Gruppo OOBD2324_18

Davide Gargiulo	david.gargiulo@studenti.unina.it	N86004689
Vincenzo Di Carluccio	vi.dicarluccio@studenti.unina.it	N86004800

UNINA DELIVERY

UNIVERSITÀ
DEGLI STUDI
DI NAPOLI
FEDERICO II

Contents

1 Introduzione

1.1 Scopo Del Documento

Questo documento fornisce una panoramica dettagliata di UninaDelivery, un sistema per la gestione della logistica e delle spedizioni di merci. Il documento è stato concepito per offrire una guida completa agli utenti del sistema, delineando le sue funzionalità, architettura e il flusso di lavoro associato.

1.2 Descrizione Del Sistema

Il database di UninaDelivery è stato progettato per ottimizzare la gestione logistica delle spedizioni di merci basandosi sugli ordini dei clienti. Il sistema permette agli operatori di pianificare in modo efficiente le spedizioni, tenendo conto di variabili cruciali come la disponibilità della merce, il suo peso, e la disponibilità di mezzi di trasporto e corrieri. Questa soluzione tecnologica mira a migliorare l'efficienza operativa, ridurre i tempi di consegna e massimizzare la soddisfazione del cliente.

1.3 Utenti Intesi/Destinatari

Questa documentazione è stata creata pensando in particolare ai seguenti destinatari:

- **Operatori Logistici:** che utilizzeranno il sistema quotidianamente per la gestione e pianificazione delle spedizioni;
- **Team di Supporto e Manutenzione:** responsabili della manutenzione e dell'aggiornamento del sistema;
- **Dirigenti Aziendali:** che necessitano di comprendere le capacità e i benefici del sistema per prendere decisioni informate a livello strategico;
- **Sviluppatori di Software:** che potrebbero avere bisogno di interfacciarsi con UninaDelivery per integrazioni o personalizzazioni.

2 Progettazione concettuale

2.1 Diagramma Delle Classi UML

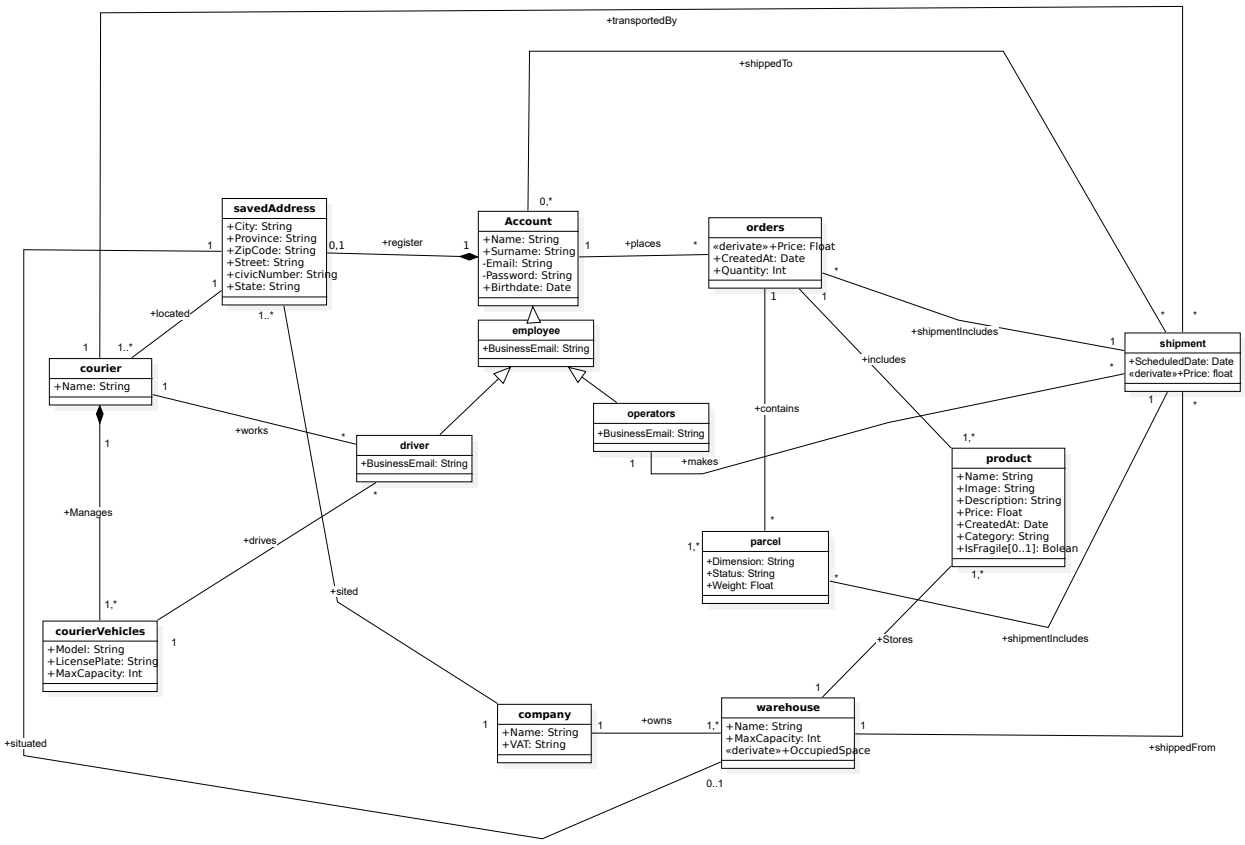

Figure 1: Diagramma delle classi UML

2.2 Leggenda del Diagramma Entità-Relazione

Nella creazione del diagramma Entità-Relazione, abbiamo impiegato diversi colori per distinguere chiaramente le varie componenti. Per assicurare la chiarezza e facilitare la comprensione del diagramma, di seguito è presentata una leggenda che spiega i simboli usati e i loro colori corrispondenti:

- **Entità**:
  - **Rettangolo Azzurro**: Rappresenta un'entità. Il nome dell'entità è posizionato all'interno del rettangolo;
  - **Rettangolo Grigio con doppia linea**: Indica un'entità debole.
- **Attributo**:
  - **Ovale Giallo**: Rappresenta un attributo. Il nome dell'attributo è posizionato all'interno dell'ovale;
  - **Ovale tratteggiato**: Indica un attributo derivato;
  - **Nome sottolineato**: Indica un attributo chiave.
- **Relazione**:
  - **Rombo Verde**: Rappresenta una relazione. Il nome della relazione è posizionato all'interno del rombo;
  - **Rombo con Doppia Linea**: Indica una relazione di tipo debole.
- **Specializzazioni**:
  - **Cerchio e Linea Rosa**: Rappresentano una specializzazione.
    - **Lettera "U" (Magnete)**: Indica la direzione della sottoclasse nella specializzazione;
    - **Lettera "d"**: Specifica una specializzazione disgiunta;
    - **Lettera "o"**: Specifica una specializzazione overlapping;
    - **Singola Linea**: Rappresenta la parzialità della specializzazione;
    - **Doppia Linea**: Rappresenta la totalità della specializzazione.

2.3 Diagramma Entità-Relazione

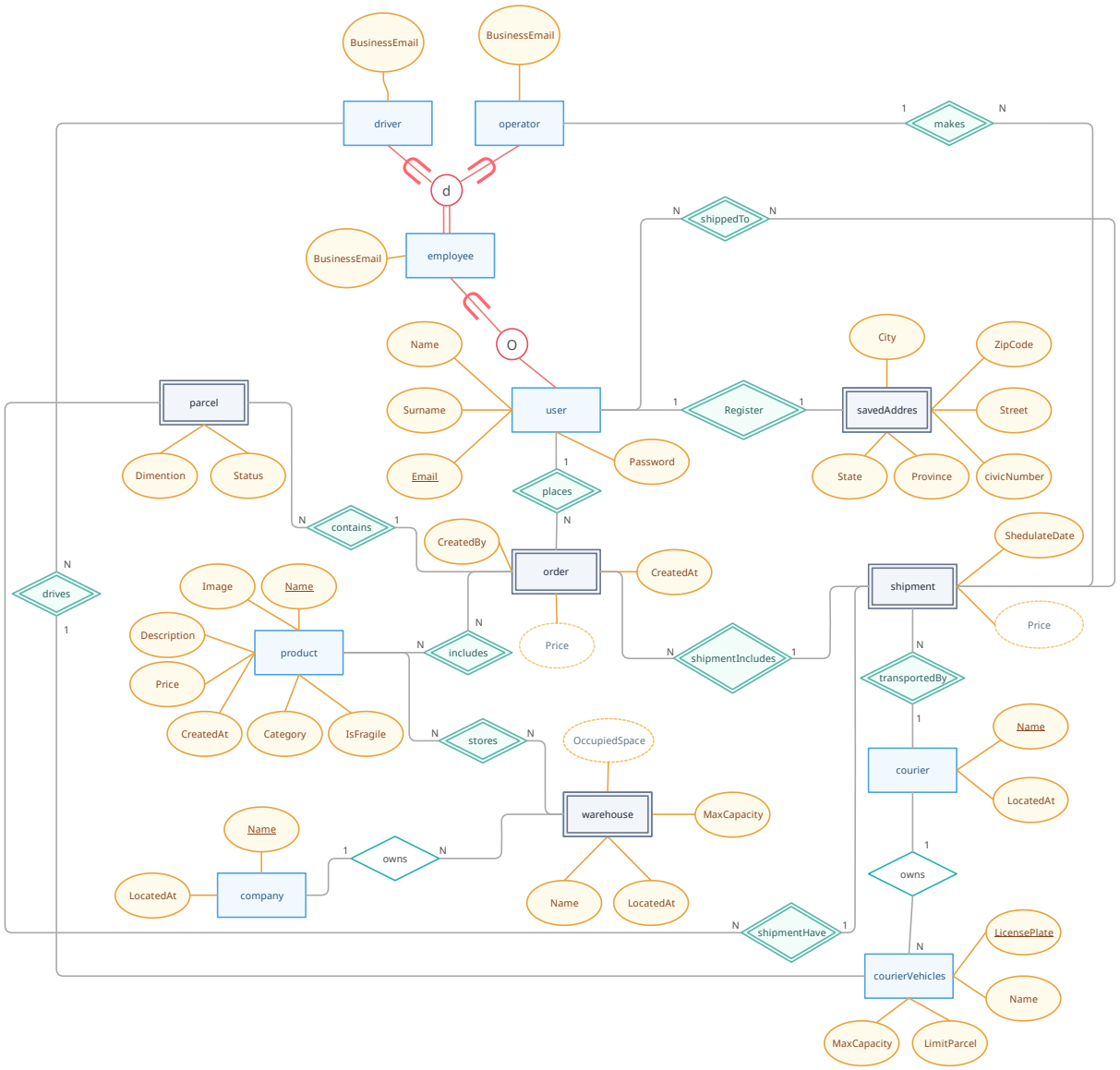

Figure 2: Diagramma Entità-Relazione

3 Ristrutturazione

3.1 Considerazioni

3.1.1 Attributi Multivalore e Composti

Durante la fase di progettazione concettuale del database, è stata presa una decisione riguardo agli attributi multivalore e composti. Abbiamo scelto di evitare l'uso di attributi multivalore o composti. Questa decisione è stata guidata dalla necessità di mantenere una struttura di database chiara e di facile gestione.

3.1.2 Specializzazioni

Le varie **specializzazioni** e le varie **generalizzazioni** sono state modellate nei seguenti modi:

- **Employee**:
  - **Tipo di Specializzazione**: Totale e disgiunta;
  - **Implementazione**: Abbiamo scelto di accorpare la classe generale **Employee** in classi specializzate.
- **Account**:
  - **Tipo di Specializzazione**: Parziale e Totale;
  - **Implementazione**: Per la specializzazione di **Account**, abbiamo deciso di trasformarla in un'associazione. Questo approccio ci ha permesso di limitare i vincoli d'integrità e di gestire in modo più efficace i campi che possono assumere valori null.

3.1.3 Attributi Derivati

Nel contesto della progettazione concettuale del database, abbiamo implementato tre attributi derivati, gestendoli nel seguente modo:

- **Attributo Price**:
  - **Gestione**: Nonostante **Price** sia un attributo derivato (tipicamente calcolato a partire da altri dati), abbiamo deciso di conservarlo direttamente nel nostro sistema.
  - **Motivazione**: Questa scelta è stata fatta per ottimizzare le prestazioni del sistema, evitando il calcolo ripetuto dei prezzi ogni volta che vengono richiesti. Conservando il prezzo calcolato, riduciamo il carico sul sistema e miglioriamo l'efficienza delle query.
- **Attributo OccupiedSpace**:
  - **Gestione**: Abbiamo scelto di conservare anche l'attributo **OccupiedSpace** all'interno del nostro sistema.
  - **Motivazione**: Questa decisione è stata presa considerando l'importanza di questo attributo, che può essere frequentemente richiesto dal sistema per la gestione degli spazi nei magazzini. Conservandolo, siamo in grado di fornire rapidamente le informazioni richieste senza la necessità di calcoli aggiuntivi.

3.1.4 Analisi delle Ridondanze

Nel processo di ottimizzazione del database, abbiamo eseguito un'analisi dettagliata per identificare e risolvere eventuali ridondanze.

- **Ridondanza negli attributi delle associazioni**:
  - **Situazione Identificata**: Dopo la ristrutturazione delle specializzazioni, abbiamo riscontrato la presenza di ridondanza negli attributi. In particolare, l'attributo **BusinessEmail** era presente sia nell'associazione **Driver** che in **Operators**, pur essendo associato a due entità diverse.
  - **Soluzione Implementata**: Per risolvere questa ridondanza, abbiamo introdotto attributi identificativi specifici per ciascuna associazione. Nell'associazione **Driver**, è stato aggiunto l'attributo **driverId** per identificare univocamente ogni driver. Analogamente, nell'associazione **Operators**, è stato introdotto **operatorId** per l'identificazione univoca degli operatori.
- **Ambiguità nei nomi delle relazioni**:
  - **Situazione Identificata**: Abbiamo notato una sovrapposizione nei nomi delle relazioni tra **Orders** e **Shipment** e tra **Parcel** e **Shipment**. Nonostante avessero lo stesso nome, queste relazioni indicavano relazioni sostanzialmente diverse.
  - **Soluzione Implementata**: Per eliminare questa ambiguità, abbiamo deciso di rinominare la relazione tra **Orders** e **Shipment** in **orderShipment**. Questo aiuta a distinguere chiaramente questa relazione da quella tra **Parcel** e **Shipment**.

3.1.5 Identificazione delle Chiavi Primarie

Abbiamo effettuato delle scelte riguardo all'uso di chiavi primarie, in particolare l'introduzione di chiavi surrogate in alcune associazioni chiave.

- **Associazione Orders**:
  - **Decisione**: Introduzione di una chiave surrogata per identificare univocamente ogni ordine chiamata **OrderId**.
  - **Motivazione**: L'uso di una chiave surrogata in questa associazione assicura che ogni ordine registrato nel sistema sia univoco. Questo non solo semplifica la gestione e il tracciamento degli ordini attuali, ma apre anche la strada per un potenziale supporto di funzionalità avanzate, come la gestione di ordini multipli o complessi in futuro.
- **Associazioni Driver e Operators**:
  - **Decisione**: Aggiunta di chiavi surrogate in entrambe le associazioni, come menzionato nell'analisi delle ridondanze.
  - **Motivazione**: Questa scelta è stata guidata dall'analisi della ridondanza, garantendo un'identificazione chiara e distinta di ogni driver e ogni operatore.
- **Associazione Company**:
  - **Decisione**: Introduzione di una chiave surrogata nell'associazione **Company**, chiamata **CompanyId**.
  - **Motivazione**: Abbiamo identificato la necessità di un identificatore unico più affidabile del solo nome aziendale, avendo riscontrato casi di aziende con lo stesso nome. La chiave surrogata assicura una distinzione precisa tra le aziende, anche in presenza di nomi omonimi.

3.1.6 Diagramma Delle Classi UML Ristrutturato

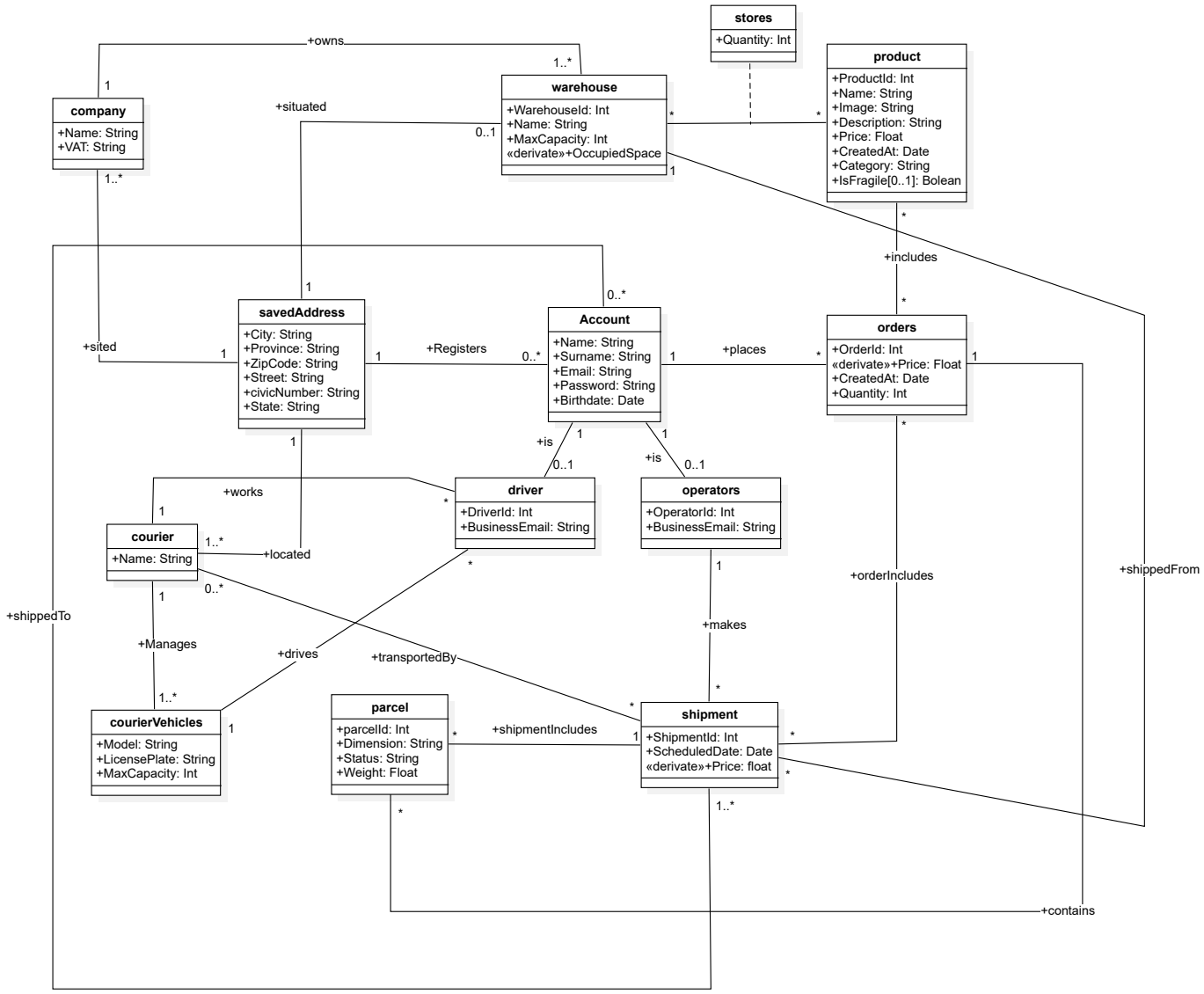

Figure 3: Diagramma delle classi UML Ristrutturato

3.2 Dizionari

3.2.1 Dizionario delle Classi

Classe	Descrizione	Attributi
Account	Utente generico che utilizza il sistema	**Name**: Nome dell'utente; **Surname**: Cognome dell'utente; **Email**: Indirizzo email (chiave primaria); **Password**: Password dell'utente. **Birthdate**: Data di nascita dell'utente;
SavedAddress	Conserva gli indirizzi salvati dagli utenti	**City**: Città in cui risiede l'utente; **Province**: Provincia in cui risiede l'utente; **ZipCode**: Codice postale della città in cui risiede l'utente; **Street**: Via in cui risiede l'utente; **CivicNumber**: Numero civico della via in cui risiede l'utente; **State**: Stato in cui risiede l'utente;
Orders	Rappresenta un ordine effettuato da un utente	**OrderId**: Identificatore univoco (chiave primaria); **CreatedBy**: Utente che ha creato l'ordine; **Price**: Prezzo totale. **Quantity**: Quantità di prodotti ordinati.
Operators	Account di un impiegato che gestisce gli ordini	**operatorId**: Identificatore univoco (chiave primaria); **BusinessEmail**: Email aziendale.
Driver	Account di un autista che consegna gli ordini	**driverId**: Identificatore univoco (chiave primaria); **BusinessEmail**: Email aziendale.
Company	Rappresenta un'azienda che vende prodotti	**Name**: Nome dell'azienda; **VAT**: Partita IVA dell'azienda (chiave primaria).
Warehouse	Rappresenta un magazzino di un'azienda	**WarehouseId**: Identificatore univoco (chiave primaria); **Name**: Nome del magazzino; **MaxCapacity**: Capacità massima del magazzino; **OccupiedSpace**: Spazio occupato attualmente nel magazzino.
Shipment	Rappresenta una spedizione	**ShipmentId**: Identificatore univoco (chiave primaria); **ScheduledDate**: Data prevista consegna; **Price**: Prezzo della spedizione.

Classe	Descrizione	Attributi
Product	Rappresenta un singolo prodotto venduto da un'azienda	**productId**: Identificatore univoco (chiave primaria); **Name**: Nome del prodotto; **Image**: Eventuale Immagine del prodotto; **Description**: Descrizione del prodotto; **Price**: Prezzo del prodotto; **CreatedAt**: Data di creazione del prodotto; **Category**: Categoria del prodotto; **IsFragile**: Indica se il prodotto è fragile.
Parcel	Rappresenta il pacco da spedire	**ParcelId**: Identificatore univoco (chiave primaria); **Dimention**: Dimensione del pacco; **Weight**: Peso del pacco; **Status**: Stato del pacco.
Courier	Rappresenta un corriere	**Name**: Nome del corriere (chiave primaria).
CourierVehicle	Rappresenta un veicolo utilizzato da un corriere	**Model**: Modello del veicolo; **LicensePlate**: Targa del veicolo (chiave primaria); **MaxCapacity**: Capacità massima del veicolo.

3.2.2 Dizionario delle Associazioni

Associazione	Descrizione	Classi Coinvolte
Places	Rappresenta la relazione tra un **Account** e un **Orders**.	**Account** [1] (places): L'account che effettua un ordine; **Orders** [∗] (placed by): Gli ordini effettuati da un account.
Registers	Rappresenta la relazione tra un **Account** e un **SavedAddress**.	**Account** [0..1] (Registers): L'account che ha salvato l'indirizzo; **SavedAddress** [1] (isRegistersBy): Gli indirizzi salvati da un account.
Is	Rappresenta la relazione tra un **Account** e un **Operators**.	**Account** [1] (Is): L'account identificato come utente; **Operators** [0..1] (has): Gli operatori associati a un account.
Is	Rappresenta la relazione tra un **Account** e un **Driver**.	**Account** [1] (Is): L'account identificato come utente; **Driver** [0..1] (has): I driver associati a un account.

Associazione	Descrizione	Classi Coinvolte
Owns	Rappresenta la relazione tra un **Company** e un **Warehouse**.	**Company** [1] (owns): La compagnia che possiede il magazzino; **Warehouse** [1..*] (owned by): I magazzini posseduti da una data compagnia.
ShippedFrom	Rappresenta la relazione tra un **Warehouse** e un **Shipment**.	**Warehouse** [1] (shippedFrom): Il magazzino da cui parte la spedizione; **Shipment** [*] (comes from): Le spedizioni partite da un dato magazzino.
TransportedBy	Rappresenta la relazione tra un **Shipment** e un **Courier**.	**Shipment** [*] (transportedBy): Le spedizioni effettuate; **Courier** [0..*] (transports): I corrieri che trasportano le spedizioni.
ShippedTo	Rappresenta la relazione tra un **Shipment** e un **Account**.	**Shipment** [1..*] (shippedTo): Le spedizioni effettuate; **Account** [0..*] (receives): Gli utenti destinatari delle spedizioni.
OrderIncludes	Rappresenta la relazione tra un **Orders** e un **Shipment**.	**Orders** [*] (ordersIncludedIn): Ordini inclusi nella spedizione; **Shipment** [*] (shipmentsContaining): Spedizioni che includono gli ordini.
ShipmentIncludes	Rappresenta la relazione tra un **Shipment** e un **Parcel**.	**Shipment** [1] (shipmentsIncluding): Spedizioni che includono pacchi; **Parcel** [*] (parcelsIncludedIn): Pacchi inclusi nelle spedizioni.
Stores	Rappresenta la relazione tra un **Warehouse** e un **Product**.	**Warehouse** [*] (stores): Magazzini che immagazzinano prodotti; **Product** [*] (is stored): Prodotti conservati nel magazzino.
Includes	Rappresenta la relazione tra un **Orders** e un **Product**.	**Orders** [1] (is included): Ordini che includono prodotti; **Product** [1..*] (includes): Prodotti inclusi negli ordini.

Associazione	Descrizione	Classi Coinvolte
Contains	Rappresenta la relazione tra un **Orders** e un **Parcel**.	**Orders** [1] (contains): Ordini che contengono pacchi; **Parcel** [∗] (is contained): Pacchi inclusi negli ordini.
Drives	Rappresenta la relazione tra un **Driver** e un **CourierVehicle**.	**Driver** [∗] (drives): Conducenti che guidano veicoli; **CourierVehicle** [1] (is drived): Veicoli guidati dai conducenti.
Manages	Rappresenta la relazione tra un **CourierVehicle** e un **Courier**.	**CourierVehicle** [1..∗] (isManagesBy): Veicoli utilizzati dai corrieri; **Courier** [1] (manages): Corrieri che hanno i veicoli.
Makes	Rappresenta la relazione tra un **Operators** e un **Shipment**.	**Operators** [1..∗] (makes): Operatori che organizzano spedizioni; **Shipment** [1] (is made): Spedizioni organizzate dagli operatori.
Sited	Rappresenta la relazione tra **Company** e **SavedAddress**.	**Company** [1] (sited): Aziende che hanno sedi; **SavedAddress** [1..∗] (is sited in): Sedi delle aziende.
Located	Rappresenta la relazione tra **Courier** e **SavedAddress**.	**Courier** [1] (located): Corrieri che hanno sedi; **SavedAddress** [1..∗] (is located in): Sedi dei corrieri.
Works	Rappresenta la relazione tra **Driver** e **Courier**.	**Driver** [1] (works): Conducenti che lavorano per corrieri; **Courier** [1] (employs): Corrieri che impiegano conducenti.

3.2.3 Dizionario dei Vincoli

Vincolo	Classi Coinvolte	Tipo	Descrizione
alfanumName	Account	Dominio	Il nome dell'utente deve contenere solo caratteri alfanumerici (lettere e numeri).
alfanumSurname	Account	Dominio	Il cognome dell'utente deve includere solamente caratteri alfanumerici.
alfnumEmail	Account	Dominio	L'indirizzo email dell'utente deve essere composto da caratteri alfanumerici e seguire il formato standard email.
alfanumDriverEmail	Driver	Dominio	L'indirizzo email del driver deve essere alfanumerico e seguire il formato standard email.
alfanumOperatorEmail	Operators	Dominio	L'indirizzo email dell'operatore deve essere alfanumerico e conformarsi al formato standard email.
capacityMaxCheck	Warehouse	Dominio	La capacità massima del magazzino deve essere un valore intero positivo.
occupiedSpaceCheck	Warehouse	Dominio	Lo spazio attualmente occupato nel magazzino deve essere espresso come un valore intero positivo.
capacityMaxVehicleCheck	CourierVehicle	Dominio	La capacità massima di carico del veicolo corriere deve essere indicata come un numero intero positivo.

Vincolo	Classi Coinvolte	Tipo	Descrizione
alfanumCategory	Product	Dominio	La categoria del prodotto deve essere definita utilizzando caratteri alfanumerici.
alfanumProductName	Product	Dominio	Il nome del prodotto deve essere formato da caratteri alfanumerici.
alfanumDescription	Product	Dominio	La descrizione del prodotto deve includere solo caratteri alfanumerici.
productPriceCheck	Product	Dominio	Il prezzo del prodotto deve essere un valore decimale positivo.
quantityProductCheck	Product	Dominio	La quantità di ogni prodotto deve essere indicata con un numero intero positivo.
checkBirthdateDriverOperator	Account, Driver	Intrarelazionale	Se l'utente è un driver, la sua età deve essere maggiore o uguale a 18 anni.
updateCapacityAfterUpdate	Warehouse, Stores	Intrarelazionale	Se è stato prelevato un prodotto da un certo magazzino, viene modificato OccupiedSpace

4 Progettazione Logica

4.1 Traduzione delle Associazioni

Associazione	Traduzione
places	Migrazione della chiave primaria di **Account** in **Orders**
contains	Migrazione della chiave primaria di **Product** in **Orders**
drives	Migrazione della chiave primaria di **CourierVehicle** in **Driver**
transportedBy	Creazione di una nuova tabella TransportedBy con migrazioni delle chiavi primarie di **Shipment** e **Courier**
manages	Migrazione della chiave primaria di **CourierVehicle** in **Courier**
located	Migrazione della chiave primaria di **Courier** in **SavedAddress**
works	Migrazione della chiave primaria di **Courier** in **Driver**
shipmentIncludes	Migrazione della chiave primaria di **Shipment** in **Parcel**
is	Migrazione della chiave primaria di **Account** in Operator
is	Migrazione della chiave primaria di **Account** in **Driver**
shippedTo	Creazione di una nuova tabella ShippedTo con migrazioni delle chiavi primarie di **Shipment** e **Account**
makes	Migrazione della chiave primaria di **Operator** in **Shipment**
sited	Migrazione della chiave primaria di **Company** in **SavedAddress**
owns	Migrazione della chiave primaria di **Company** in **Warehouse**
situated	Migrazione della chiave primaria di **Warehouse** in **SavedAddress**
registers	Migrazione della chiave primaria di **Account** in **SavedAddress**
stores	Creazione di una nuova tabella Stores con migrazioni delle chiavi primarie di **Warehouse** e **Product**
includes	Creazione di una nuova tabella Includes con migrazioni delle chiavi primarie di **Product** e **Orders**
orderIncludes	Creazione di una nuova tabella OrderIncludes con migrazioni delle chiavi primarie di **Orders** e **Shipment**
shippedFrom	Migrazione della chiave primaria di **Shipment** in **Warehouse**

Account
Name, Surname, Email, Password, Birthdate

CourierVehicle
LicensePlate, Model, MaxCapacity

Driver
DriverId, BusinessEmail, *Email*, *LicensePlate*

Orders
OrderId, Price, CreatedAt, Quantity, *Email*

Operators
OperatorId, BusinessEmail, *Email*

Shipment
ShipmentId, ScheduledDate, Price, *OperatorId*

Parcel
ParcelId, Dimention, Status, Weight, *ShipmentId*

Company
VAT, Name

Warehouse
WarehouseId, Name, MaxCapacity, OccupiedSpace, *VAT*, *ShipmentId*

SavedAddress
AddressId, City, Province, ZipCode, Street, CivicNumber, State, *Name*, *VAT*, *WarehouseId*, *Email*

Product
ProductId, Name, Image, Description, Price, CreatedAt, Category, IsFragile

Stores
Quantity, *WarehouseId*, *ProductId*

Courier
Name, *LicensePlate*

TransportedBy
ShipmentId, *Name*

ShippedTo
ShipmentId, *Email*

includes
OrderId, *ProductId*

5 Schema Fisico

5.1 Creazione delle Tabelle

5.1.1 Account

```
CREATE TABLE Account (
    Name VARCHAR(255),
    Surname VARCHAR(255),
    Email VARCHAR(255) PRIMARY KEY,
    Password VARCHAR(255),
    Birthdate DATE
);
ALTER TABLE Account
ADD CONSTRAINT alfanumName CHECK (Name ~ '^[A-Za-z0-9]+$'),
ADD CONSTRAINT alfanumSurname CHECK (Surname ~ '^[A-Za-z0-9]+$'),
ADD CONSTRAINT alfanumEmail CHECK (Email ~ '^[A-Za-z0-9@.]+$');
```

5.1.2 CourierVehicle

```
CREATE TABLE CourierVehicle (
    LicensePlate VARCHAR(255) PRIMARY KEY,
    Model VARCHAR(255),
    MaxCapacity INT
);
ALTER TABLE CourierVehicle
ADD CONSTRAINT capacityMaxVehicleCheck CHECK (MaxCapacity > 0);
```

5.1.3 Driver

```
CREATE TABLE Driver (
    DriverId INT PRIMARY KEY,
    BusinessEmail VARCHAR(255),
    Email VARCHAR(255),
    LicensePlate VARCHAR(255),
    FOREIGN KEY (Email) REFERENCES Account(Email),
    FOREIGN KEY (LicensePlate) REFERENCES CourierVehicle(LicensePlate)
);
ALTER TABLE Driver
ADD CONSTRAINT alfanumDriverEmail CHECK (BusinessEmail ~ '^[A-Za-z0-9@.]+$');
```

5.1.4 Operator

```
CREATE TABLE Operators (
    OperatorId INT PRIMARY KEY,
    BusinessEmail VARCHAR(255),
    Email VARCHAR(255),
    FOREIGN KEY (Email) REFERENCES Account(Email)
);
ALTER TABLE Operators
ADD CONSTRAINT alfanumOperatorEmail CHECK (BusinessEmail ~ '^[A-Za-z0-9@.]+$');
```

5.1.5 Warehouse

```
CREATE TABLE Warehouse (
    WarehouseId INT PRIMARY KEY,
    Name VARCHAR(255),
    MaxCapacity INT,
    OccupiedSpace INT,
    VAT VARCHAR(255),
    ShipmentId INT,
    FOREIGN KEY (VAT) REFERENCES Company(VAT),
    FOREIGN KEY (ShipmentId) REFERENCES Shipment(ShipmentId)
);
ALTER TABLE Warehouse
ADD CONSTRAINT capacityMaxCheck CHECK (MaxCapacity > 0),
ADD CONSTRAINT occupiedSpaceCheck CHECK (OccupiedSpace >= 0);
```

5.1.6 Shipment

```
CREATE TABLE Shipment (
    ShipmentId INT PRIMARY KEY,
    ScheduledDate DATE,
    Price DECIMAL(10, 2),
    OperatorId INT,
    FOREIGN KEY (OperatorId) REFERENCES Operators(OperatorId)
);
```

5.1.7 Parcel

```
CREATE TABLE Parcel (
    ParcelId INT PRIMARY KEY,
    Dimension VARCHAR(255),
    Status VARCHAR(255),
    Weight DECIMAL(10, 2),
    ShipmentId INT,
    FOREIGN KEY (ShipmentId) REFERENCES Shipment(ShipmentId)
);
```

5.1.8 Company

```
CREATE TABLE Company (
    VAT VARCHAR(255) PRIMARY KEY,
    Name VARCHAR(255)
);
```

5.1.9 Product

```
CREATE TABLE Product (
    ProductId INT PRIMARY KEY,
    Name VARCHAR(255),
    Image VARCHAR(255),
    Description TEXT,
    Price DECIMAL(10, 2),
    CreatedAt DATE,
    Category VARCHAR(255),
    IsFragile BOOLEAN
);
ALTER TABLE Product
ADD CONSTRAINT alfanumCategory CHECK (Category ~ '^[A-Za-z0-9]+$'),
ADD CONSTRAINT alfanumProductName CHECK (Name ~ '^[A-Za-z0-9]+$'),
ADD CONSTRAINT alfanumDescription CHECK (Name ~ '^[A-Za-z0-9]+$'),
ADD CONSTRAINT productPriceCheck CHECK (Price > 0);
```

5.1.10 Order

```
CREATE TABLE Orders (
    OrderId INT PRIMARY KEY,
    Price DECIMAL(10, 2),
    CreatedAt DATE,
    Quantity INT,
    Email VARCHAR(255),
    ProductId INT,
    FOREIGN KEY (Email) REFERENCES Account(Email)
);
```

5.1.11 SavedAddress

```
CREATE TABLE SavedAddress (
    AddressId INT PRIMARY KEY,
    City VARCHAR(255),
    Province VARCHAR(255),
    ZipCode VARCHAR(255),
    Street VARCHAR(255),
    CivicNumber VARCHAR(255),
    State VARCHAR(255),
    Name VARCHAR(255),
    VAT VARCHAR(255),
    WarehouseId INT,
    Email VARCHAR(255),
    FOREIGN KEY (Name) REFERENCES Courier(Name),
    FOREIGN KEY (VAT) REFERENCES Company(VAT),
    FOREIGN KEY (WarehouseId) REFERENCES Warehouse(WarehouseId),
    FOREIGN KEY (Email) REFERENCES Account(Email)
);
```

5.1.12 Stores

```
CREATE TABLE Stores (
    Quantity INT,
    WarehouseId INT,
    ProductId INT,
    FOREIGN KEY (WarehouseId) REFERENCES Warehouse(WarehouseId),
    FOREIGN KEY (ProductId) REFERENCES Product(ProductId)
);
```

5.1.13 Courier

```
CREATE TABLE Courier (
    Name VARCHAR(255) PRIMARY KEY,
    LicensePlate VARCHAR(255),
    FOREIGN KEY (LicensePlate) REFERENCES CourierVehicle(LicensePlate)
);
```

5.1.14 TransportedBy

```
CREATE TABLE TransportedBy (
    ShipmentId INT,
    Name VARCHAR(255),
    FOREIGN KEY (ShipmentId) REFERENCES Shipment(ShipmentId),
    FOREIGN KEY (Name) REFERENCES Courier(Name)
);
```

5.1.15 ShippedTo

```
CREATE TABLE ShippedTo (
    ShipmentId INT,
    Email VARCHAR(255),
    FOREIGN KEY (ShipmentId) REFERENCES Shipment(ShipmentId),
    FOREIGN KEY (Email) REFERENCES Account(Email)
);
```

5.1.16 Includes

```
CREATE TABLE Includes (
    OrderId INT,
    ProductId INT,
    FOREIGN KEY (OrderId) REFERENCES Orders(OrderId),
    FOREIGN KEY (ProductId) REFERENCES Product(ProductId)
);
```

5.2 Definizione del Trigger

5.2.1 checkBirthdateDriverOperator

```
CREATE OR REPLACE FUNCTION checkAgeBeforeInsertOrUpdateDriver()
RETURNS TRIGGER AS $$
DECLARE
    age INT;
BEGIN
    -- Calcola l'età basata sulla data di nascita
    SELECT EXTRACT(YEAR FROM age(CURRENT_DATE, Birthdate)) INTO age
    FROM Account WHERE Email = NEW.Email;

    -- Controlla se l'età è inferiore a 18
    IF age < 18 THEN
        RAISE EXCEPTION 'L'autista deve avere almeno 18 anni.';
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER checkBirthdateDriver
BEFORE INSERT ON Driver
FOR EACH ROW
EXECUTE FUNCTION checkAgeBeforeInsertOrUpdateDriver();
```

5.2.2 updateCapacityAfterUpdate

```sql
CREATE OR REPLACE FUNCTION update_warehouse_capacity()
RETURNS TRIGGER AS $$
BEGIN
    UPDATE Warehouse
    SET OccupiedSpace = OccupiedSpace + NEW.Quantity - OLD.Quantity
    WHERE WarehouseId = NEW.WarehouseId;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER updateCapacityAfterUpdate
AFTER UPDATE ON Stores
FOR EACH ROW EXECUTE FUNCTION update_warehouse_capacity();
```

5.3 Definizione della Funzione

```sql
CREATE OR REPLACE FUNCTION process_new_order(
    p_order_id INT,
    p_product_id INT,
    p_email VARCHAR,
    p_quantity INT,
    p_price DECIMAL(10,2),
    p_created_at DATE
)
RETURNS VOID AS $$
DECLARE
    available_quantity INT;
BEGIN
    -- Query per ottenere la quantità disponibile del prodotto richiesto dal magazzino.
    SELECT Quantity INTO available_quantity FROM Stores WHERE ProductId = p_product_id;
    IF available_quantity < p_quantity THEN
        RAISE EXCEPTION 'Quantità insufficiente nel magazzino.';
    END IF;

    -- Inserimento del nuovo ordine nella tabella Orders.
    INSERT INTO Orders (OrderId, Price, CreatedAt, Quantity, Email, ProductId)
    VALUES (p_order_id, p_price, p_created_at, p_quantity, p_email, p_product_id);

    -- Aggiornamento della quantità del prodotto nel magazzino
    UPDATE Stores
    SET Quantity = Quantity - p_quantity
    WHERE ProductId = p_product_id;
END;
$$ LANGUAGE plpgsql;
```

6 Conclusione

Con il completamento di questa documentazione, abbiamo fornito una panoramica dettagliata del sistema UninaDelivery, delineando le sue componenti chiave, la struttura dei dati, le associazioni tra le varie classi, e i vincoli di dominio. Questo documento serve come un punto di riferimento essenziale per comprendere l'architettura e il funzionamento del sistema, offrendo una guida utile sia per gli sviluppatori che per gli utenti finali.

Le specifiche presentate in questa documentazione sono state studiate per garantire che il sistema UninaDelivery sia robusto, efficiente e facile da utilizzare. La progettazione del database, insieme alle relazioni definite e ai vincoli applicati, assicura l'integrità, la sicurezza e la coerenza dei dati gestiti. Inoltre, l'attenzione ai dettagli nella definizione delle classi e delle loro relazioni è volta a facilitare l'espansione e la manutenzione futura del sistema.