNS VARCHAR AS $$
DECLARE
        forn VARCHAR;
        nummax INTEGER;
        c1 CURSOR IS
         SELECT count(numero_inventario_generico) AS numbeni, fornitore
         FROM bene
         WHERE sotto_categoria_bene = sottocategoria
         GROUP BY fornitore;

BEGIN
        nummax := 0;
        forn := 'nessuno';
        FOR rec IN c1
        LOOP
                IF rec.numbeni > nummax THEN
                        nummax := rec.numbeni;
                        forn := rec.fornitore;
                END IF;
        END LOOP;

        RETURN forn;
END; $$ LANGUAGE plpgsql;
-- statistica 3 --
-- il fornitore che fornito un bene con l'importo  piu' basso o piu' alto, di una data
sottocategoria e in una dato periodo  --
-- PL/SQL --
CREATE OR REPLACE FUNCTION statistica3( sottocategoria IN VARCHAR, datainizio IN DATE, datafine IN
DATE )
RETURN VARCHAR IS
forn VARCHAR(11);

BEGIN
```

```
        SELECT fornitore INTO forn
        FROM bene
        WHERE sotto_categoria_bene = sottocategoria
                AND data_acquisto >= datainizio
                AND data_acquisto <= datafine
                AND (importo >= ALL (
                        SELECT importo
                        FROM bene
                        WHERE sotto_categoria_bene = sottocategoria
                ) OR importo <= ALL (
                        SELECT importo
                        FROM bene
                        WHERE sotto_categoria_bene = sottocategoria
                ));

        RETURN forn;
END;
-- statistica3.1: fornitore che ha fornito un bene della sotto categoria data con l'importo più
basso nel periodo specificato
-- PL/SQL --
CREATE OR REPLACE FUNCTION statistica31( sottocategoria IN VARCHAR, datainizio IN DATE, datafine IN
DATE )
RETURN VARCHAR IS
forn VARCHAR(11);

BEGIN
        SELECT fornitore INTO forn
        FROM bene
        WHERE sotto_categoria_bene = sottocategoria
                AND data_acquisto >= datainizio AND data_acquisto <= datafine
                AND importo <= ALL (
                        SELECT importo
                        FROM bene
                        WHERE sotto_categoria_bene = sottocategoria
                );

        RETURN forn;
END;
-- PL/pgSQL --
CREATE OR REPLACE FUNCTION statistica31( sottocategoria IN VARCHAR, datainizio IN DATE, datafine IN
DATE )
RETURNS VARCHAR AS $$
DECLARE
        forn VARCHAR(11);

BEGIN
        SELECT fornitore INTO forn
        FROM bene
        WHERE sotto_categoria_bene = sottocategoria
                AND data_acquisto >= datainizio AND data_acquisto <= datafine
                AND importo <= ALL (
```

```
                        SELECT importo
                        FROM bene
                        WHERE sotto_categoria_bene = sottocategoria
                );

        RETURN forn;
END;
$$ language plpgsql;

-- statistica3.2: fornitore che ha fornito un bene della sotto categoria data con l'importo più alto nel periodo specificato
-- PL/SQL --
CREATE OR REPLACE FUNCTION statistica32( sottocategoria IN VARCHAR, datainizio IN DATE, datafine IN DATE )
RETURN VARCHAR IS
forn VARCHAR(11);

BEGIN
        SELECT fornitore INTO forn
        FROM bene
        WHERE sotto_categoria_bene = sottocategoria
                AND data_acquisto >= datainizio AND data_acquisto <= datafine
                AND importo >= ALL (
                        SELECT importo
                        FROM bene
                        WHERE sotto_categoria_bene = sottocategoria
                );

        RETURN forn;
END;
```

```
-- PL/pgSQL --
CREATE OR REPLACE FUNCTION statistica32( sottocategoria IN VARCHAR, datainizio IN DATE, datafine IN DATE )
RETURNS VARCHAR AS $$
DECLARE
        forn VARCHAR(11);

BEGIN
        SELECT fornitore INTO forn
        FROM bene
        WHERE sotto_categoria_bene = sottocategoria
                AND data_acquisto >= datainizio AND data_acquisto <= datafine
                AND importo >= ALL (
                        SELECT importo
                        FROM bene
                        WHERE sotto_categoria_bene = sottocategoria
```

```
                );

        RETURN forn;
END;
$$ language plpgsql;
-- statistica 4 - visualizzare la percentuale dei beni acquistati in un dato periodo,  --
-- di una data categoria, coperti da un bando di finanziamento. --
-- PL/SQL --
CREATE OR REPLACE FUNCTION statistica4( categoria IN VARCHAR, datainizio IN DATE, datafine IN DATE )
RETURN VARCHAR IS
percent NUMBER;
benitot NUMBER;
benifinanziati NUMBER;

BEGIN
        SELECT count(*) INTO benitot
        FROM bene B JOIN sottocategoriabene S
                ON B.sotto_categoria_bene = S.codice
        WHERE S.categoria_bene = categoria
                AND B.data_acquisto >= datainizio
                AND B.data_acquisto <= datafine;

        SELECT count(*) into benifinanziati
        FROM bene B JOIN finanziamento F
                ON F.bene = B.numero_inventario_generico
                        JOIN sottocategoriabene S
                        ON B.sotto_categoria_bene = S.codice
                        WHERE S.categoria_bene = categoria
                                AND B.data_acquisto >= datainizio
                                AND B.data_acquisto <= datafine;

    IF benitot = 0 THEN
        percent := 0;
    ELSE
        percent := benifinanziati /  benitot * 100;
    END IF;

        RETURN percent || '%';
END;
```

```
-- PL/pgSQL --
CREATE OR REPLACE FUNCTION statistica4( categoria IN VARCHAR, datainizio IN DATE, datafine IN DATE )
RETURNS VARCHAR AS $$
DECLARE
        percent NUMERIC;
        benitot NUMERIC;
        benifinanziati NUMERIC;

BEGIN
        SELECT count(*) INTO benitot
        FROM bene B JOIN sottocategoriabene S
                ON B.sotto_categoria_bene = S.codice
        WHERE S.categoria_bene = categoria
                AND B.data_acquisto >= datainizio
                AND B.data_acquisto <= datafine;

        SELECT count(*) into benifinanziati
        FROM bene B JOIN finanziamento F
                ON F.bene = B.numero_inventario_generico
                JOIN sottocategoriabene S
                ON B.sotto_categoria_bene = S.codice
                        WHERE S.categoria_bene = categoria
                        AND B.data_acquisto >= datainizio
                        AND B.data_acquisto <= datafine;

    IF benitot = 0 THEN
        percent := 0;
    ELSE
        percent := benifinanziati /  benitot * 100;
    END IF;

        RETURN percent || '%';
END;
$$ language plpgsql;
```

Documentazione Tecnica

L'applicazione è stata realizzata utilizzando le tecnologie JSP e Java Servlet per la generazione dinamica di HTML in base alle richieste, l'API JBDC per la comunicazione col database, il linguaggio javascript per i controlli lato client della correttezza dei dati inseriti nei form e il software cron per l'esecuzione di processi batch.

L'accesso all'applicazione avviene attraverso una serlvet di login; quest'ultima verifica la presenza dell'utente nel database, la corrispondenza della password e infine, a seconda del tipo di utente, effettua un redirect verso la home page relativa. Se l'operazione di login va a buon fine viene creata una sessione, associandogli il tipo di utente e il suo identificativo (email); ciò consente di verificare in tutte le altre pagine che l'utente che vi accede sia abilitato. In questo modo ad esempio gli utenti di tipo dipendente non possono accedere alle pagine degli amministratori e così via. Per l'avvio della connessione con il database è stata scritta un'apposita classe che in base ai valori specificati nelle variabili *driver* e *url* (le uniche stringhe da modificare per passare da un DBMS ad un altro) crea la connessione se non presente o la restituisce se già esistente. Ogni volta che è necessario accedere alla base di dati viene richiesta la connessione e successivamente si procede alla creazione degli oggetti necessari.

Tutte le operazioni di aggiornamento e cancellazione che potrebbero causare violazioni di vincoli di integrità sono state gestite manualmente attraverso transazioni JDBC, agendo sulle tabelle referenziate soltanto dopo aver aggiornato quelle referenti.

Per un uso ottimale e semplice dell'applicazione, si è scelto uno schema di navigazione molto semplice ed intuitivo, in cui la maggior parte delle operazioni contestuali sono disponibili direttamente nella pagina interessata, senza passare attraverso altre pagine; si da inoltre all'utente la possibilità di filtrare i risultati delle varie ricerche e/o elenchi, nonché di ordinare i risultati in base ad alcuni campi semplicemente cliccando sul relativo header. Queste funzionalità sono state realizzate attraverso l'uso dei costrutti LIKE e ORDER BY offerti da SQL.

Processi Batch

La gestione di processi batch e l'effettuazione di operazioni in momenti precisi è stata implementata sfruttando il sotfware “cron” presente nei sistemi GNU/Linux, che consente di specificare tempi e cadenze in un cui un determinato programma dev'essere eseguito. Per controllare le scadenze dei beni e il report mensile sono state realizzate delle apposite classi:
CheckScadenze e Report.
In particolare la classe Report, avviata ogni 1° del mese, istanzia due oggetti java.sql.Date che rappresentano la data iniziale e la data finale del mese precedente, tenendo in considerazione i mesi di 30/31/28 giorni nonché gli anni bisestili.
Viene poi effettuata una query sul database, selezionando i beni acquistati in una data compresa tra quelle di cui sopra e i risultati vengono spediti in formato HTML a tutti gli amministratori e addetti amministrativi.
La classe CheckScadenze invece, avviata ogni giorno alle 13:30, controlla per ogni bene di categoria SA, la data di scadenza. Se la differenza tra questa e la data corrente è 30, 15 o 1 giorno, spedisce una mail di alert agli amministratori, riportando il numero inventario generico del bene in scadenza e il numero di giorni rimasti.

Figura 2: contenuto del file /etc/crontab

```
# controllo scadenze beni ogni giorno alle 13.30
30 13   * * *   root    checkScadenze.sh

# invio report beni inventariati ogni 1° del mese a mezzanotte
0 0     1 * *   root    report.sh
```

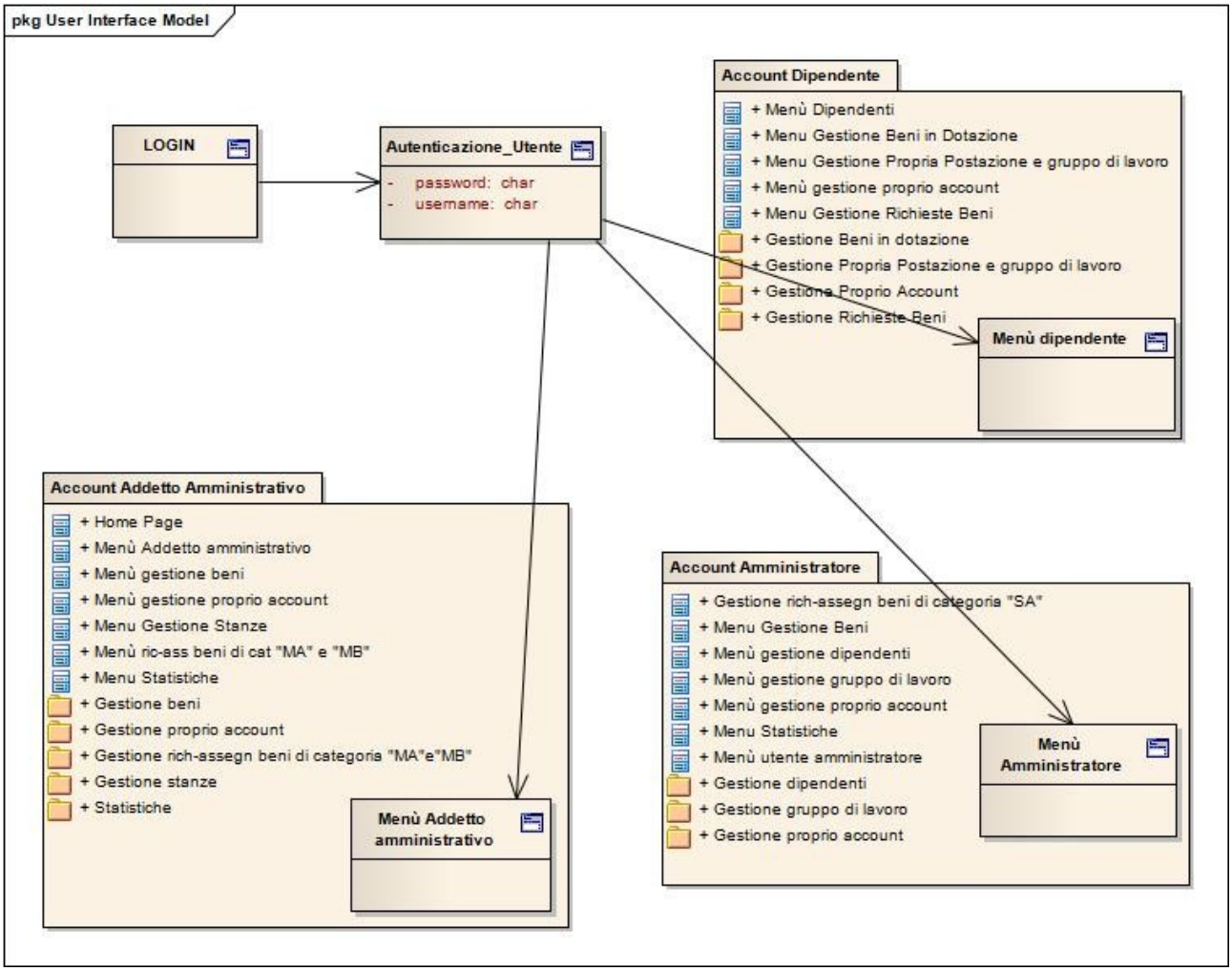
Figura 3: Script di avvio della classe CheckScadenze

```
#! /bin/sh

cd /percoso/fino/a/lmadb/
java -cp ./build/classes/:./lib/oracle.jar:./lib/postgres.jar: util/CheckScadenza
cd
```

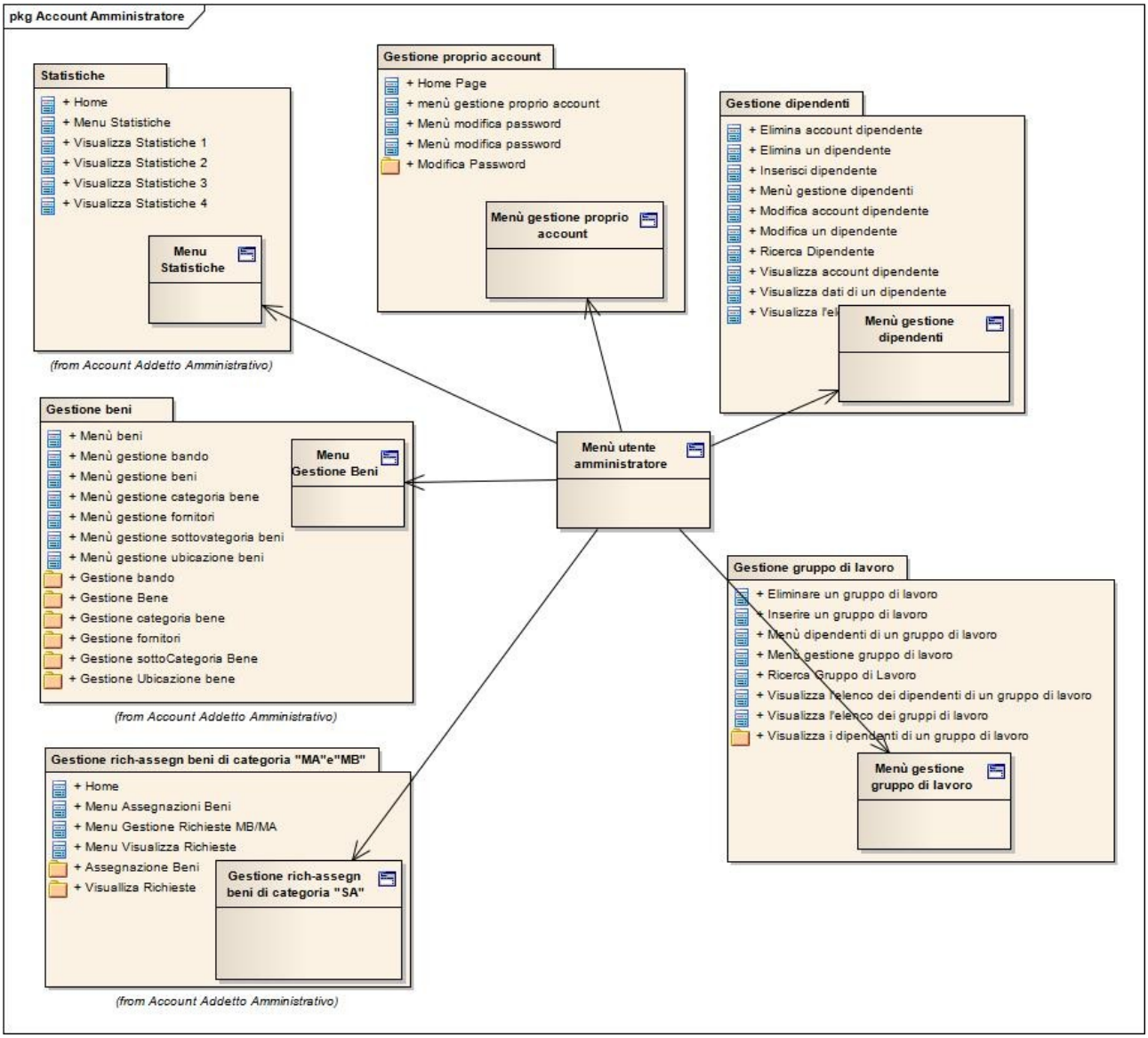
Figura 4: Script di avvio della classe Report

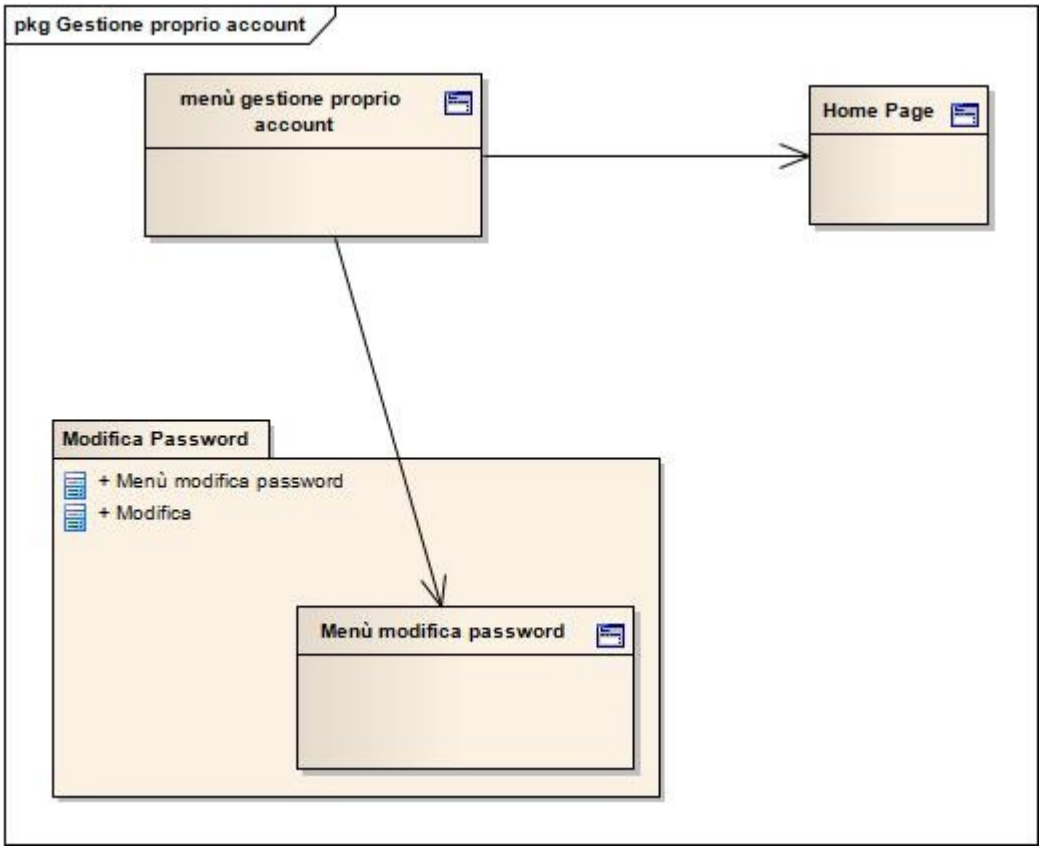
```
#! /bin/sh

cd /percoso/fino/a/lmadb/
java -cp ./build/classes/:./lib/oracle.jar:./lib/postgres.jar: util/Report
cd
```

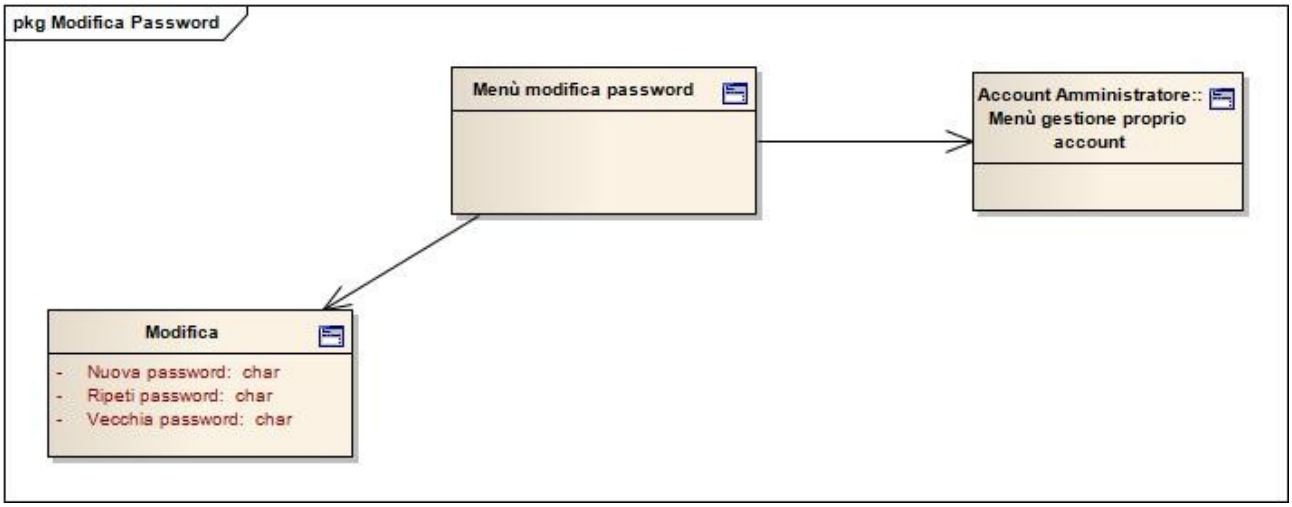
Schema interfacce

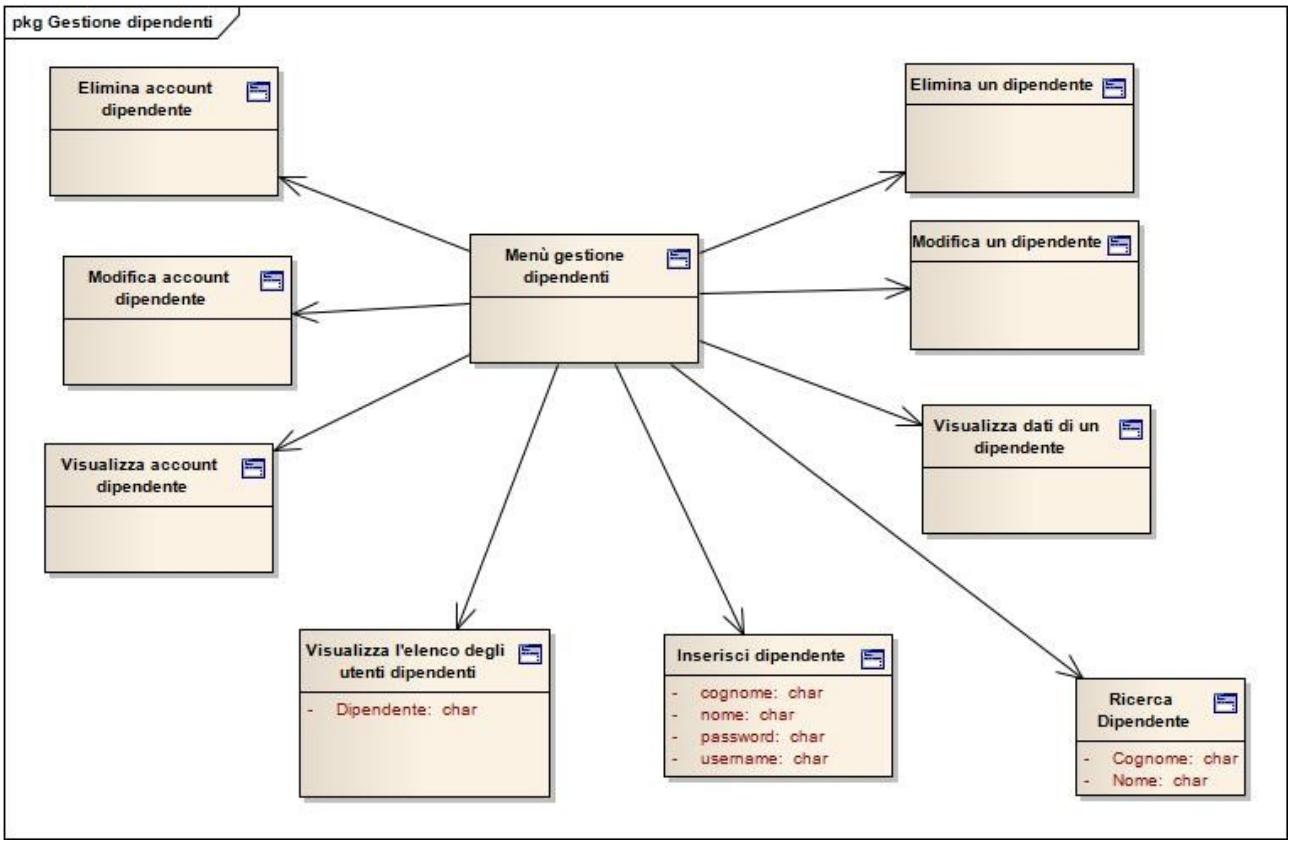


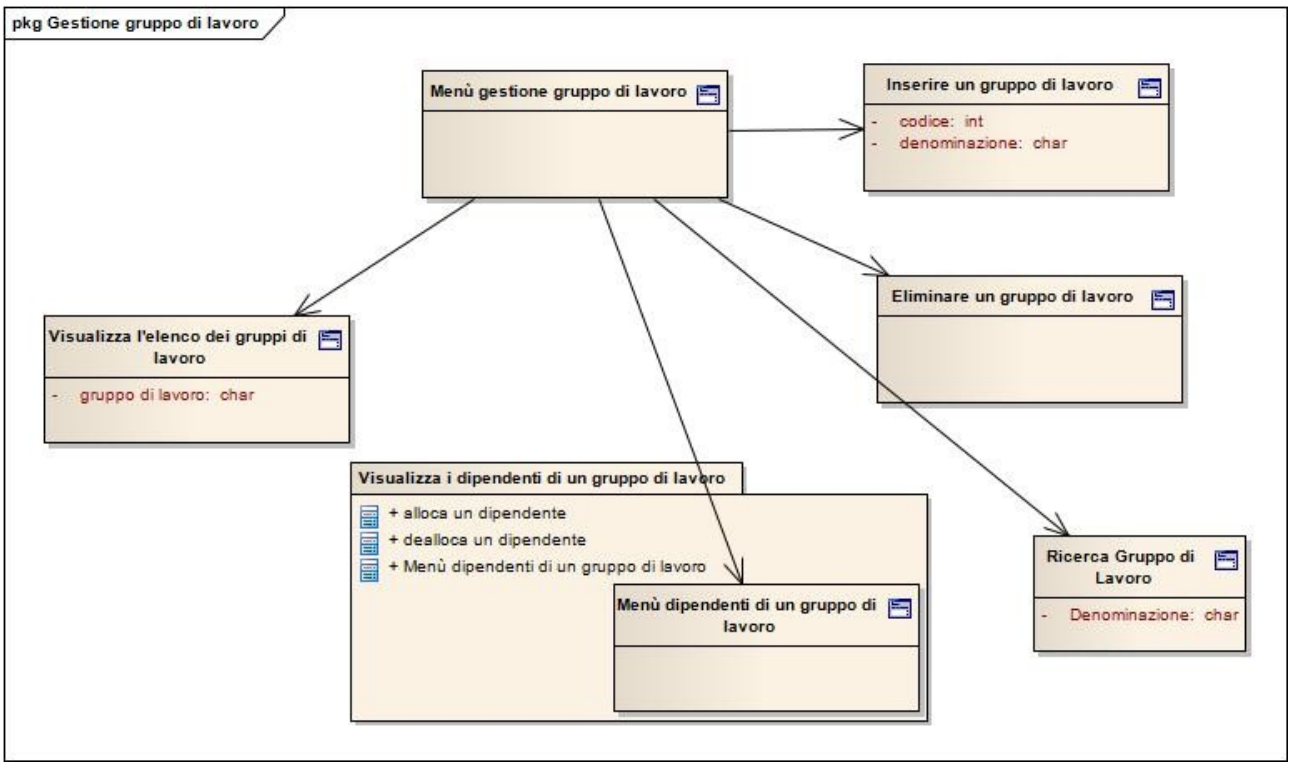
Casi d'uso Amministratore

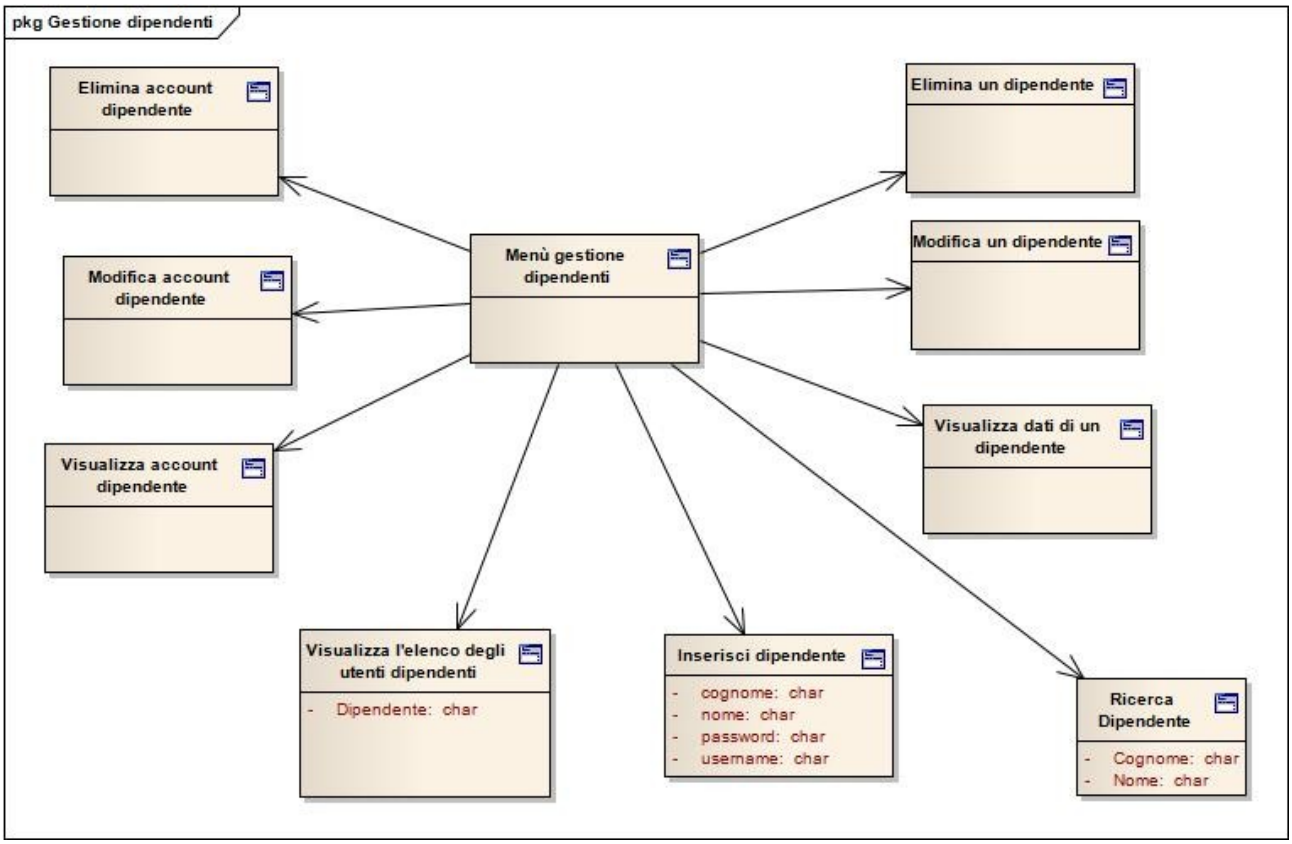



pkg Gestione proprio account
menù gestione proprio account
Home Page
Modifica Password
+ Menù modifica password
+ Modifica
Menù modifica password

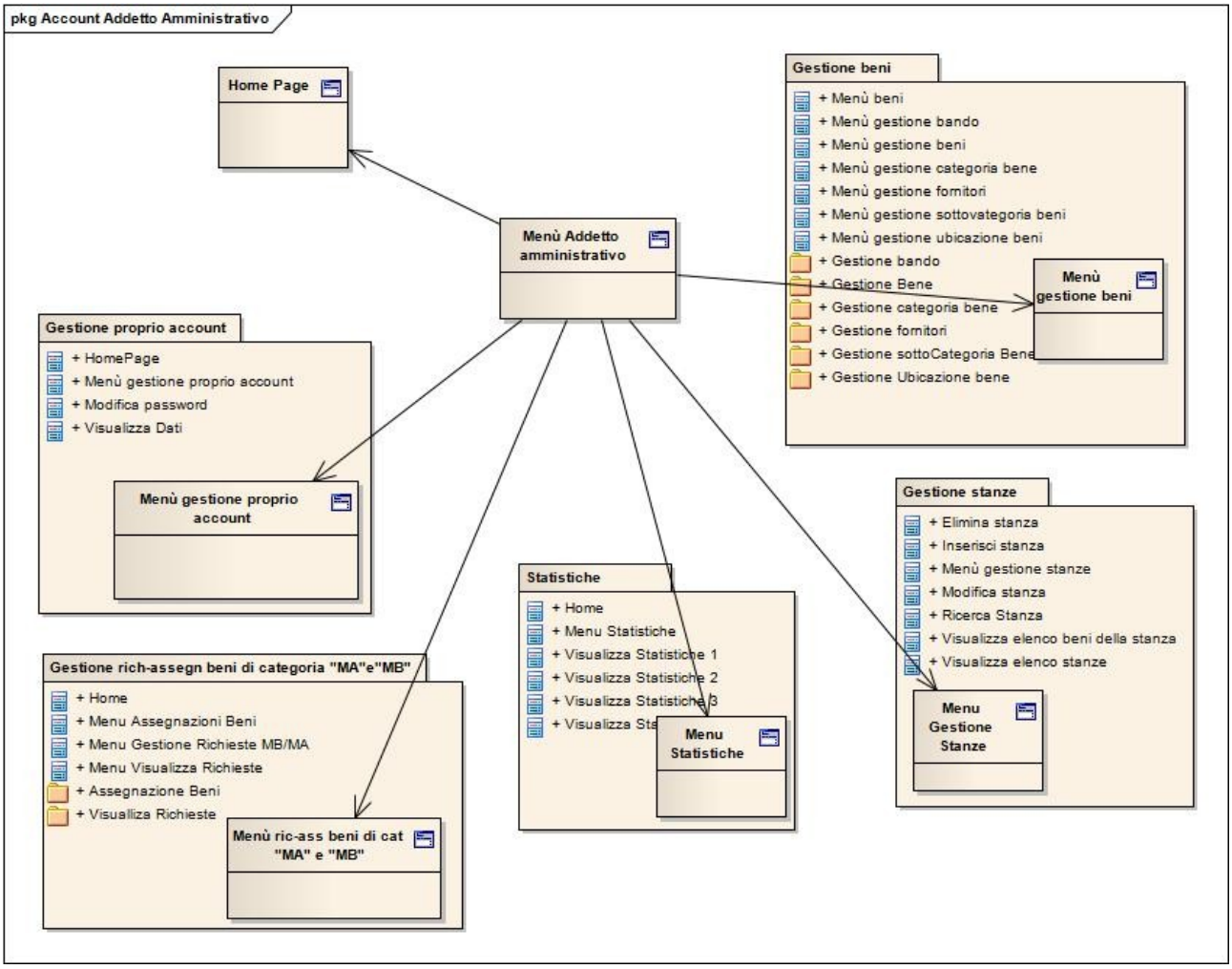

pkg Modifica Password
Menù modifica password
Account Amministratore:: Menù gestione proprio account
Modifica
- Nuova password: char
- Ripeti password: char
- Vecchia password: char

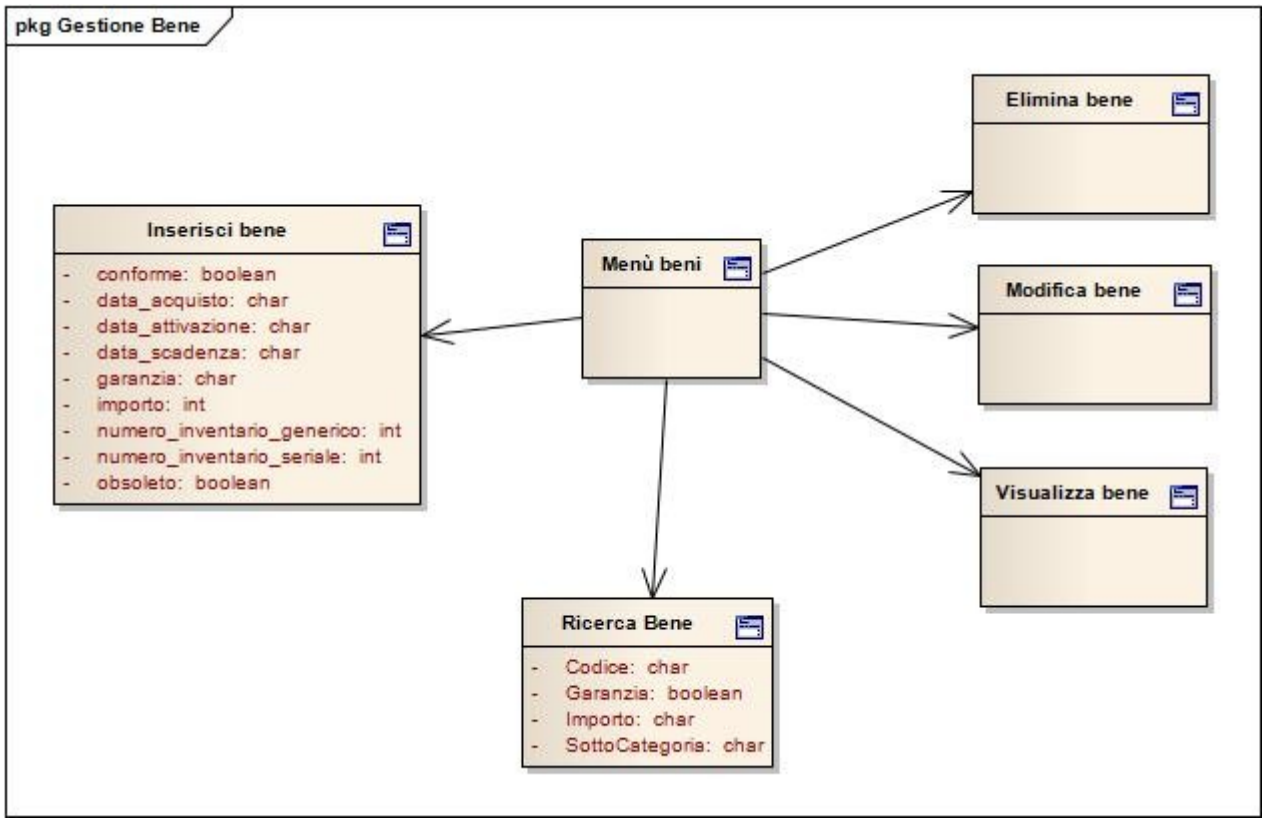

pkg Gestione dipendenti
Elimina account dipendente
Elimina un dipendente
Menù gestione dipendenti
Modifica un dipendente
Modifica account dipendente
Visualizza dati di un dipendente
Visualizza account dipendente
Visualizza l'elenco degli utenti dipendenti
- Dipendente: char
Inserisci dipendente
- cognome: char
- nome: char
- password: char
- username: char
Ricerca Dipendente
- Cognome: char
- Nome: char

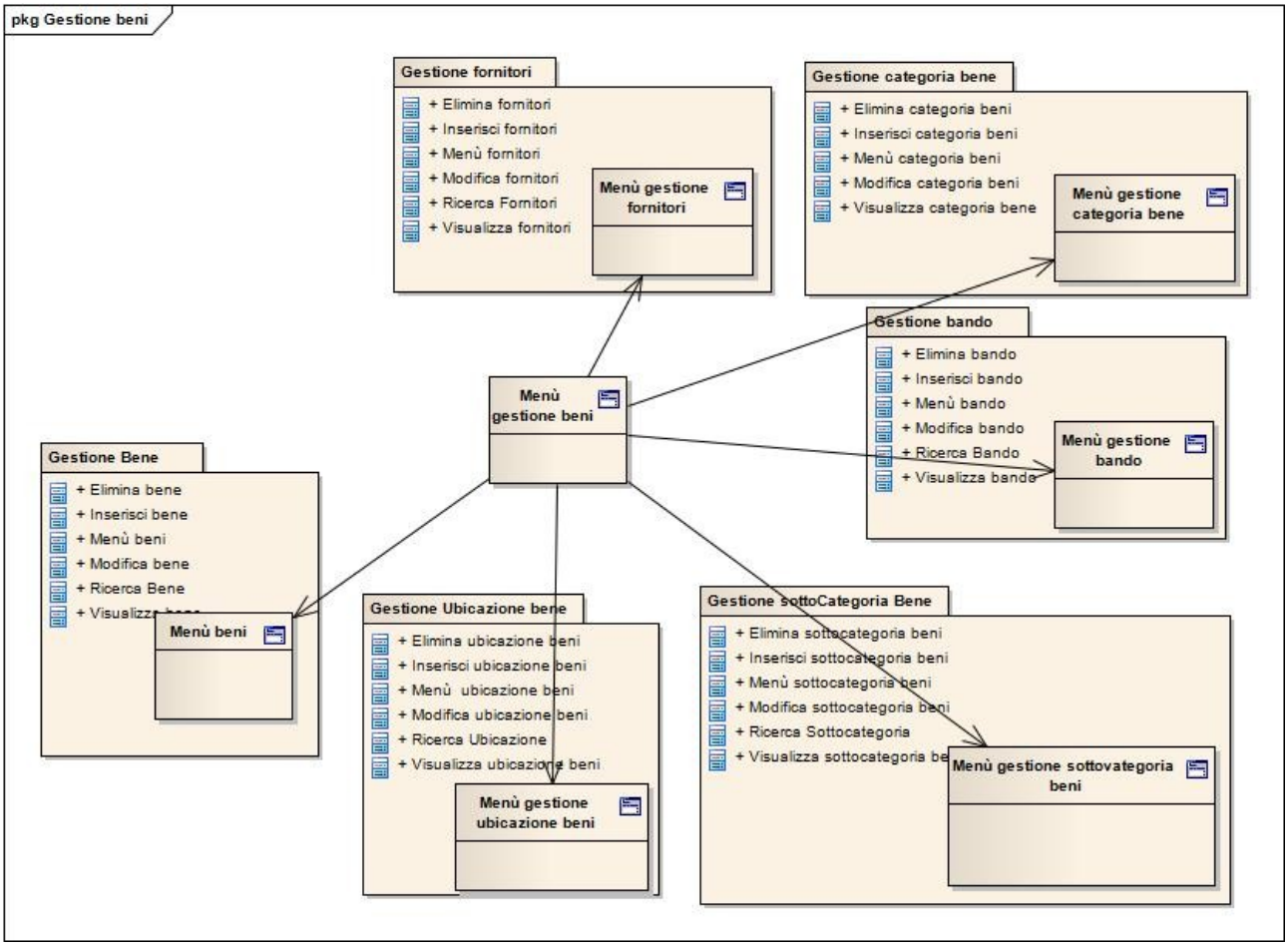

pkg Gestione gruppo di lavoro
Menù gestione gruppo di lavoro
Inserire un gruppo di lavoro
- codice: int
- denominazione: char
Eliminare un gruppo di lavoro
Visualizza l'elenco dei gruppi di lavoro
- gruppo di lavoro: char
Visualizza i dipendenti di un gruppo di lavoro
+ alloca un dipendente
+ dealloca un dipendente
+ Menù dipendenti di un gruppo di lavoro
Menù dipendenti di un gruppo di lavoro
Ricerca Gruppo di Lavoro
- Denominazione: char

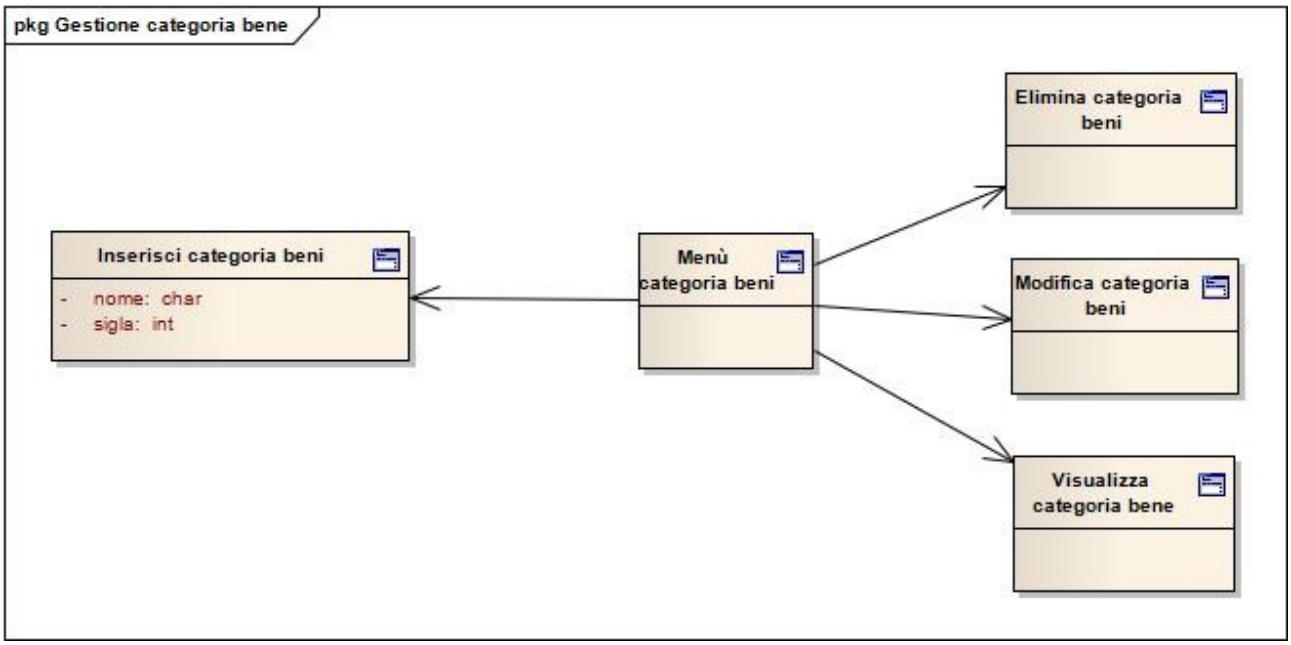

pkg Gestione dipendenti
Elimina account dipendente
Elimina un dipendente
Menù gestione dipendenti
Modifica account dipendente
Modifica un dipendente
Visualizza account dipendente
Visualizza dati di un dipendente
Visualizza l'elenco degli utenti dipendenti
- Dipendente: char
Inserisci dipendente
- cognome: char
- nome: char
- password: char
- username: char
Ricerca Dipendente
- Cognome: char
- Nome: char

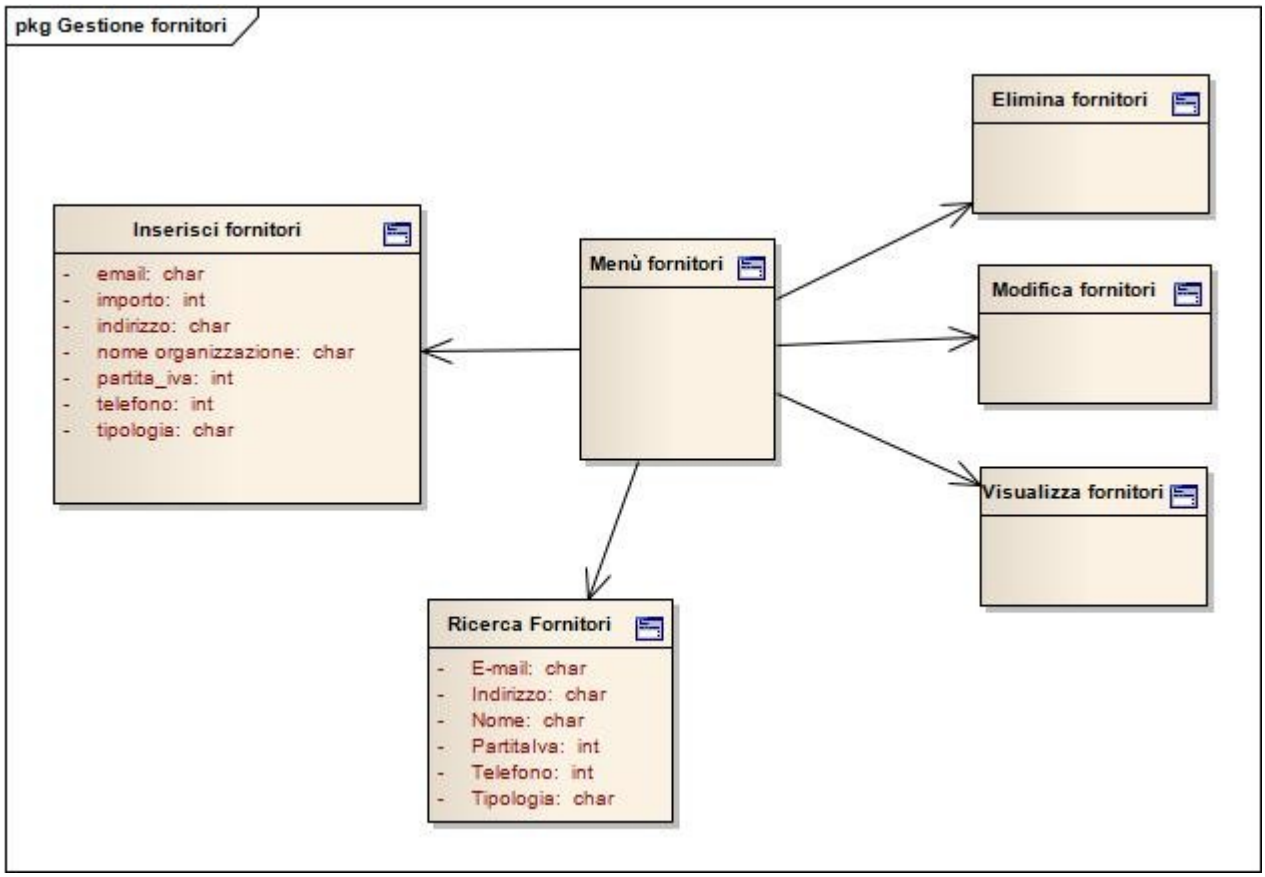
Casi d'uso Addetto Amministrativo

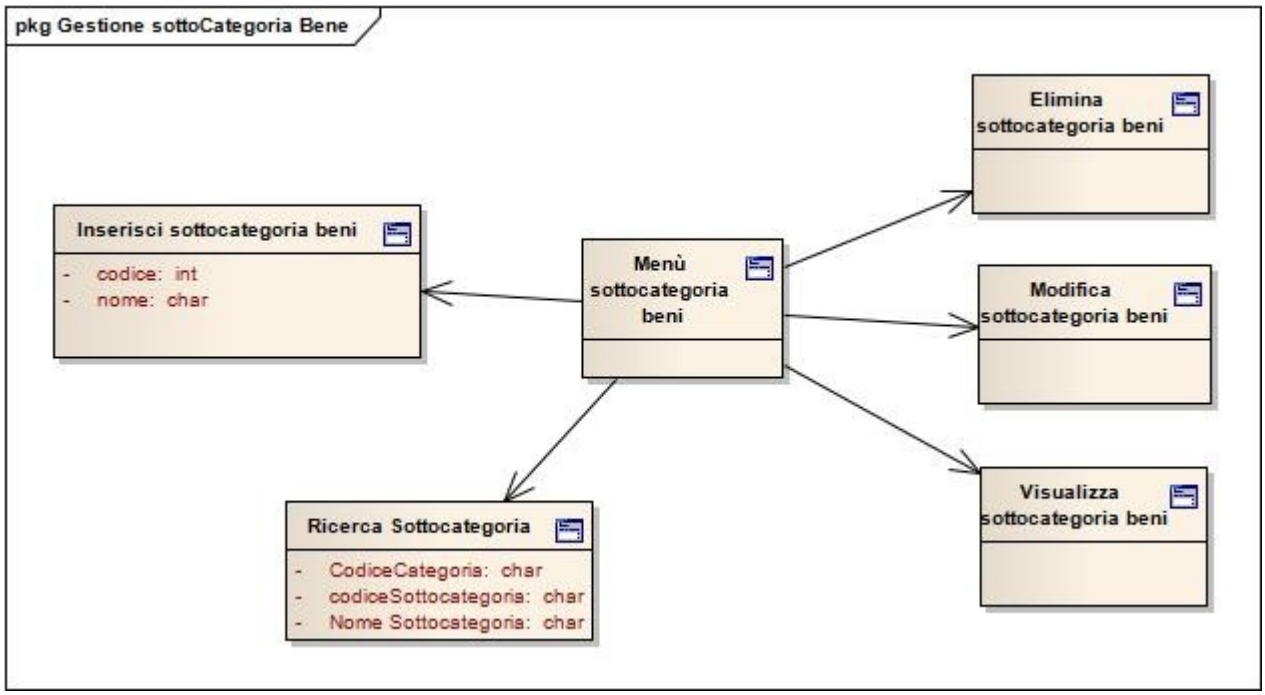


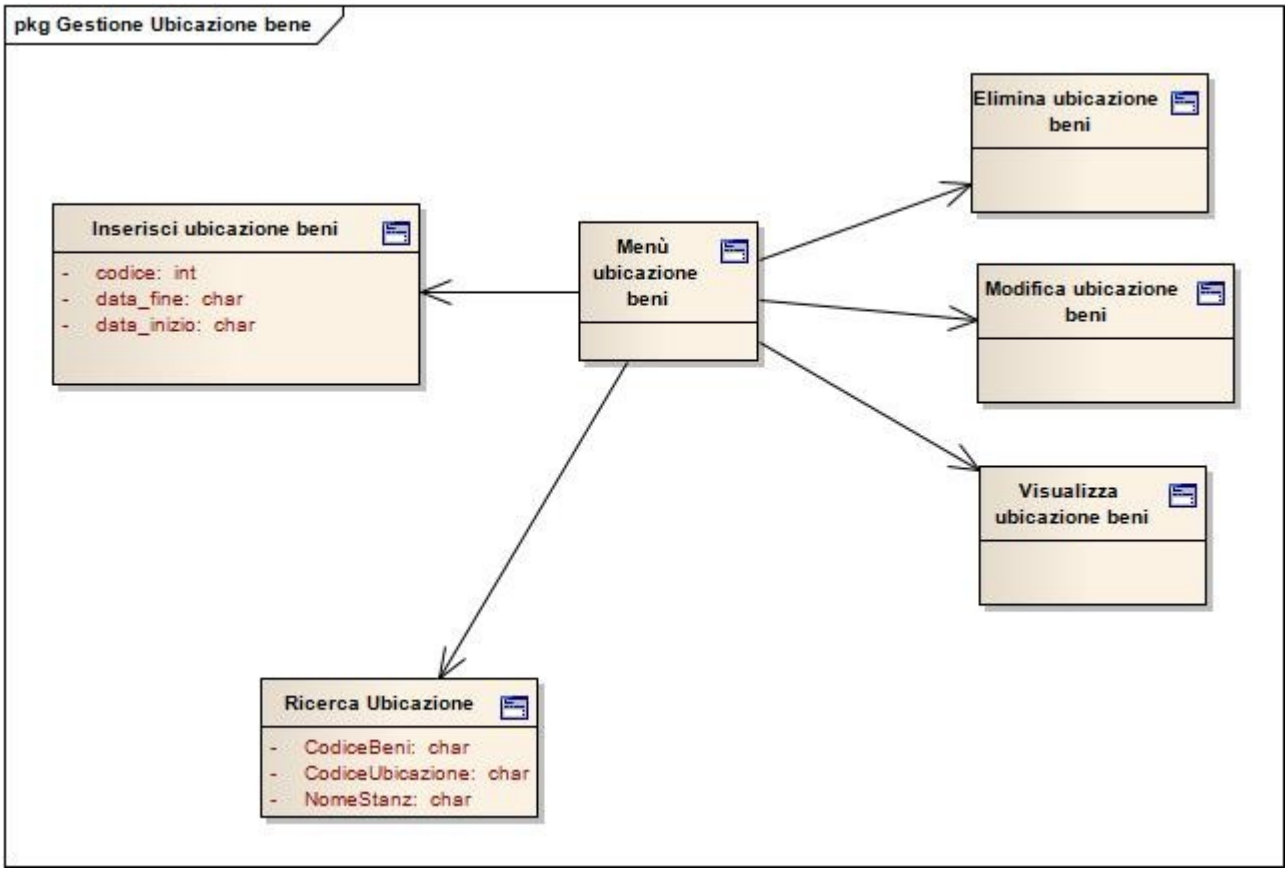

pkg Gestione Bene
Elimina bene
Inserisci bene
- conforme: boolean
- data_acquisto: char
- data_attivazione: char
- data_scadenza: char
- garanzia: char
- importo: int
- numero_inventario_generico: int
- numero_inventario_seriale: int
- obsoleto: boolean
Menù beni
Modifica bene
Visualizza bene
Ricerca Bene
- Codice: char
- Garanzia: boolean
- Importo: char
- SottoCategoria: char

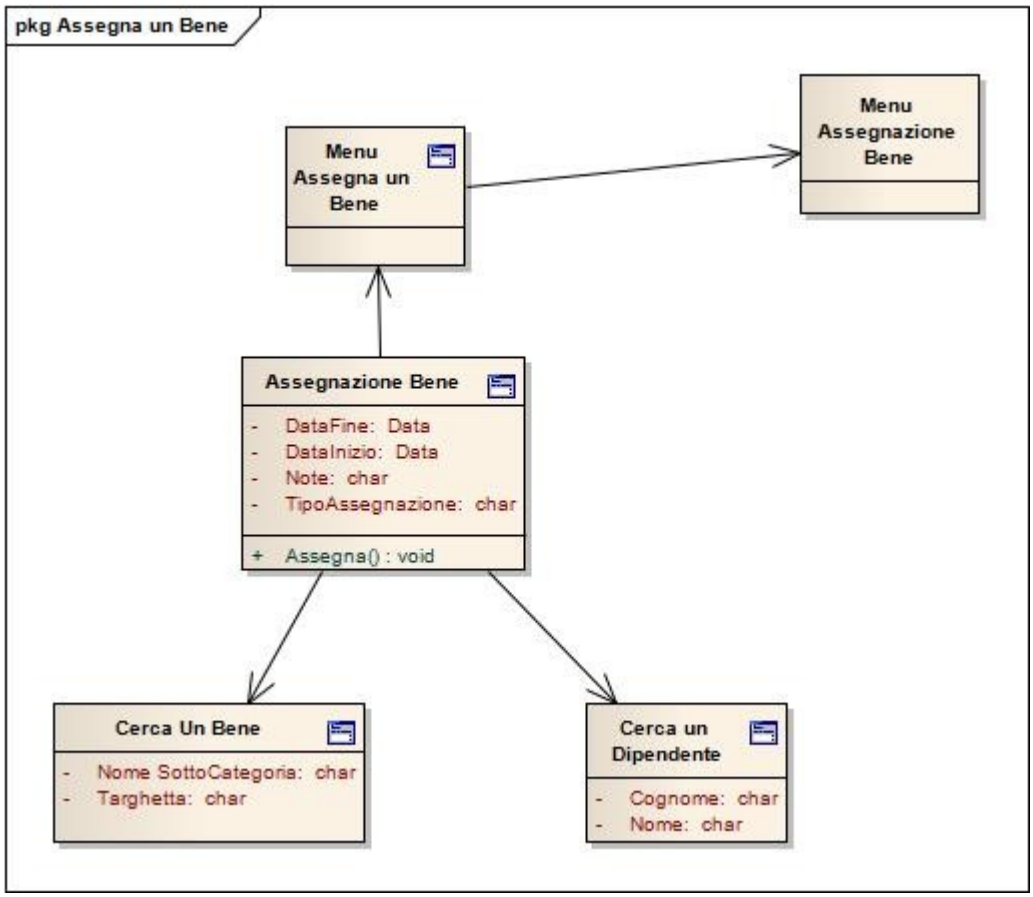

pkg Gestione beni
Gestione fornitori
+ Elimina fornitori
+ Inserisci fornitori
+ Menù fornitori
+ Modifica fornitori
+ Ricerca Fornitori
+ Visualizza fornitori
Menù gestione fornitori
Gestione categoria bene
+ Elimina categoria beni
+ Inserisci categoria beni
+ Menù categoria beni
+ Modifica categoria beni
+ Visualizza categoria bene
Menù gestione categoria bene
Gestione bando
+ Elimina bando
+ Inserisci bando
+ Menù bando
+ Modifica bando
+ Ricerca Bando
+ Visualizza bando
Menù gestione bando
Menù gestione beni
Gestione Bene
+ Elimina bene
+ Inserisci bene
+ Menù beni
+ Modifica bene
+ Ricerca Bene
Menù beni
Gestione Ubicazione bene
+ Elimina ubicazione beni
+ Inserisci ubicazione beni
+ Menù ubicazione beni
+ Modifica ubicazione beni
+ Ricerca Ubicazione
+ Visualizza ubicazione beni
Menù gestione ubicazione beni
Gestione sottoCategoria Bene
+ Elimina sottocategoria beni
+ Inserisci sottocategoria beni
+ Menù sottocategoria beni
+ Modifica sottocategoria beni
+ Ricerca Sottocategoria
+ Visualizza sottocategoria be
Menù gestione sottovategoria beni

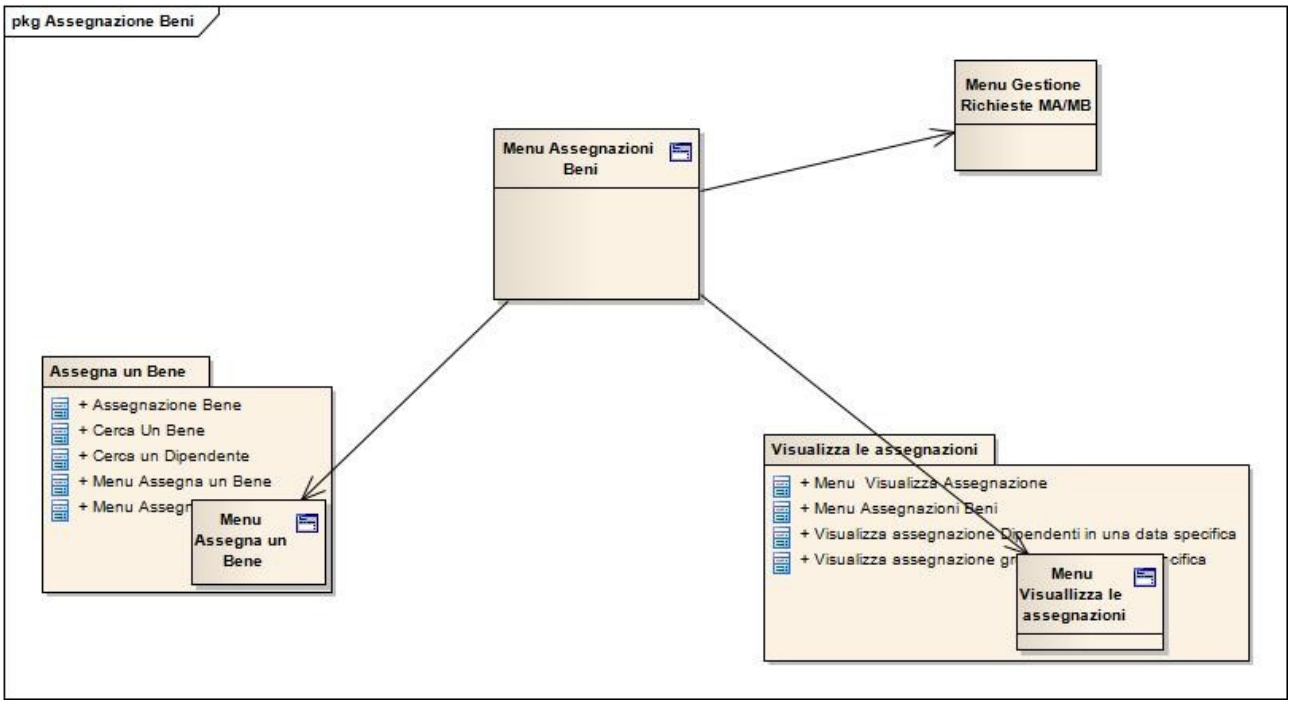

pkg Gestione categoria bene
Elimina categoria beni
Inserisci categoria beni
- nome: char
- sigla: int
Menù categoria beni
Modifica categoria beni
Visualizza categoria bene

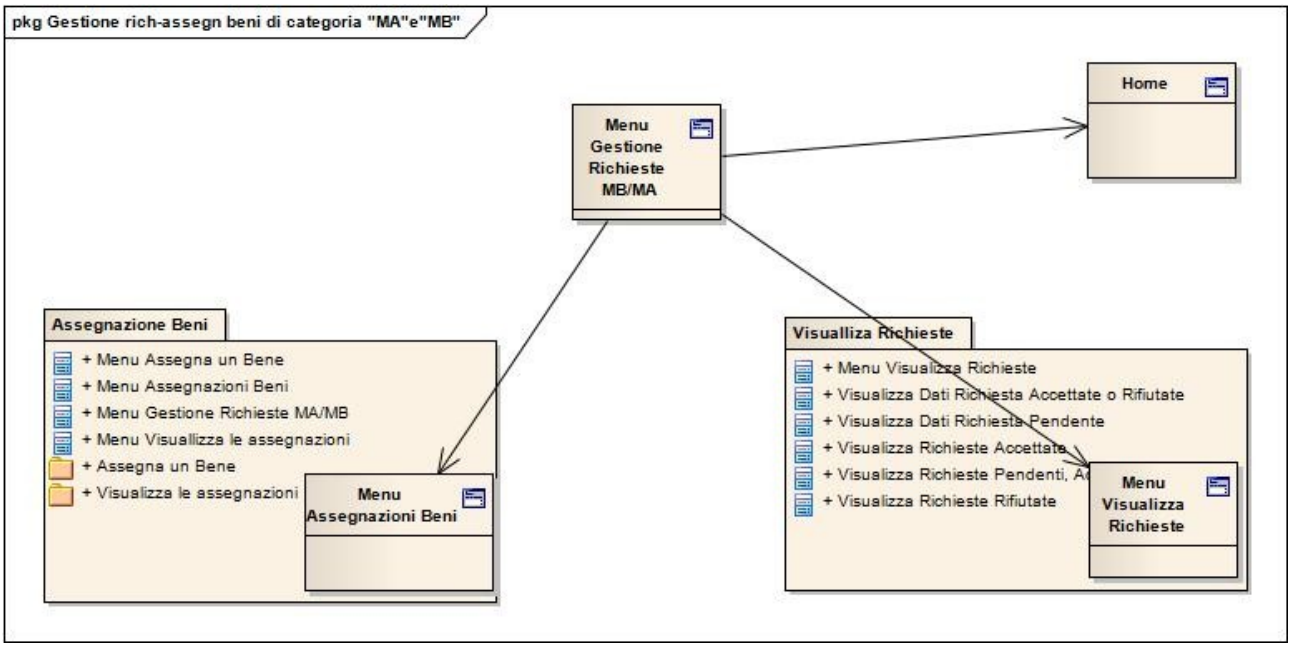

pkg Gestione fornitori
Elimina fornitori
Inserisci fornitori
- email: char
- importo: int
- indirizzo: char
- nome organizzazione: char
- partita_iva: int
- telefono: int
- tipologia: char
Menù fornitori
Modifica fornitori
Visualizza fornitori
Ricerca Fornitori
- E-mail: char
- Indirizzo: char
- Nome: char
- PartitaIva: int
- Telefono: int
- Tipologia: char

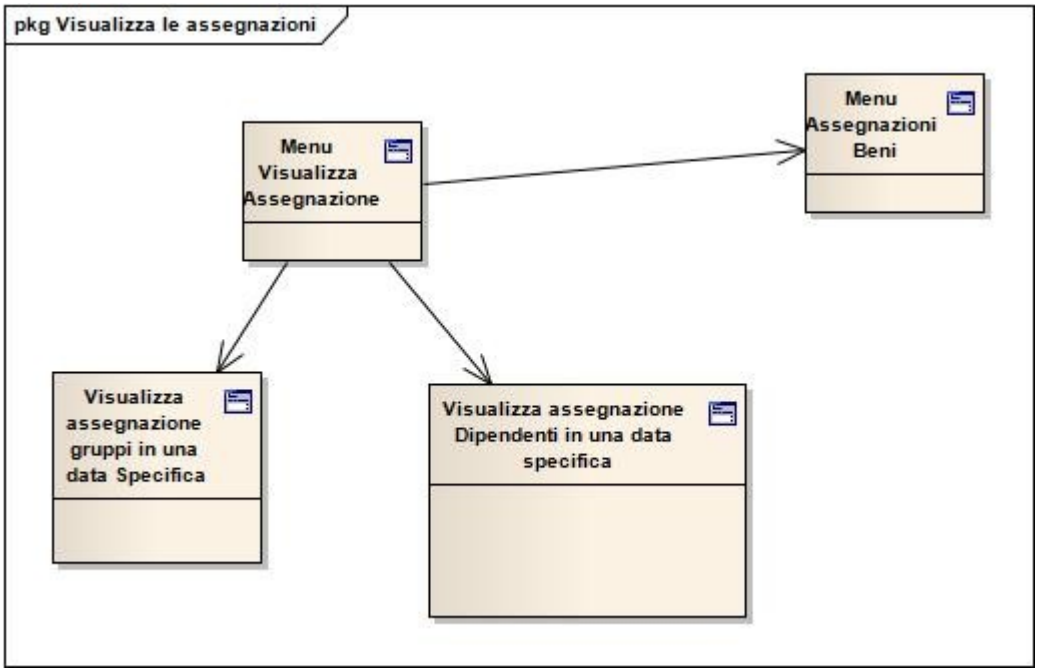

pkg Gestione sottoCategoria Bene
Elimina sottocategoria beni
Inserisci sottocategoria beni
- codice: int
- nome: char
Menù sottocategoria beni
Modifica sottocategoria beni
Visualizza sottocategoria beni
Ricerca Sottocategoria
- CodiceCategoria: char
- codiceSottocategoria: char
- Nome Sottocategoria: char

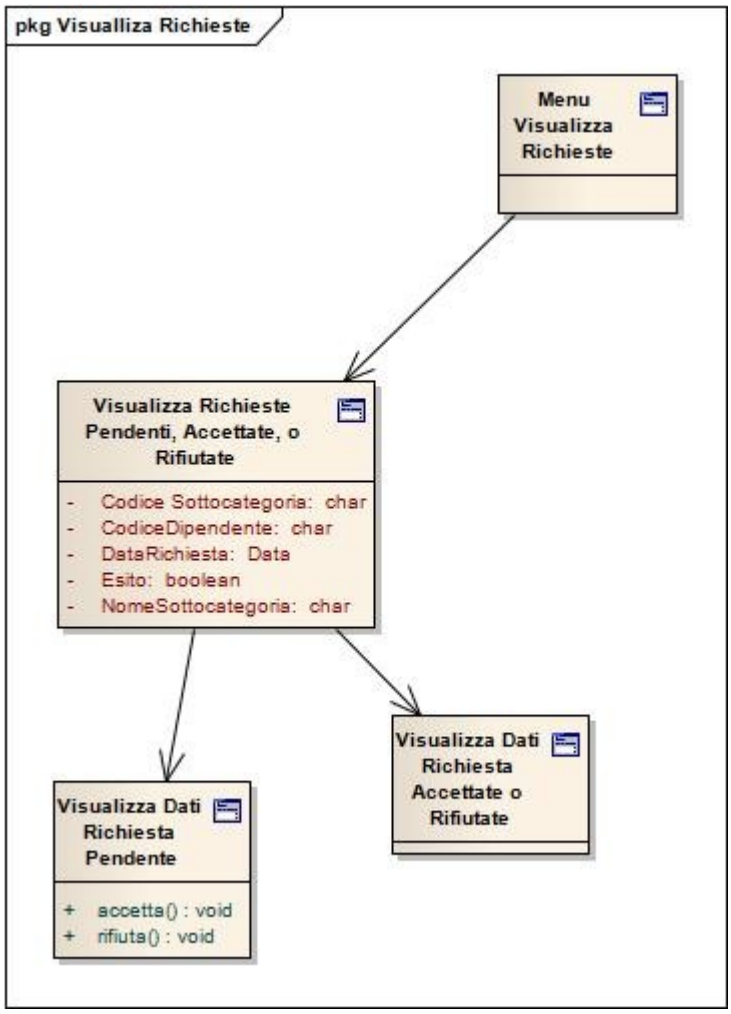

pkg Gestione Ubicazione bene
Elimina ubicazione beni
Inserisci ubicazione beni
- codice: int
- data_fine: char
- data_inizio: char
Menù ubicazione beni
Modifica ubicazione beni
Visualizza ubicazione beni
Ricerca Ubicazione
- CodiceBeni: char
- CodiceUbicazione: char
- NomeStanz: char

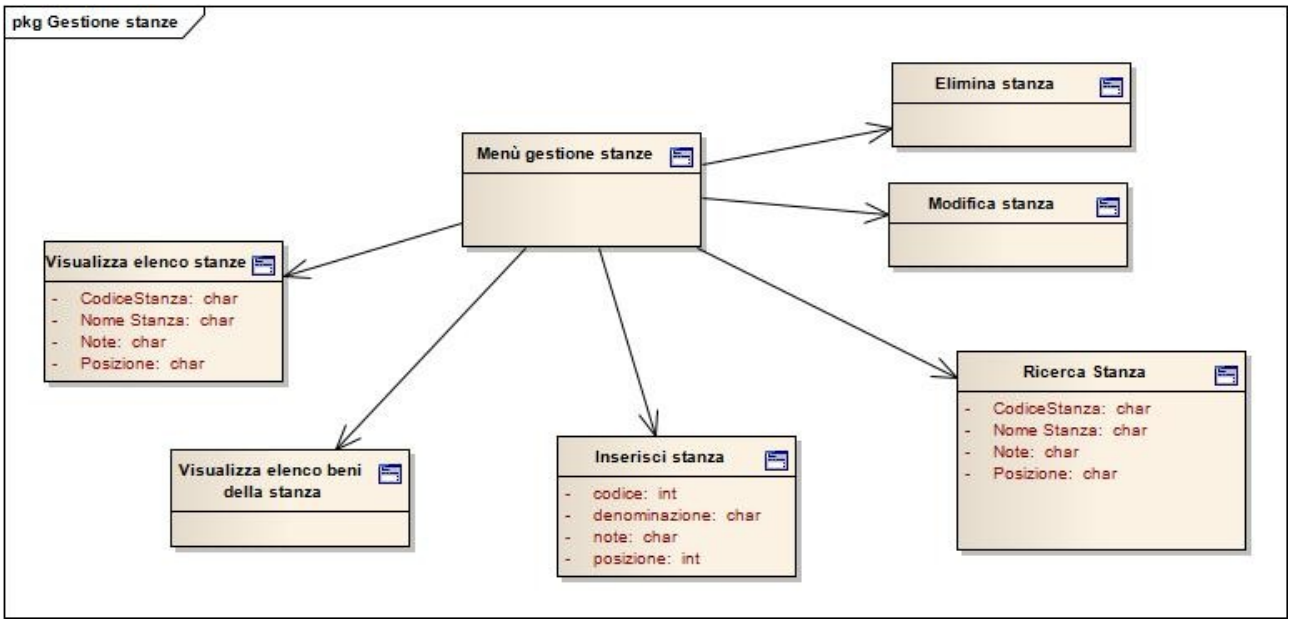

pkg Assegna un Bene
Menu Assegna un Bene
Menu Assegnazione Bene
Assegnazione Bene
- DataFine: Data
- DataInizio: Data
- Note: char
- TipoAssegnazione: char
+ Assegna() : void
Cerca Un Bene
- Nome SottoCategoria: char
- Targhetta: char
Cerca un Dipendente
- Cognome: char
- Nome: char

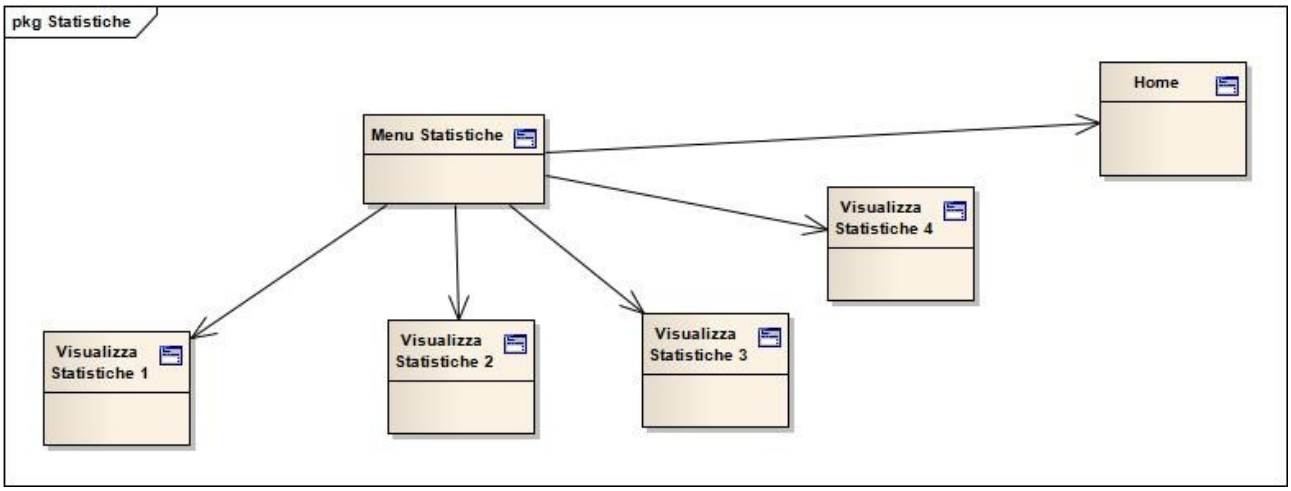

pkg Assegnazione Beni
Menu Gestione Richieste MA/MB
Menu Assegnazioni Beni
Assegna un Bene
+ Assegnazione Bene
+ Cerca Un Bene
+ Cerca un Dipendente
+ Menu Assegna un Bene
Menu Assegna un Bene
Visualizza le assegnazioni
+ Menu Visualizza Assegnazione
+ Menu Assegnazioni Beni
+ Visualizza assegnazione Dipendenti in una data specifica
Menu Visuallizza le assegnazioni


pkg Gestione rich-assegn beni di categoria "MA"e"MB"
Home
Menu Gestione Richieste MB/MA
Assegnazione Beni
+ Menu Assegna un Bene
+ Menu Assegnazioni Beni
+ Menu Gestione Richieste MA/MB
+ Menu Visuallizza le assegnazioni
+ Assegna un Bene
+ Visualizza le assegnazioni
Menu Assegnazioni Beni
Visualliza Richieste
+ Menu Visualizza Richieste
+ Visualizza Dati Richiesta Accettate o Rifiutate
+ Visualizza Dati Richiesta Pendente
+ Visualizza Richieste Accettate
+ Visualizza Richieste Pendenti, A
+ Visualizza Richieste Rifiutate
Menu Visualizza Richieste

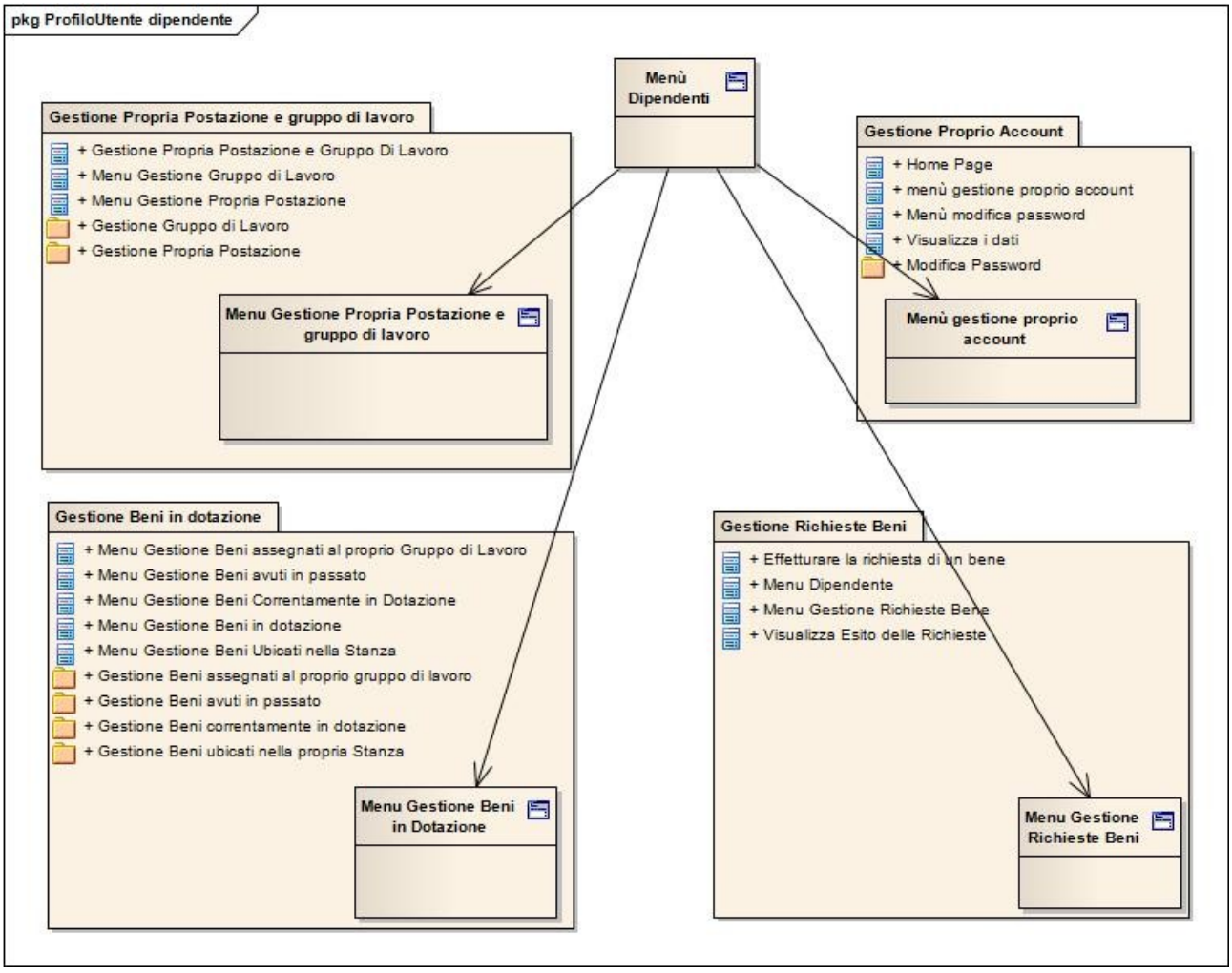

pkg Visualizza le assegnazioni
Menu Visualizza Assegnazione
Menu Assegnazioni Beni
Visualizza assegnazione gruppi in una data Specifica
Visualizza assegnazione Dipendenti in una data specifica

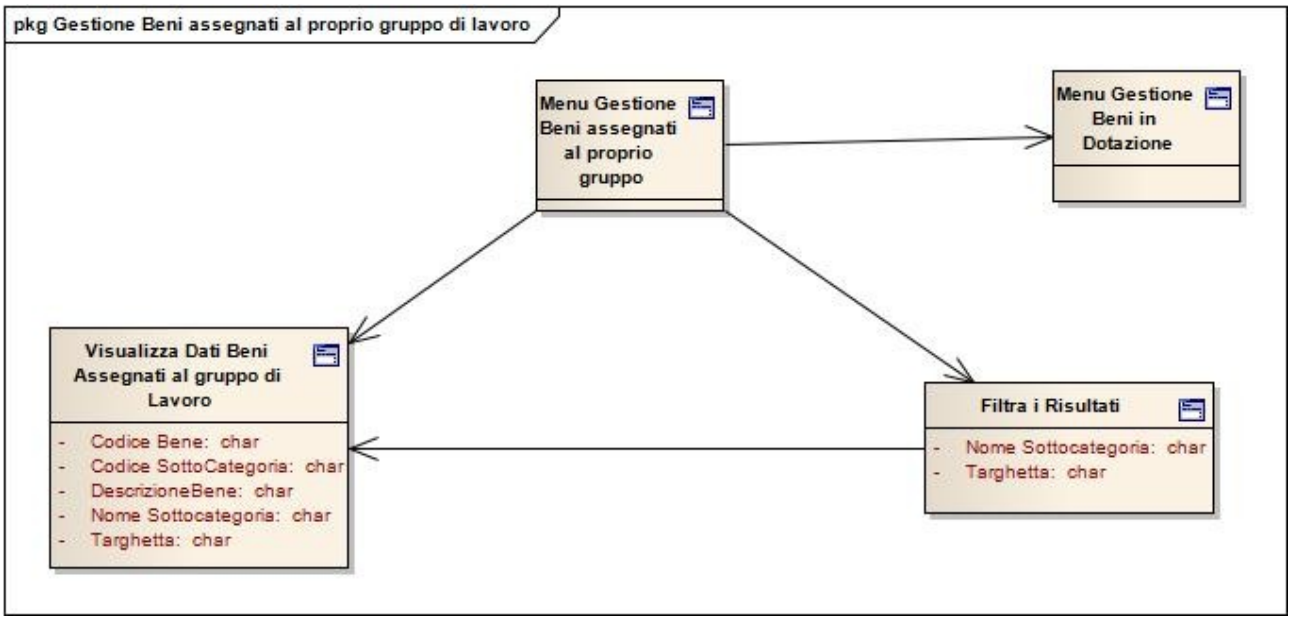

pkg Visualliza Richieste
Menu Visualizza Richieste
Visualizza Richieste Pendenti, Accettate, o Rifiutate
- Codice Sottocategoria: char
- CodiceDipendente: char
- DataRichiesta: Data
- Esito: boolean
- NomeSottocategoria: char
Visualizza Dati Richiesta Accettate o Rifiutate
Visualizza Dati Richiesta Pendente
+ accetta() : void
+ rifiuta() : void

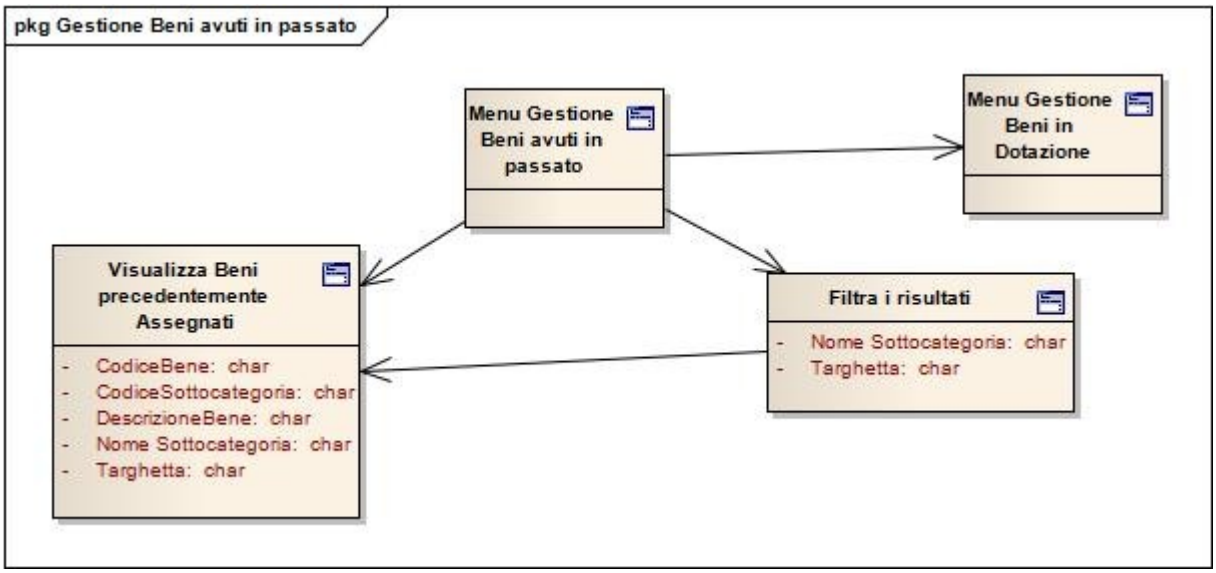

pkg Gestione stanze
Elimina stanza
Menù gestione stanze
Modifica stanza
Visualizza elenco stanze
- CodiceStanza: char
- Nome Stanza: char
- Note: char
- Posizione: char
Ricerca Stanza
- CodiceStanza: char
- Nome Stanza: char
- Note: char
- Posizione: char
Visualizza elenco beni della stanza
Inserisci stanza
- codice: int
- denominazione: char
- note: char
- posizione: int

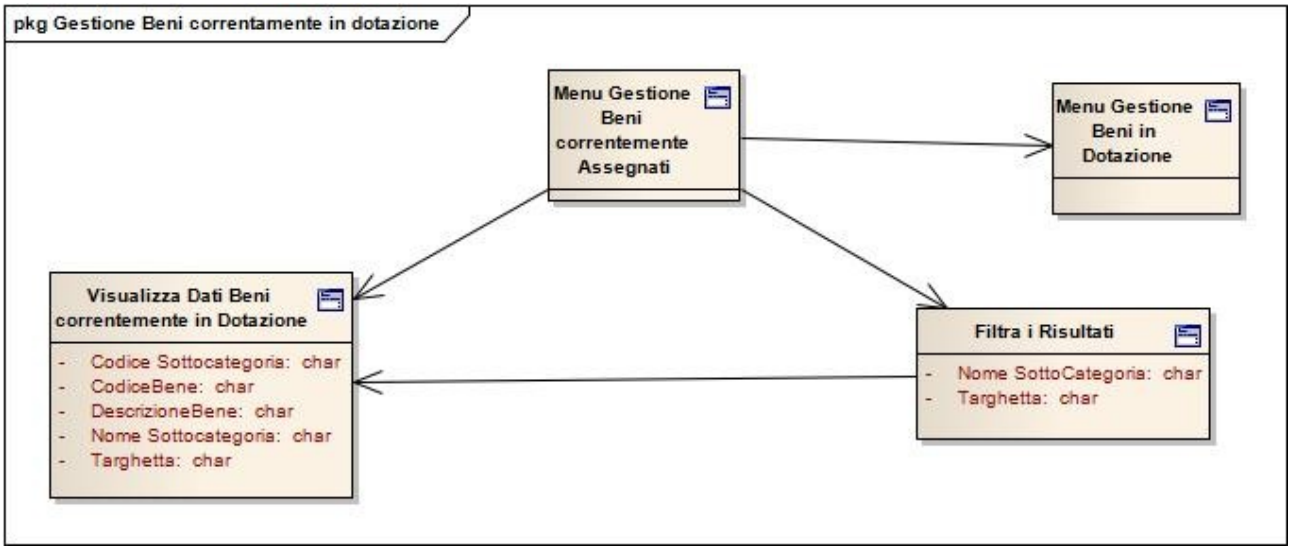


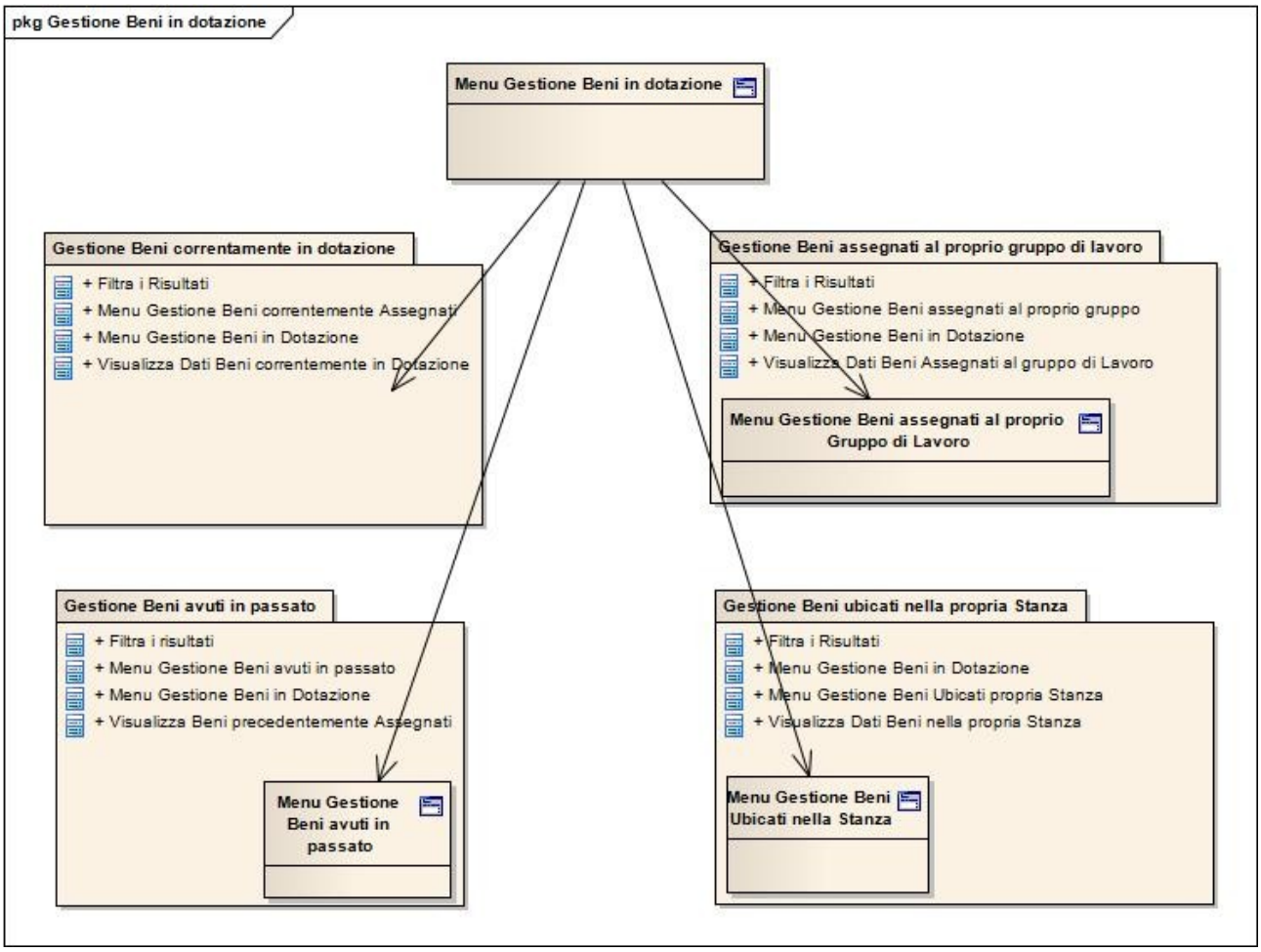
Casi d'uso Dipendente

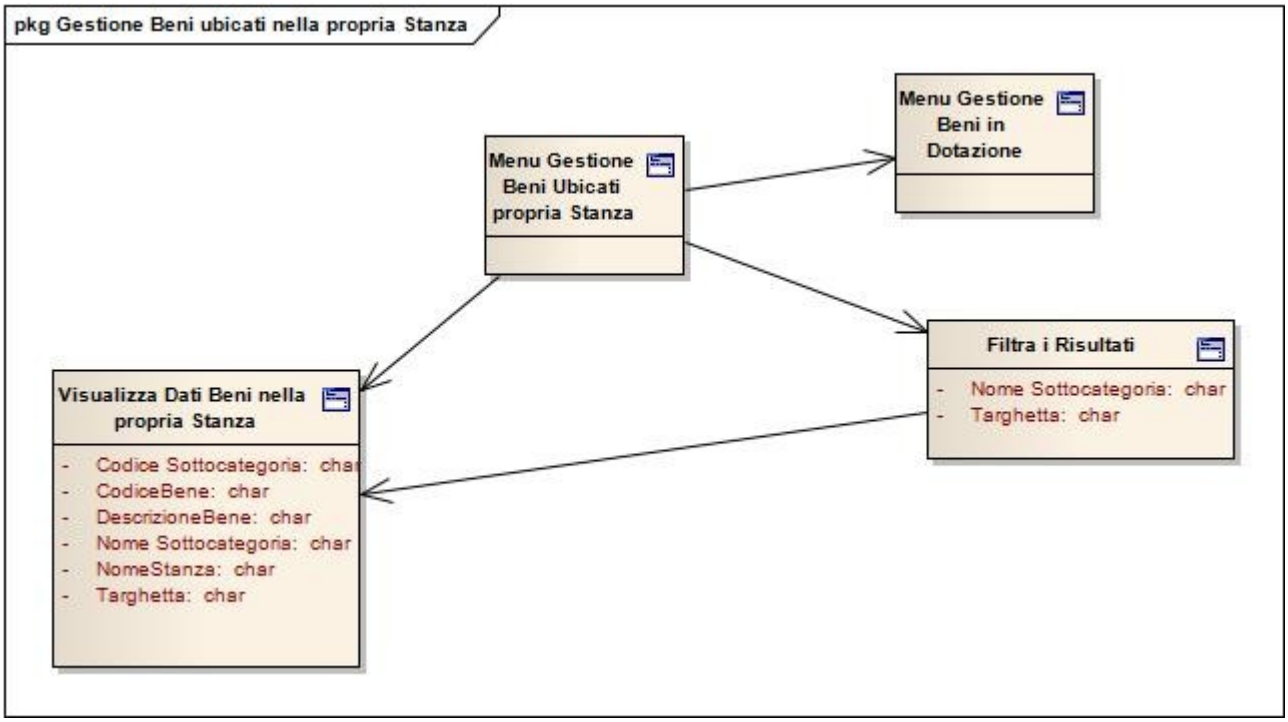


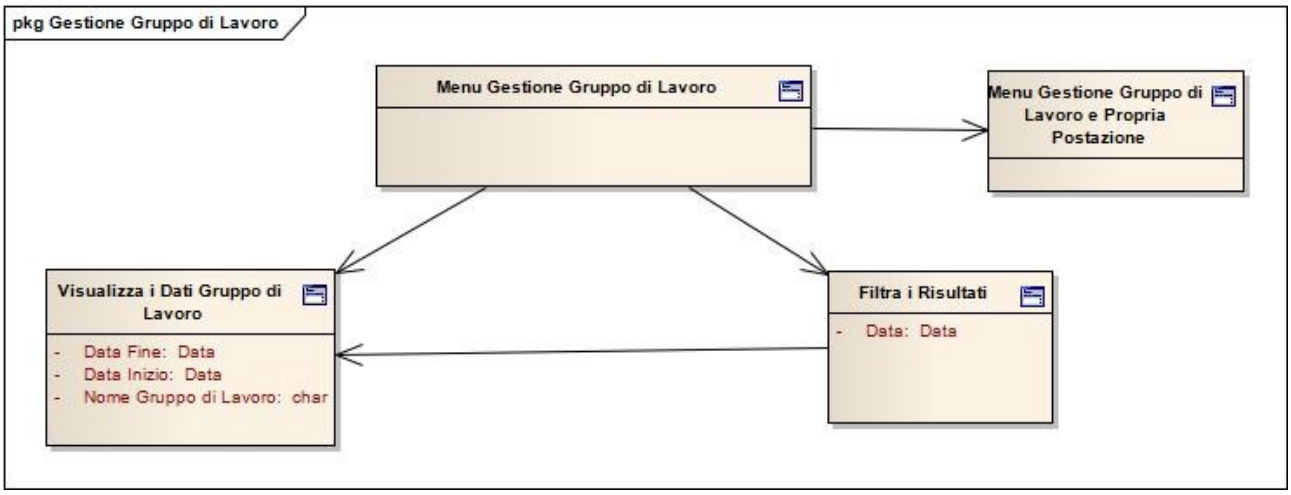

pkg Gestione Beni assegnati al proprio gruppo di lavoro
Menu Gestione Beni assegnati al proprio gruppo
Menu Gestione Beni in Dotazione
Visualizza Dati Beni Assegnati al gruppo di Lavoro
- Codice Bene: char
- Codice SottoCategoria: char
- DescrizioneBene: char
- Nome Sottocategoria: char
- Targhetta: char
Filtra i Risultati
- Nome Sottocategoria: char
- Targhetta: char

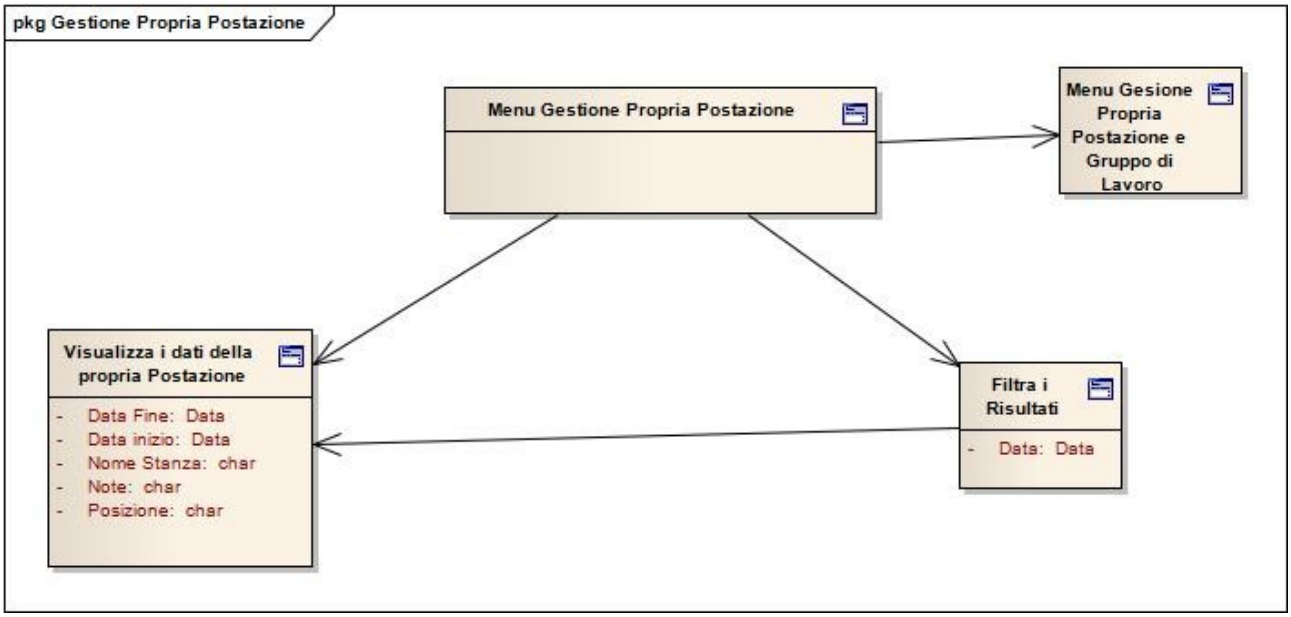

pkg Gestione Beni avuti in passato
Menu Gestione Beni avuti in passato
Menu Gestione Beni in Dotazione
Visualizza Beni precedentemente Assegnati
- CodiceBene: char
- CodiceSottocategoria: char
- DescrizioneBene: char
- Nome Sottocategoria: char
- Targhetta: char
Filtra i risultati
- Nome Sottocategoria: char
- Targhetta: char

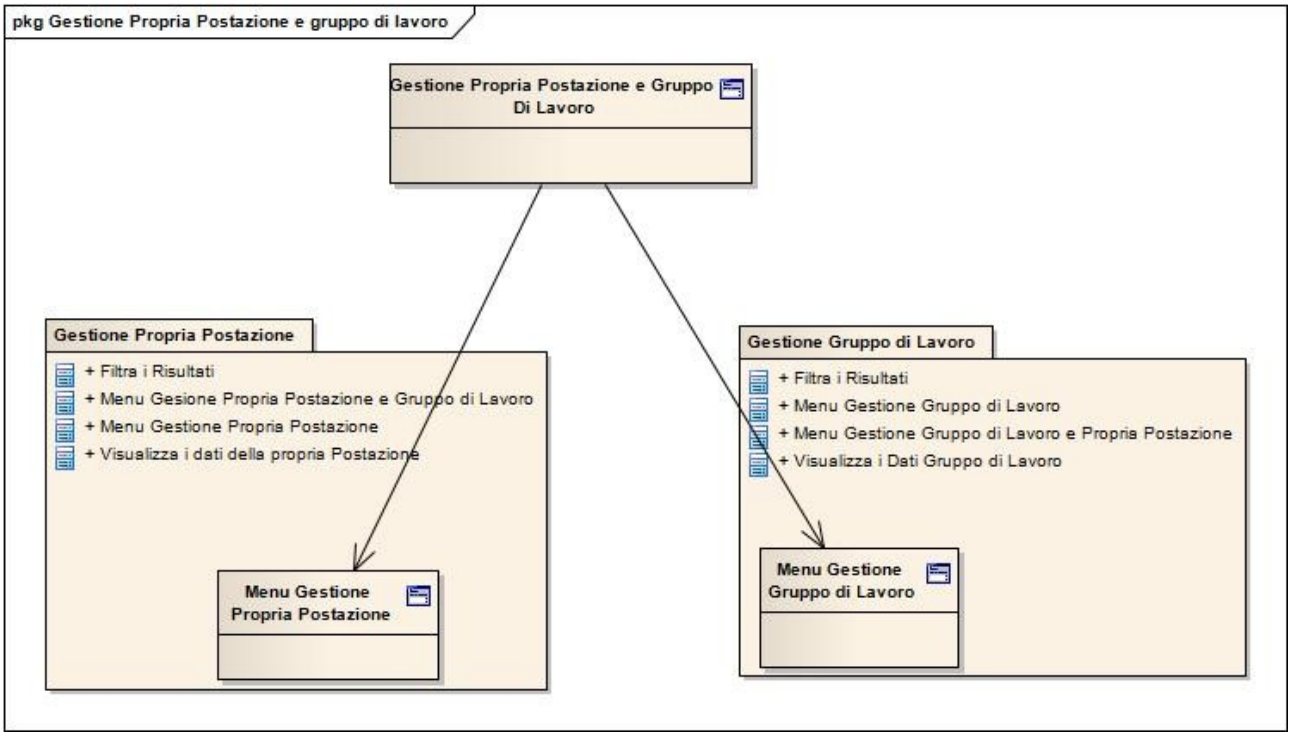

pkg Gestione Beni correntamente in dotazione
Menu Gestione Beni correntemente Assegnati
Menu Gestione Beni in Dotazione
Visualizza Dati Beni correntemente in Dotazione
- Codice Sottocategoria: char
- CodiceBene: char
- DescrizioneBene: char
- Nome Sottocategoria: char
- Targhetta: char
Filtra i Risultati
- Nome SottoCategoria: char
- Targhetta: char

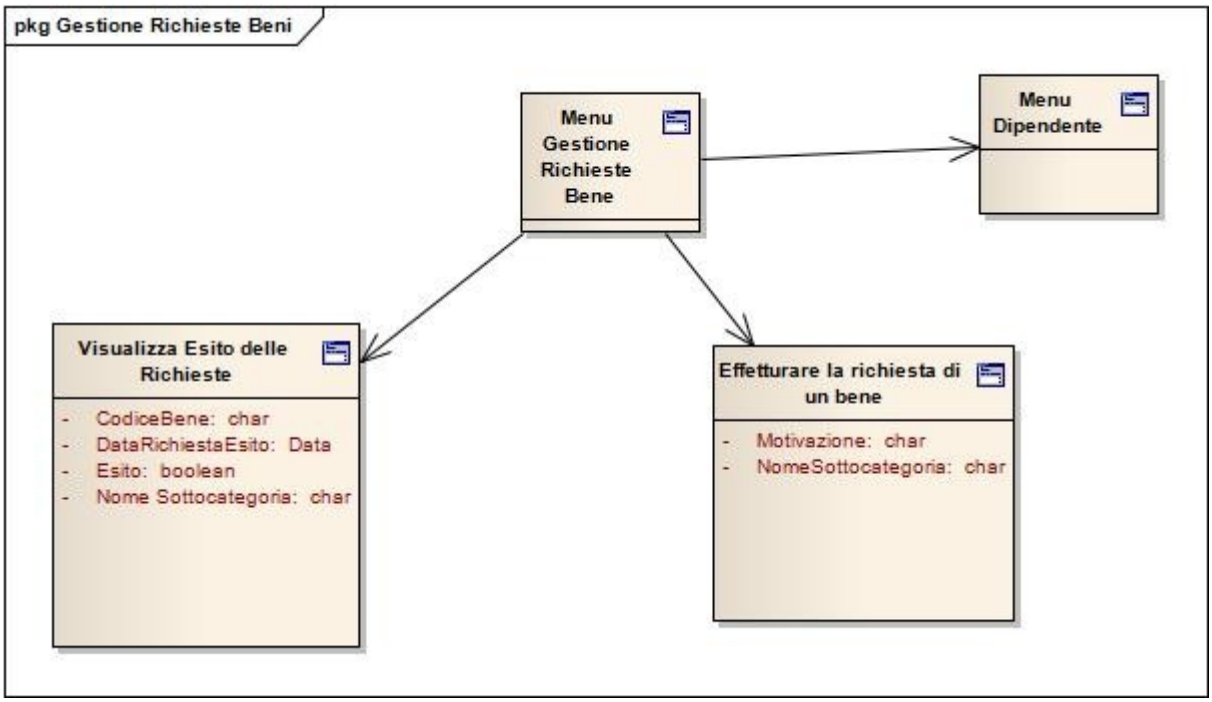

pkg Gestione Beni in dotazione
Menu Gestione Beni in dotazione
Gestione Beni correntamente in dotazione
+ Filtra i Risultati
+ Menu Gestione Beni correntemente Assegnati
+ Menu Gestione Beni in Dotazione
+ Visualizza Dati Beni correntemente in Dotazione
Gestione Beni assegnati al proprio gruppo di lavoro
+ Filtra i Risultati
+ Menu Gestione Beni assegnati al proprio gruppo
+ Menu Gestione Beni in Dotazione
+ Visualizza Dati Beni Assegnati al gruppo di Lavoro
Menu Gestione Beni assegnati al proprio Gruppo di Lavoro
Gestione Beni avuti in passato
+ Filtra i risultati
+ Menu Gestione Beni avuti in passato
+ Menu Gestione Beni in Dotazione
+ Visualizza Beni precedentemente Assegnati
Menu Gestione Beni avuti in passato
Gestione Beni ubicati nella propria Stanza
+ Filtra i Risultati
+ Menu Gestione Beni in Dotazione
+ Menu Gestione Beni Ubicati propria Stanza
+ Visualizza Dati Beni nella propria Stanza
Menu Gestione Beni Ubicati nella Stanza


pkg Gestione Beni ubicati nella propria Stanza
Menu Gestione Beni in Dotazione
Menu Gestione Beni Ubicati propria Stanza
Filtra i Risultati
- Nome Sottocategoria: char
- Targhetta: char
Visualizza Dati Beni nella propria Stanza
- Codice Sottocategoria: char
- CodiceBene: char
- DescrizioneBene: char
- Nome Sottocategoria: char
- NomeStanza: char
- Targhetta: char


pkg Gestione Gruppo di Lavoro
Menu Gestione Gruppo di Lavoro
Menu Gestione Gruppo di Lavoro e Propria Postazione
Visualizza i Dati Gruppo di Lavoro
- Data Fine: Data
- Data Inizio: Data
- Nome Gruppo di Lavoro: char
Filtra i Risultati
- Data: Data


pkg Gestione Propria Postazione
Menu Gestione Propria Postazione
Menu Gesione Propria Postazione e Gruppo di Lavoro
Visualizza i dati della propria Postazione
- Data Fine: Data
- Data inizio: Data
- Nome Stanza: char
- Note: char
- Posizione: char
Filtra i Risultati
- Data: Data


pkg Gestione Propria Postazione e gruppo di lavoro
Gestione Propria Postazione e Gruppo Di Lavoro
Gestione Propria Postazione
+ Filtra i Risultati
+ Menu Gesione Propria Postazione e Gruppo di Lavoro
+ Menu Gestione Propria Postazione
+ Visualizza i dati della propria Postazione
Menu Gestione Propria Postazione
Gestione Gruppo di Lavoro
+ Filtra i Risultati
+ Menu Gestione Gruppo di Lavoro
+ Menu Gestione Gruppo di Lavoro e Propria Postazione
+ Visualizza i Dati Gruppo di Lavoro
Menu Gestione Gruppo di Lavoro


pkg Gestione Richieste Beni
Menu Gestione Richieste Bene
Menu Dipendente
Visualizza Esito delle Richieste
- CodiceBene: char
- DataRichiestaEsito: Data
- Esito: boolean
- Nome Sottocategoria: char
Effetturare la richiesta di un bene
- Motivazione: char
- NomeSottocategoria: char