UNIVERSITÀ
DEGLI STUDI
DI PALERMO

TESINA
Ingegneria del Software
Docente Valeria Seidita

SDD
(SYSTEM DESIGN DOCUMENT)

Luca Al Daire
Fabio Palmese
Emanuele Di Miceli
Gabriele Tornatore

Ingegneria Informatica
Anno Accademico 2017/18

Indice

INTRODUZIONE 2
Obiettivi generali 2
Acronimi e definizioni 2
ARCHITETTURA SOFTWARE CORRENTE 3
ARCHITETTURA SOFTWARE PROPOSTA 3
Hardware-Software Mapping 3
Decomposizione del sistema 4
Package del sistema 5
Package Server 6
Package Utente 6
Package Dipendente 7
Package Deposito 8
Package Logistica 8
Package Amministrativo 8
Package Accesso 8
Package Connection 8
DATABASE 9
Diagramma Master-Slave 10
Diagramma Entità-Relazione 11
Diagramma ERD 12
Dizionari 13
Informazioni aggiuntive 19

INTRODUZIONE

Obiettivi generali

Il progetto si propone come un software di gestione delle funzionalità relative ad un'azienda di autotrasporti, viene quindi messa a disposizione del dipendente dell'azienda una suite di strumenti necessari alla gestione amministrativa, logistica e organizzativa (gestione dipendenti, mezzi, linee e fermate) sotto opportuna autenticazione. Inoltre il software gestisce anche l'interazione coi clienti dell'azienda (ovvero i cittadini a cui interessa usufruire dei servizi offerti), fornendo le funzionalità che permettono di determinare i percorsi delle linee e i dettagli, le relative fermate e percorsi individuati da posizione iniziale e finale inserite a piacere dal cliente. A supporto di quest'ultima si utilizza una mappa che rende immediata la fruizione delle informazioni ricevute.

È importante sottolineare che una funzione principe del sistema è quella di garantire la genuinità dei dati con cui i dipendenti lavorano, tuttavia rimane a loro il compito di inserire dei dati validi per una elaborazione efficace. Inoltre si vogliono consegnare al committente due applicativi principali, uno da distribuire al pubblico, utile al reperimento delle informazioni su linee, fermate e percorsi, e uno da distribuire internamente all'azienda, utile per facilitare i compiti svolti finora manualmente dagli impiegati.

Acronimi e definizioni

Acronimi:

DBMS: Database Management System
ODD: Object Design Document
RAD: Requirements Analysis Document
SDD: System Design Document
UML: Unified Modelling Language

Definizioni:

Dipendente: utente che utilizza la parte gestionale e organizzativa del software, può accedere a funzionalità diverse a seconda del ruolo, si differenzia in:

- Amministrativo
- Logistica
- Deposito
- Autista

Cliente: utente che utilizza la parte non gestionale del software, ricavando informazioni su linee, fermate e percorsi (coincide con il cittadino che ha deciso di usufruire dei servizi dell'azienda di trasporti).

Linea: percorso cittadino composto da fermate e coperto dagli automezzi dell'azienda di trasporti.

Fermata: posizione della città raggiunta dai mezzi dell'azienda in cui si effettua lo scarico e il carico dei passeggeri.

Percorso: indicazione riguardante le linee da prendere e le fermate di passaggio di queste ultime, per raggiungere una determinata posizione da un punto di partenza prestabilito.

Cedolino: documento contenente i dettagli sulla retribuzione mensile del dipendente.

Cartellino: consente di determinare l'ingresso e l'uscita dal luogo di lavoro.

Login: operazione che permette l'accesso alle funzionalità del sistema previa identificazione del dipendente attraverso le credenziali di accesso.

Logout: operazione complementare a quella di Login, consente al dipendente di abbandonare la schermata delle operazioni consentendogli la deautenticazione.

Form: finestra di dialogo che permette all'utente l'interazione col sistema.

DBMS: software utilizzato per la gestione del DataBase.

DataBase: archivio di dati strutturato.

Nome Utente: alias del dipendente necessario al Login

ARCHITETTURA SOFTWARE CORRENTE

Non essendoci software presistenti non vi sono architetture già proposte per questo tipo di software.

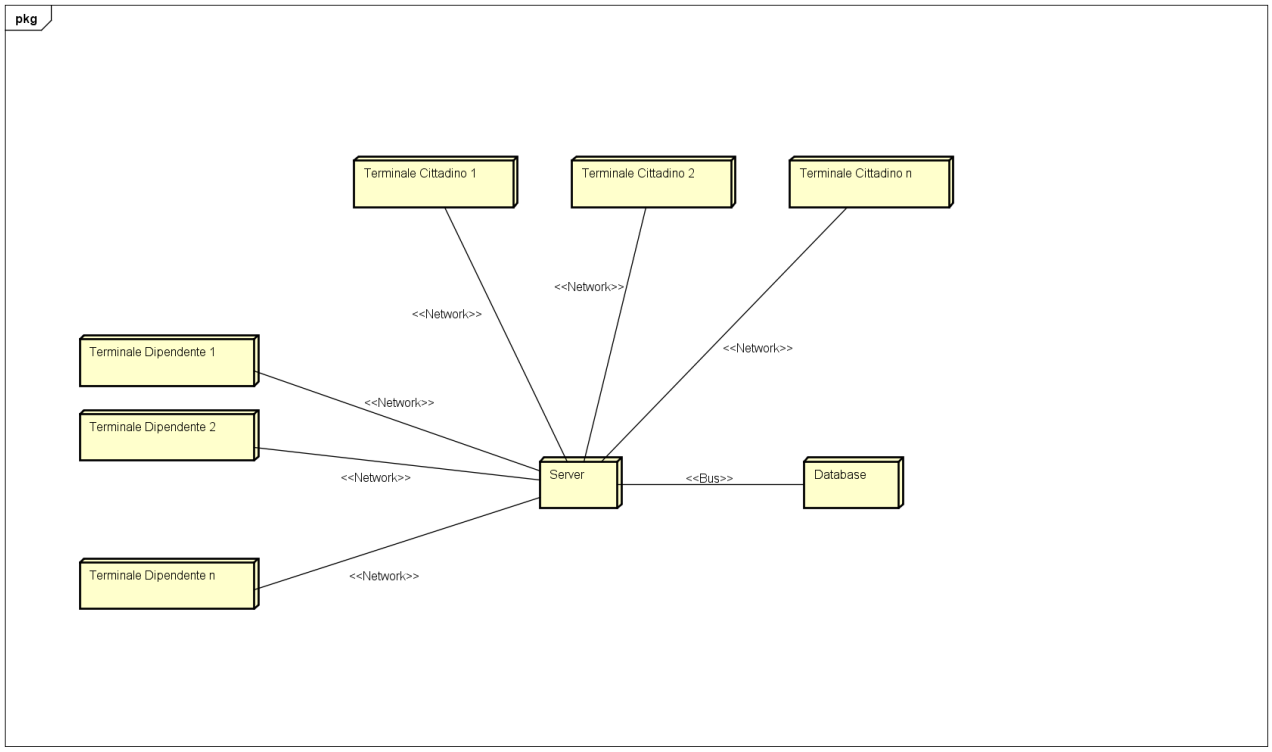
ARCHITETTURA SOFTWARE PROPOSTA

Il sistema che si andrà a sviluppare sfrutta l'architettura Client-Server. Tale sistema è costituito da tre componenti principali, due di tipo client e 1 di tipo server. La prima componente client si occupa di fornire assistenza ai dipendenti della società, la seconda supporta il cittadino durante la ricerca del mezzo pubblico adeguato da poter utilizzare. La componente server si occupa di coordinare il tutto fornendo i servizi necessari e le informazioni recuperate dal Database centrale dell'azienda. Solo il server sotto opportuna autenticazione permette al Client di modificare i dati presenti. Il client dedicato al cittadino ha libero accesso solo alle informazioni riguardante le linee e gli orari. Il linguaggio di programmazione utilizzato è Java, mentre il database è realizzato tramite MySQL. La comunicazione tra il codice ed il database è ottenuta col supporto della classe java JDBC (Java DataBase Connectivity).

Hardware-Software Mapping

Il fulcro del sistema è il server centrale. Tramite lui i vari terminali possono scambiarsi informazioni ed interagire con il Database.

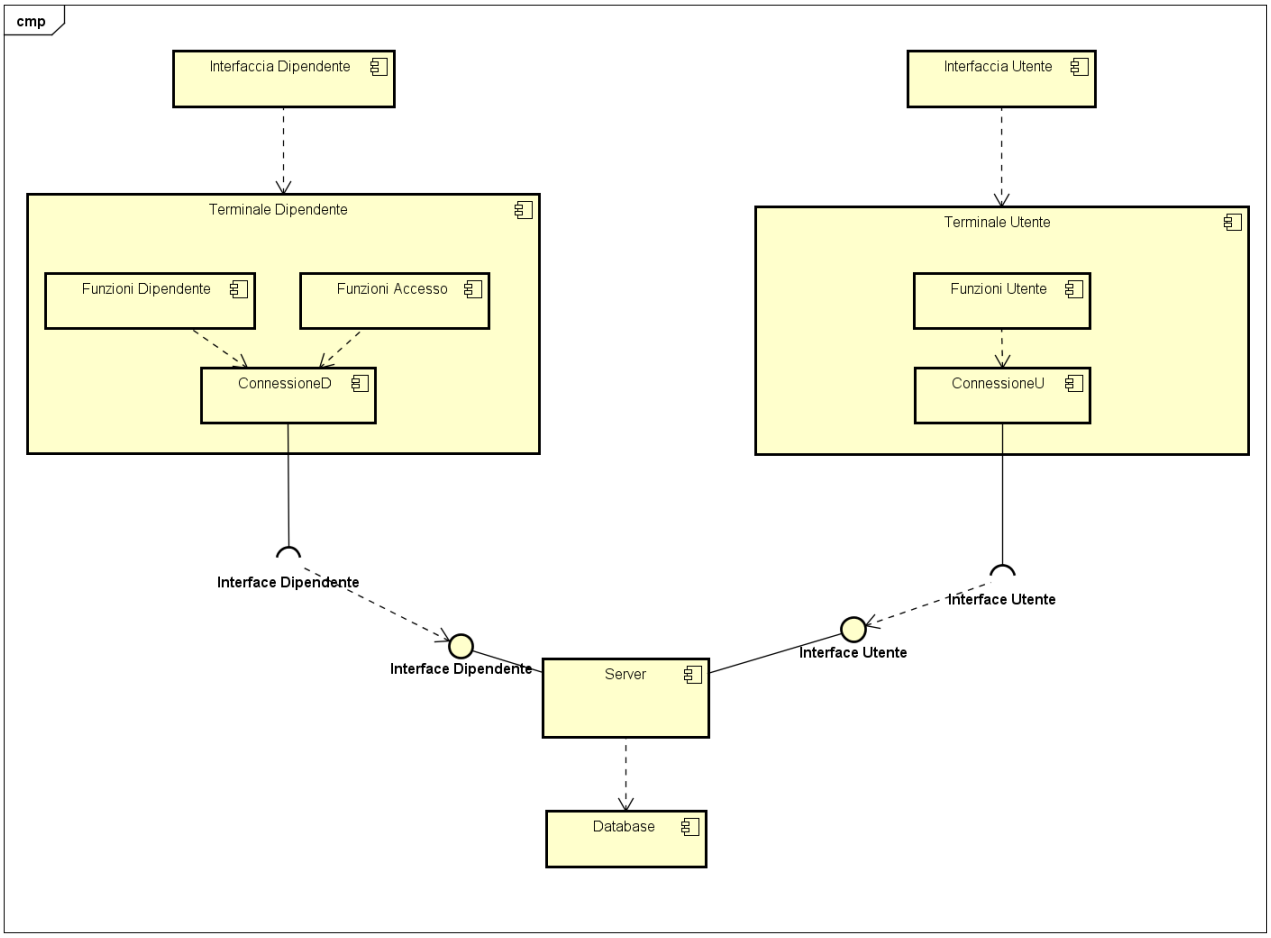
Diagramma di Depoyment del sistema:



Decomposizione del sistema

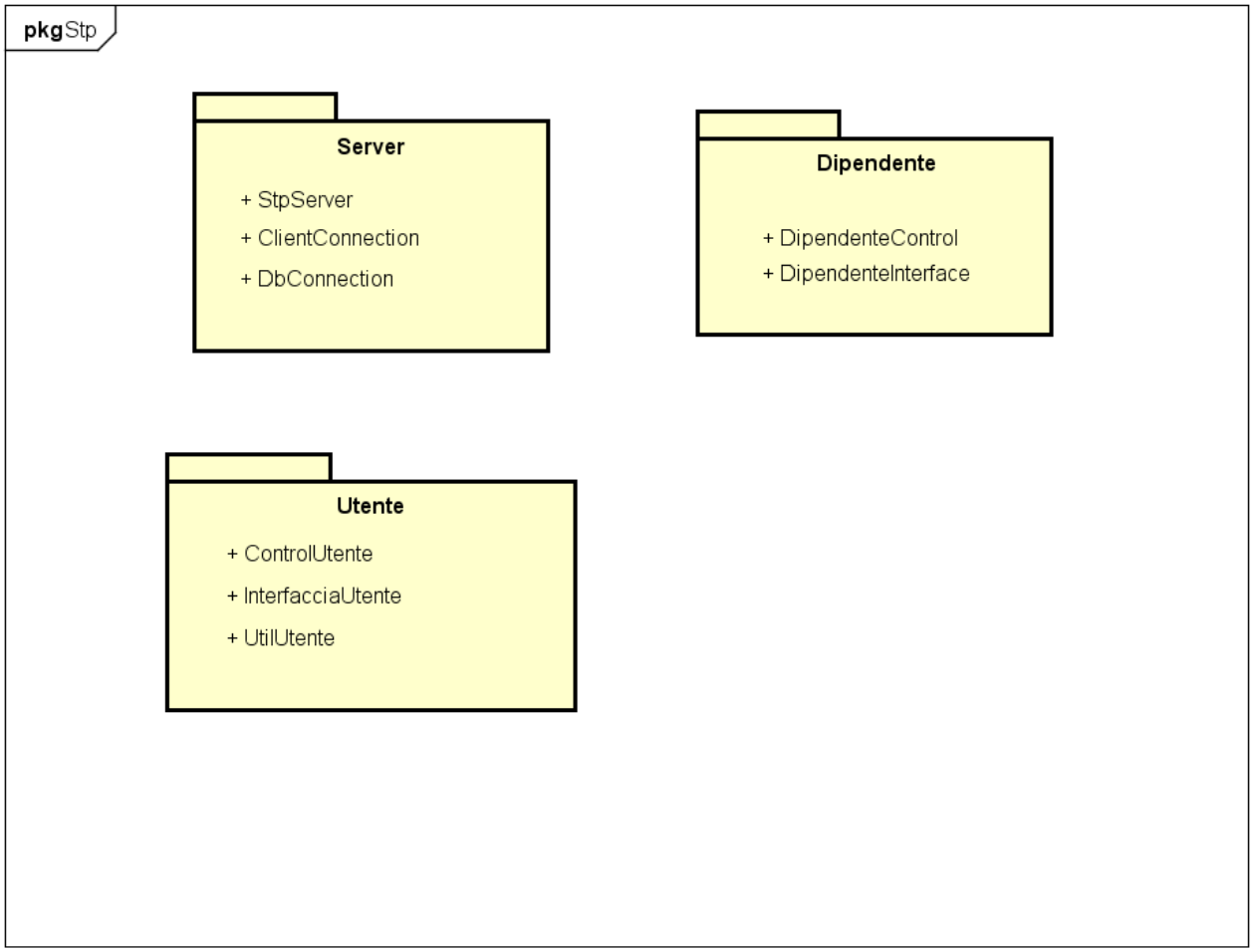
Dall'analisi del diagramma di deployment è possibile ricavare diverse componenti. La parte Server si occupa di gestire tutte le informazioni e fornirle ai relativi Terminali (Sia Dipendente che Utente). Entrambi i terminali sfruttano una connessione TCP/IP per poter comunicare con il Server che fornisce un'interfaccia prestabilita ai vari terminali, in modo tale da limitare la corruzione dei dati presenti nel DB. Infatti solo il server può avere accesso ai dati nel DataBase. Solo tramite opportuna autentificazione da parte del Dipendente è possibile modificare i dati presenti. L'utente può solo visualizzare un insieme limitato di Informazioni.

Di seguito la decomposizione del sistema:



Package del sistema

Il Sistema può essere suddiviso nei suddetti macro package:



Package Server

Il Package del server è composto da:

- StpServer: gestione del server
- ClientConnection: si occupa di rispondere ad un determinato terminale che ha contattato il server
- DbConnecion: fornisce gli strumenti per accedere ai dati nel DB

Package Utente

Il Package dell'utente è composto da:

- ControlUtente: contiene le varie classi che si occupano di gestire le azioni dell'utente (CercaPercorso, CercaLinea, CercaFermata)
- InterfacciaUtente: le interfacce con cui l'utente può interagire
- UtilUtente: contiene le varie classi di supporto per l'utente (Linea, Fermata)

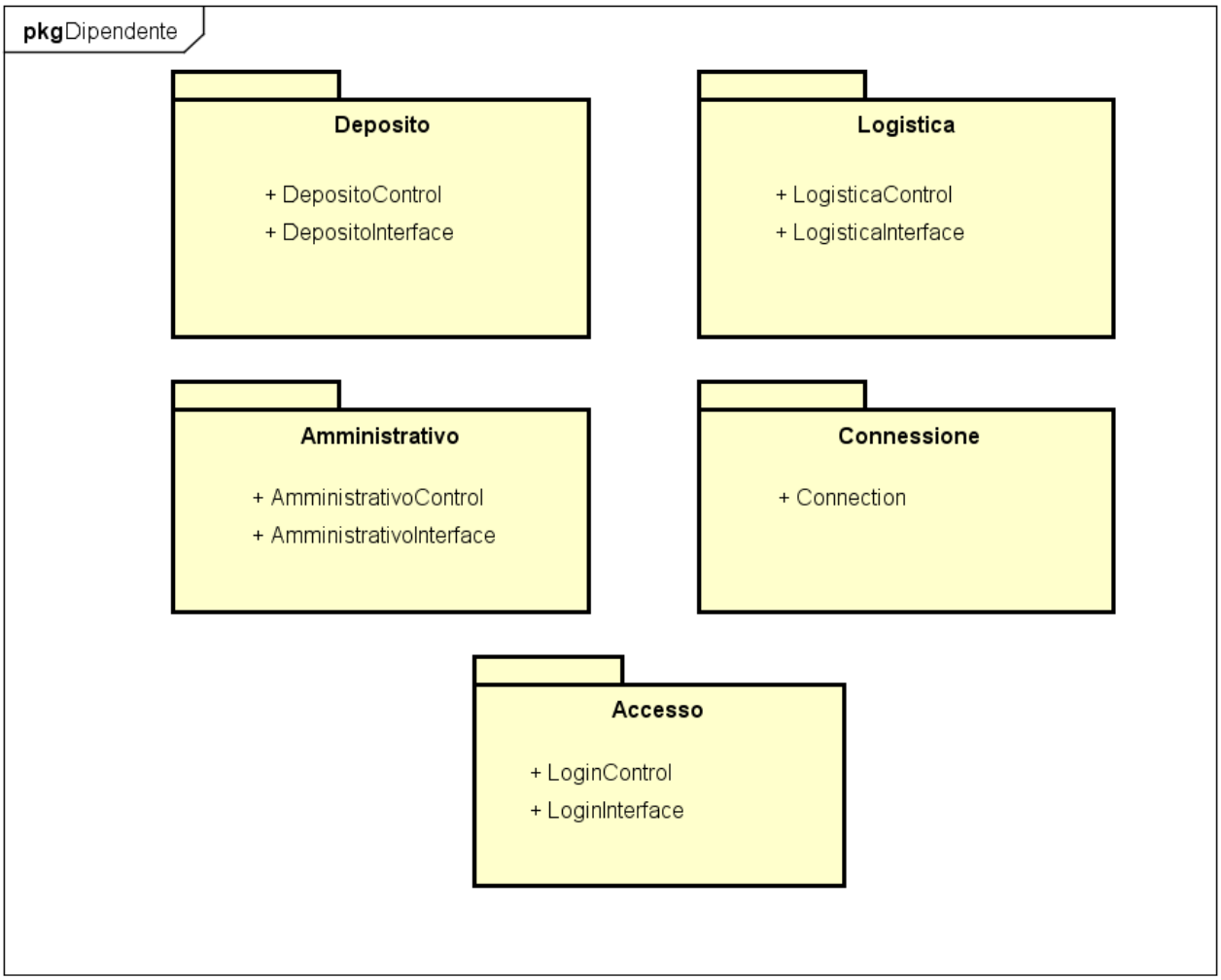
Package Dipendente

Il package del dipendente ha una struttura più complessa dei due sopraindicati poiché possiede funzioni più complesse e ampie.

Esso possiede:

- DipendenteControl: contiene le varie classi che si occupano di gestire le azioni base del dipendente
- DipendenteInterface: contiene le interfacce base che il dipendente ha a disposizione

Di seguito si mostra una vista più ampia del package dipendente:



Package Deposito

Il Package deposito è composto da:

- DepositoControl: contiene le varie classi che si occupano di gestire le azioni dell'addetto al deposito (GestioneAutobus, EntrataUscitaAutobus, GestioneGuasti)
- InterfacciaUtente: le interfacce con cui l'addetto al deposito può interagire

Package Logistica

Il Package logistica è composto da:

- LogisticaControl: contiene le varie classi che si occupano di gestire le azioni dell'addetto alla logistica (GestioneLinee, AssociaAutistaALinea, GestioneFermate)
- LogisticaUtente: le interfacce con cui l'addetto alla logistica può interagire

Package Amministrativo

Il Package amministrativo è composto da:

- AmministrativoControl: contiene le varie classi che si occupano di gestire le azioni dell'Amministrativo (GestionePersonale, GestioneTurni, CompilaCedolino)
- AmministrativoUtente: le interfacce con cui l'amministrativo può interagire

Package Accesso

Il Package amministrativo permette l'autentificazione del dipendente ed è composto da:

- LoginControl: contiene i vari controlli di accesso
- LoginInterface: le interfacce con cui il dipendente può interagire per effettuare il login

Package Connection

Il Package amministrativo contiene un'unica classe che fornisce le varie funzioni per poter connettersi con il server:

- Connection

DATABASE

Descrizione verbale del database:

Il database che di seguito proponiamo attraverso uno schema ERD ed i relativi dizionari che specificano i dettagli di ogni tabella, consente una corretta e appropriata gestione dei dati di interesse.
Il cardine del database è certamente il dipendente, è possibile memorizzare tutti i dettagli necessari alla sua descrizione plenaria, poiché nel nostro sistema si sono individuati quattro ruoli e di conseguenza altrettante specializzazioni per il dipendente generico, è stato necessario costruire anche una tabella "Ruolo" che descrivesse le differenze a livello di modalità lavorative di ciascun dipendente.
Altre due funzionalità fondamentali per un dipendente sono la possibilità di effettuare il timbro del cartellino e di visualizzare i propri turni lavorativi, è per questo che nel nostro database sono presenti le tabelle "Cartellino" e "Turno" che permettono rispettivamente di registrare l'ingresso e l'uscita nella giornata lavorativa e di memorizzare i turni di lavoro. E' importante sottolineare che i dipendenti di tipo autista sono l'unica categoria di dipendenti ad avere non nullo il campo "RefIDLinea", questo poiché chiaramente solo agli autisti viene assegnata una linea.

Un dipendente possiede certamente un cedolino, nell'omonima tabella è possibile conservare tutti i dettagli di quest'ultimo. E' inoltre possibile tenere traccia di assunzioni o licenziamenti attraverso la tabella "RegistroAssunzioni".

Per quanto riguarda le informazioni relative alla gestione logistica dell'azienda, ovvero l'organizzazione di deposito, fermate, linee e automezzi, il database propone cinque tabelle:

- Linea: contiene tutti i dettagli delle linee.
- Fermata: contiene tutti i dettagli delle fermate.
- LineaAssociaFermata: rappresenta la relazione di passaggio di una linea per una determinata fermata.
- Autobus: contiene tutti i dettagli degli autobus disponibili.
- RegistroDeposito: consente di registrare l'uscita oppure l'entrata di un autobus nel deposito, identificando l'autista associato.

Nota: in rosso indicherò i vincoli multipli ad esempio UNIQUE(RefIdlinea, RefIdFermata).

Diagramma Master-Slave

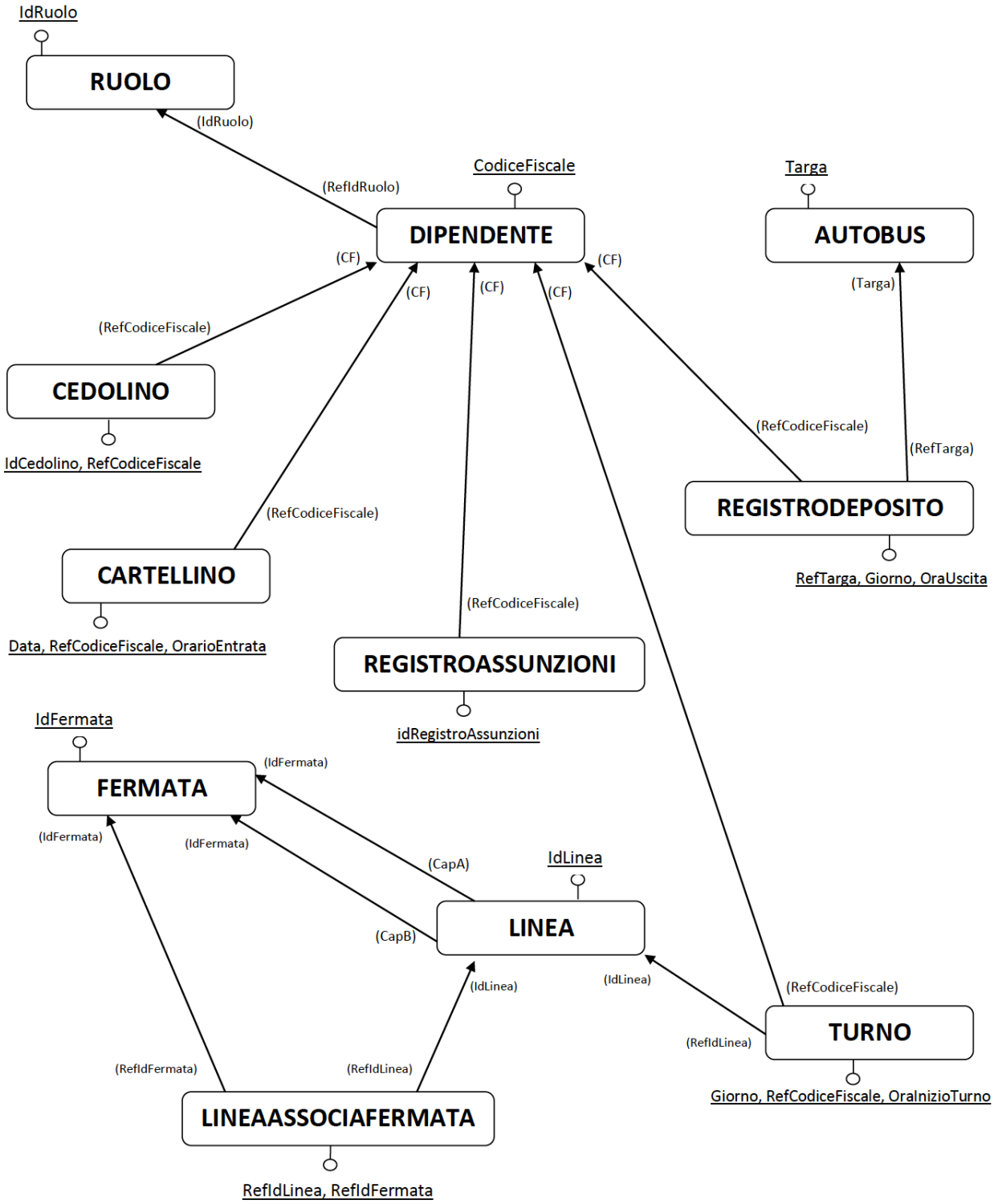


Diagramma Entità-Relazione

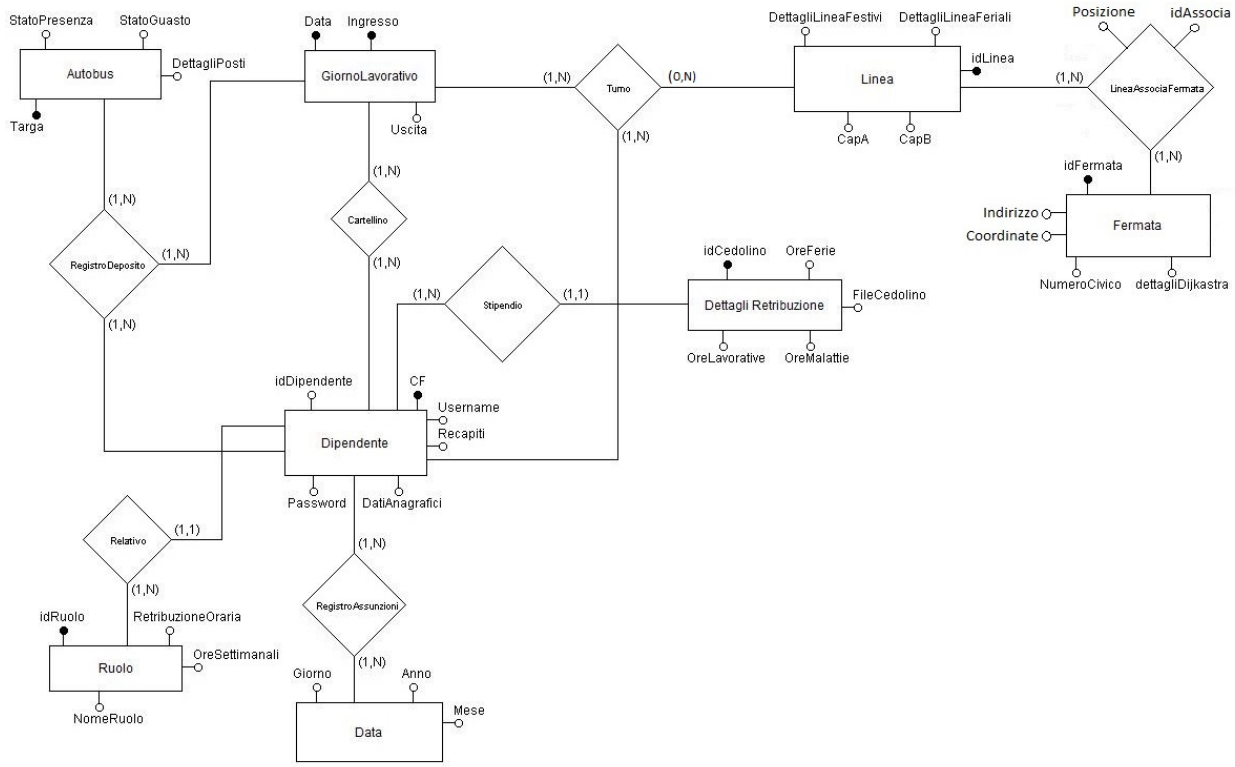
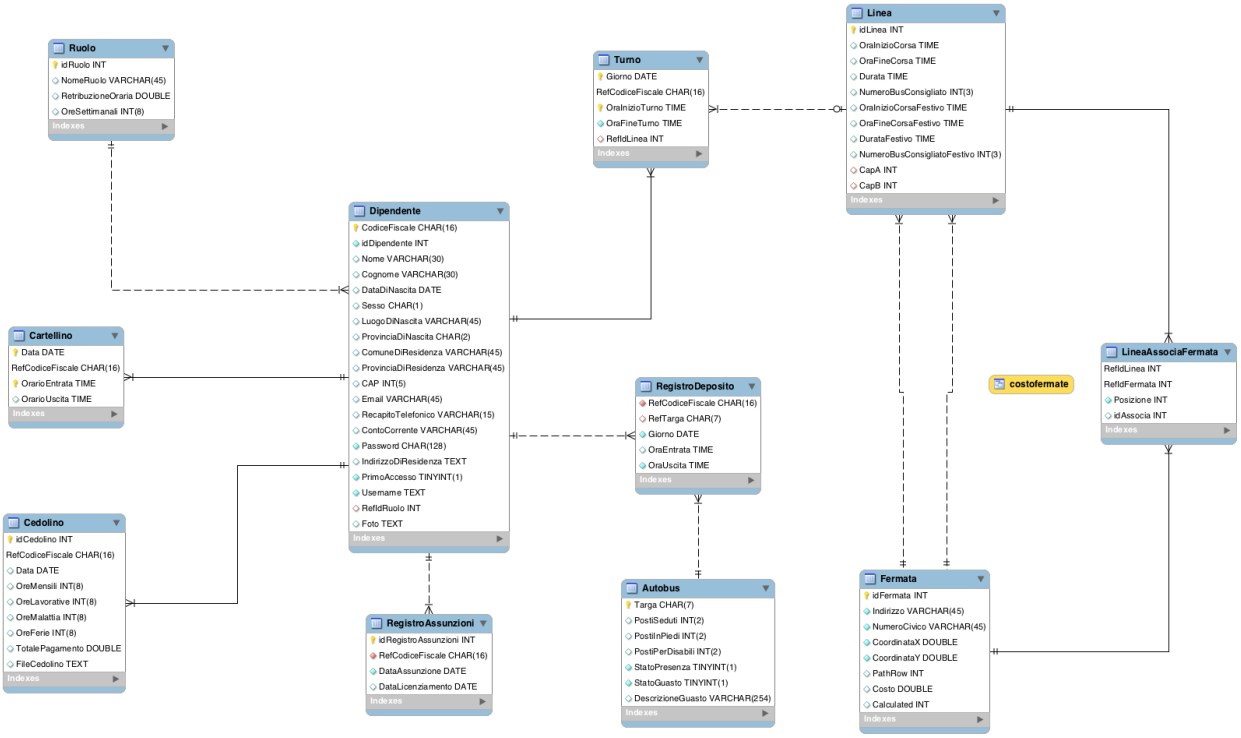


Diagramma ERD



Dizionari

Ruolo

Nome Colonna	Tipo	Vincoli	Descrizione
idRuolo	INT	PK, AUTO_INCREMENT	Individua univocamente un ruolo.
NomeRuolo	VARCHAR(45)		Rappresenta il nome del ruolo.
RetribuzioneOraria	DOUBLE		Specifica la retribuzione oraria relativa al ruolo.
OreSettimanali	INT(8)		Specifica il numero delle ore settimanali lavorative che competono al ruolo.

Cartellino

Nome Colonna	Tipo	Vincoli	Descrizione
Data	DATE	PK	Specifica la data della giornata lavorativa.
RefCodiceFiscale	CHAR(16)	PK, FK	Riferimento al Codice Fiscale del dipendente.
OrarioEntrata	TIME	PK	Specifica l'orario di entrata del dipendente nella giornata lavorativa.
OrarioUscita	TIME		Specifica l'orario di uscita del dipendente nella giornata lavorativa.

Fermata

Nome Colonna	Tipo	Vincoli	Descrizione
idFermata	INT	PK, AUTO_INCREMENT	Individua univocamente una fermata.
Indirizzo	VARCHAR(45)	NOT NULL, UNIQUE	Rappresenta l'indirizzo della fermata.
NumeroCivico	VARCHAR(45)	NOT NULL, UNIQUE	Specifica il numero civico della fermata.
CoordinataX	DOUBLE	NOT NULL	Specifica la latitudine della fermata.
CoordinataY	DOUBLE	NOT NULL	Specifica la longitudine della fermata.
PathRow	INT	DEFAULT NULL	Valore utili per il calcolo del percorso minimo.
Costo	DOUBLE	DEFAULT NULL	Valore utili per il calcolo del percorso minimo.
Calculated	INT	DEFAULT NULL	Valore utili per il calcolo del percorso minimo.

Autobus

Nome Colonna	Tipo	Vincoli	Descrizione
Targa	CHAR(7)	PK	Individua univocamente un autobus.
PostiSeduti	INT(2)		Specifica il numero di posti seduti sull'autobus.
PostiInPiedi	INT(2)		Specifica il numero di posti in piedi sull'autobus.
PostiPerDisabili	INT(2)		Specifica il numero di posti per disabili sull'autobus.
StatoPresenza	TINYINT(1)	NOT NULL	Determina lo stato di presenza dell' autobus nel deposito.
StatoGuasto	TINYINT(1)	NOT NULL	Determina lo stato del guasto dell'autobus.
DescrizioneGuasto	VARCHAR(254)		Descrive il guasto riscontrato sull'autobus.

RegistroDeposito

Nome Colonna	Tipo	Vincoli	Descrizione
RefCodiceFiscale	CHAR(16)	FK, NOT NULL	Riferimento al Codice Fiscale del dipendente.
RefTarga	CHAR(7)	FK UNIQUE	Riferimento alla targa dell'autobus.
Giorno	DATE	NOT NULL, UNIQUE	Individua la data di uscita dell'autobus dal deposito.
OraUscita	TIME		Specifica l'orario di uscita dell'autobus dal deposito.
OraEntrata	TIME	NOT NULL, UNIQUE	Specifica l'orario di entrata dell'autobus nel deposito.

Turno

Nome Colonna	Tipo	Vincoli	Descrizione
Giorno	DATE	PK	Specifica la data del turno.
RefCodiceFiscale	CHAR(16)	PK, FK	Riferimento al Codice Fiscale del dipendente.
OraInizioTurno	TIME	PK	Specifica l'orario di inizio turno.
OraFineTurno	TIME	NOT NULL	Specifica l'orario di fine turno.
RefIdLinea	INT	FK	Riferimento all'id della linea.

Linea

Nome Colonna	Tipo	Vincoli	Descrizione
idLinea	INT	PK	Individua univocamente una linea.
OraInizioCorsa	TIME		Specifica l'orario della prima corsa.
OraFineCorsa	TIME		Specifica l'orario dell'ultima corsa.
Durata	TIME		Tempo che occorre per percorrere interamente il percorso della linea.
NumeroBusConsigliato	INT(3)		Indica il numero di autobus consigliati per la linea.
OraInizioCorsaFestivo	TIME		Specifica l'orario della prima corsa nei giorni festivi.
OraFineCorsaFestivo	TIME		Specifica l'orario dell'ultima corsa nei giorni festivi.
DurataFestivo	TIME		Tempo che occorre per percorrere interamente il percorso della linea nei giorni festivi.
NumeroBusConsigliatoFestivo	INT(3)		Indica il numero di autobus consigliati per la linea nei giorni festivi.
CapA	INT	FK	Primo capolinea
CapB	INT	FK	Secondo capolinea

LineaAssociaFermata

Nome Colonna	Tipo	Vincoli	Descrizione
RefIdLinea	INT	PK, FK	Riferimento all'id della linea.
RefIdFermata	INT	PK, FK	Riferimento all'id della fermata.
Posizione	INT	NOT NULL	Posizione della fermata all'interno della linea.
idAssocia	INT	AUTO_INCREMENT, INDEX	ID associato alla relazione tra fermata e linea utile per Dijkstra

Cedolino

Nome Colonna	Tipo	Vincoli	Descrizione
idCedolino	INT	PK, AUTO_INCREMENT	Identifica univocamente un cedolino.
RefCodiceFiscale	CHAR(16)	PK, FK	Riferimento al Codice Fiscale del dipendente.
Data	DATE		Specifica la data della compilazione del cedolino.
OreMensili	INT(8)		Specifica il numero delle ore lavorative che il dipendente deve effettuare in un mese.
OreLavorative	INT(8)		Specifica le ore lavorative effettive che il dipendente ha effettuato nel mese.
OreMalattia	INT(8)		Specifica il numero delle ore di malattia del dipendente nel mese.
OreFerie	INT(8)		Specifica il numero delle ore di ferie del dipendente nel mese.
TotalePagamento	DOUBLE		Rappresenta l'importo mensile di pagamento elargito dall'azienda al dipendente.
FileCedolino	TEXT		Identifica il nome del file contente il cedolino.

RegistroAssunzioni

Nome Colonna	Tipo	Vincoli	Descrizione
idRegistroAssunzioni	INT	PK, AUTO_INCREMENT	Identifica univocamente un'assunzione.
RefCodiceFiscale	CHAR(16)	FK, NOT NULL	Riferimento al Codice Fiscale del dipendente.
DataAssunzione	DATE	NOT NULL	Individua la data di assunzione.
DataLicenziamento	DATE		Individua l'eventuale data di licenziamento.

Dipendente

Nome Colonna	Tipo	Vincoli	Descrizione
CodiceFiscale	CHAR(16)	PK	Identifica univocamente un dipendente.
idDipendente	INT	UNIQUE, AUTO_INCREMENT, NOT NULL	Matricola del dipendente, utile a gestire il dipendente nell'azienda.
Nome	VARCHAR(30)		Rappresenta il nome del dipendente.
Cognome	VARCHAR(30)		Rappresenta il cognome del dipendente.
DataDiNascita	DATE		Specifica la data di nascita del dipendente.
Sesso	CHAR(1)		Specifica il sesso del dipendente.
LuogoDiNascita	VARCHAR(45)		Specifica il luogo di nascita del dipendente.
ProvinciaDiNascita	CHAR(2)		Specifica la provincia di nascita del dipendente.
ComuneDiResidenza	VARCHAR(45)		Specifica il comune di residenza del dipendente.
ProvinciaDiResidenza	VARCHAR(45)		Specifica la provincia di residenza del dipendente.
CAP	INT(5)		Specifica il codice di avviamento postale del dipendente.
Email	VARCHAR(45)		Specifica l'indirizzo e-mail del dipendente.
RecapitoTelefonico	VARCHAR(15)		Specifica il recapito telefonico del dipendente.
ContoCorrente	VARCHAR(45)		Specifica l'indirizzo del conto corrente del dipendente.
Password	CHAR(128)	NOT NULL	Specifica la password che permette al dipendente di accedere al sistema, criptata in SHA512.
IndirizzoDiResidenza	TEXT		Specifica l'indirizzo di residenza del dipendente.

PrimoAccesso	TINYINT(1)	NOT NULL	Permette di determinare se il dipendente deve ancora effettuare il primo accesso al sistema.
Username	TEXT	NOT NULL	Identifica l'username del dipendente, utilizzato per accedere al sistema.
RefIdRuolo	INT	FK	E' un riferimento al ruolo del dipendente.
Foto	TEXT		Identifica il nome del file contente la fototessera del dipendente.

Informazioni aggiuntive

Vista utilizzata nell'implementazione dell'algoritmo di Dijkstra contiene i costi in termini di distanza fra una fermata e la successiva appartenenti alla stessa linea:

```
CREATE VIEW costofermate (PathRow, IdLinea, FromIdFermata, ToIdFermata, Costo)
AS SELECT l1.idAssocia,  l1.RefIdLinea, f1.idFermata, f2.idFermata, lat_lng_distance(f1.CoordinataX, f1.CoordinataY, f2.CoordinataX, f2.CoordinataY)
FROM fermata f1, fermata f2, lineaassociafermata l1, lineaassociafermata l2
WHERE f1.idFermata=l1.RefIdFermata AND f2.idFermata=l2.RefIdFermata
AND l1.RefIdLinea=l2.RefIdLinea
AND (l1.Posizione=l2.Posizione-1 OR (l1.Posizione = (SELECT MAX(l3.posizione) FROM lineaassociafermata AS l3 WHERE l3.RefIdLinea=l1.RefIdLinea))
AND (l2.Posizione = (SELECT MIN(l3.posizione) FROM lineaassociafermata AS l3 WHERE l3.RefIdLinea=l1.RefIdLinea)));
```

Funzione di supporto che determina la distanza fra due punti identificati da latitudine e longitudine:

```
DELIMITER $$
CREATE FUNCTION lat_lng_distance (lat1 FLOAT, lng1 FLOAT, lat2 FLOAT, lng2 FLOAT)
    RETURNS FLOAT
    DETERMINISTIC
    BEGIN
        RETURN 6371 * 2 * ASIN(SQRT(
            POWER(SIN((lat1 - abs(lat2)) * pi()/180 / 2),
            2) + COS(lat1 * pi()/180 ) * COS(abs(lat2) *
            pi()/180) * POWER(SIN((lng1 - lng2) *
            pi()/180 / 2), 2) ));
    END$$
DELIMITER;
```

Implementazione dell'algoritmo di Dijkstra attraverso procedure MySql, si tratta della classica formulazione dell'algoritmo che prevede il calcolo dei costi elaborato nella vista precedentemente definita ("costoFermate") e ottenuto con l'ausilion dei parametri: PathRow, Costo, Calculated.

La prima parte della procedura determina tutti i percorsi minimi da un nodo (fermata) iniziale, mentre la seconda parte stabilisce relativamente alla posizione finale quale dei percorsi precedentemente calcolati sia quello d'interesse per il cliente:

```
DELIMITER $$
CREATE PROCEDURE dijkastra(fromId int, toId int, double lat, double lng)
BEGIN
  DECLARE vFromNodeID, vToNodeID, vToNodeIDNear, vNodeID, vPathID, vIdLinea INT;
  declare vCost double;
  DECLARE vFromNodeName, vToNodeName int;
  UPDATE fermata SET PathRow = NULL,Costo = NULL, Calculated = 0;

  SET vFromNodeID = fromId;
  IF vFromNodeID IS NULL THEN
    SELECT CONCAT('From node name not found.' );
  ELSE
    BEGIN
      SET vNodeID = vFromNodeID;
      SET vToNodeID = toId;
      IF vToNodeID IS NULL THEN
        SELECT CONCAT('To node name not found.' );
      ELSE
        BEGIN
          UPDATE fermata SET Costo=0 WHERE idFermata = vFromNodeID;
          WHILE vNodeID IS NOT NULL DO
            BEGIN
              UPDATE
                fermata AS src
                JOIN costofermate AS paths ON paths.FromIdFermata = src.idFermata
                JOIN fermata AS dest ON dest.idFermata = Paths.ToIdFermata
              SET dest.Costo = CASE
                                 WHEN dest.Costo IS NULL THEN src.Costo + Paths.Costo
                                 WHEN src.Costo + Paths.Costo < dest.Costo THEN src.Costo + Paths.Costo
                                 ELSE dest.Costo
                               END,
                  dest.PathRow = Paths.PathRow
              WHERE
                src.idFermata = vNodeID
                AND (dest.Costo IS NULL OR src.Costo + Paths.Costo < dest.Costo)
                AND dest.Calculated = 0;

              UPDATE fermata SET Calculated = 1 WHERE idFermata = vNodeID;

              SET vNodeID = ( SELECT idFermata FROM fermata
                              WHERE Calculated = 0 AND Costo IS NOT NULL
                              ORDER BY Costo LIMIT 1
                            );
            END;
          END WHILE;
        END;
      END IF;
    END;
  END IF;
  IF EXISTS( SELECT 1 FROM fermata WHERE idFermata = vToNodeID AND Costo IS NULL ) THEN
    SELECT CONCAT( 'Node ',vNodeID, ' missed.' );
  ELSE
    BEGIN
      SET vToNodeIDNear = (SELECT idFermata FROM fermata where lat_lng_distance(CoordinataX, CoordinataY ,  lat, lng) <= all (SELECT lat_lng_distance(CoordinataX, CoordinataY ,  lat, lng) FROM fermata WHERE idFermata<>vToNodeID));

      DROP TEMPORARY TABLE IF EXISTS map;
      CREATE TEMPORARY TABLE map (
        RowID INT PRIMARY KEY AUTO_INCREMENT,
        idNodoA int,
        idNodoB int,
        costo double,
        idLinea int
      ) ENGINE=MEMORY;
      WHILE vFromNodeID <> vToNodeID AND vFromNodeID <> vToNodeIDNear DO
        BEGIN
          SELECT
            src.idFermata,dest.idFermata,dest.Costo,dest.PathRow, Paths.idLinea
            INTO vFromNodeName, vToNodeName, vCost, vPathID, vIdLinea
          FROM
            fermata AS dest
            JOIN costofermate AS Paths ON Paths.PathRow = dest.PathRow
            JOIN fermata AS src ON src.idFermata = Paths.FromIdFermata
          WHERE dest.idFermata = vToNodeID;

          INSERT INTO Map(idNodoA,idNodoB,Costo, idLinea) VALUES(vFromNodeName,vToNodeName,vCost, vIdLinea);


          SET vToNodeID = (SELECT FromIdFermata FROM costofermate WHERE PathRow = vPathID);
        END;
      END WHILE;
      SELECT m.idNodoA, m.idNodoB, m.costo, m.idLinea, f1.CoordinataX AS CoordinataXA, f1.CoordinataY AS CoordinataYA, f2.CoordinataX AS CoordinataXB, f2.CoordinataY AS CoordinataYB
      FROM Map AS m, fermata f1, fermata f2
      WHERE f1.idFermata= m.idNodoA AND f2.idFermata=m.idNodoB ORDER BY RowID DESC;
      DROP TEMPORARY TABLE Map;
    END;
  END IF;
END;
$$
DELIMITER;
```