

Progettazione e Sviluppo di una Web
Application in Java EE per la gestione delle
spedizioni di una società di trasporti e
logistica

Alessio Bonadio, Cosimo Casini, Riccardo Grazzi

10 luglio 2017



Indice

1	Introduzione	4
2	Requisiti	5
2.1	Requisiti Funzionali	5
2.2	Requisiti sui dati	5
2.3	Requisiti Operativi	6
3	Progettazione e sviluppo	7
3.1	Casi D'uso	7
3.2	Modello Concettuale	8
3.3	Page Navigation Diagram	9
3.3.1	Mockups	10
3.4	Domain Model	10
3.5	Architettura Implementativa	12
3.5.1	Domain Logic	12
3.5.2	Domain Model	14
3.5.3	DAO	15
3.5.4	Business Logic	16
4	Test	19
5	Interfaccia Web	21
5.1	Home Page	21
6	Considerazioni finali e sviluppi futuri	23
A	Mockup	24

Capitolo 1

Introduzione

Lo scopo dell'elaborato è quello di progettare e realizzare un sistema informatico che permetta di gestire le spedizioni in un'azienda di trasporti e logistica, tenendo in considerazione anche i diversi ruoli attribuiti agli utenti. Il lavoro si è articolato in diverse fasi: inizialmente l'analisi dei requisiti, una fase di progettazione vera e propria, una di Test e infine un primo approccio all'implementazione dell'interfaccia Web. Questa relazione è articolata in modo da ripercorrere esattamente i passi descritti. Saranno dettagliate le varie fasi e riportati i procedimenti che hanno portato allo sviluppo del sistema.

Capitolo 2

Requisiti

2.1 Requisiti Funzionali

Il sistema deve permettere la gestione delle “bolle” di spedizione (**waybills**) da parte degli operatori del corriere (**operators**), dei guidatori dei camion (**drivers**) e dei clienti (**customers**). La bolla può essere creata anche dal cliente ma deve passare la validazione di un operatore. La bolla creata dall’operatore è automaticamente accettata al momento della creazione.

Il sistema permette agli operatori e ai clienti (aziende o privati) di **autenticarsi** con username e password. Ogni operatore appartiene ad una filiale, le bolle sono associate alla filiale dell’operatore che le crea/valida.

Un operatore deve poter fare il **CRUD del cliente** e delle relative tariffe: aggiungere un nuovo cliente, aggiungere e modificarne le tariffe. Inoltre può **Assegnare i record di spedizioni ai driver** per quel giorno.

Un operatore può **Aggiungere un cliente**, che deve essere associato ad una filiale. Il cliente a sua volta può proporre bolle che partono solo da quest’ultima.

Dopo l’accettazione un cliente deve poter visualizzare le informazioni relative al **tracking** del pacco, anch’esse contenute nella bolla.

Il sistema permette al driver di **Leggere la bolla** e **Aggiungere la firma del ricevente**, la data e l’ora di consegna nella bolla. Il driver può accedere solo alle bolle a lui assegnati.

2.2 Requisiti sui dati

- Una **Bolla** è composta da: l’operatore che la valida, il cliente che la invia, la data di validazione, l’istante di tempo in cui viene consegnata, la filiale di partenza, la filiale d’arrivo, il numero dei colli (**Items**),

il peso totale, il volume totale, il costo del trasporto (dipendente da: tariffa, peso, volume, destinazione), le informazioni sul tracking e i dati del destinatario

- I dati sul **Destinatario** di una bolla sono: nome, cognome, CAP, via, località, provincia, telefono, firma.
- Un **Utente** deve avere: username, password e un livello di accesso (Customer, Driver, Operator).
- Un **Driver** deve avere una zona di consegna, informazioni sulla disponibilità per un determinato giorno, la lista delle spedizioni per il giorno corrente e informazioni sul camion (modello, tipo, portata, volume).
- Un **Cliente** contiene: l'operatore che l'ha acquisito, nome (dell'azienda o del privato), indirizzo, stato (bloccato o meno) e la sua lista delle tariffe.
- Una **Tariffa** contiene: una funzione di prezzo basata su peso e volume della bolla, una zona (Italia o nazione estera) e le date di inizio e di scadenza.
- Una **Filiale** contiene: nome, indirizzo e altre info.

2.3 Requisiti Operativi

È richiesto lo sviluppo del sistema utilizzando Java EE con annotazioni JPA per la persistenza e CDI per la gestione dei controller. Non è richiesto lo sviluppo dell'interfaccia grafica e dei test simulativi dei casi d'uso.

Capitolo 3

Progettazione e sviluppo

3.1 Casi D'uso

Dallo studio dei requisiti funzionali sono emersi i casi d'uso mostrati nello Use Case Diagram in Figura 3.1.

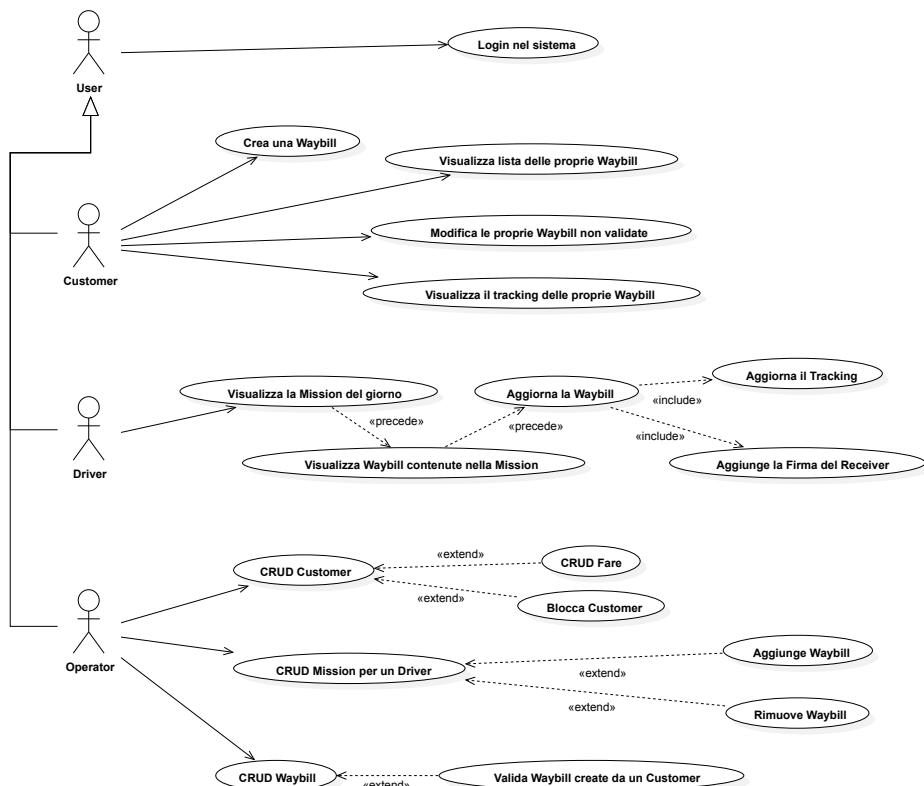


Figura 3.1: Use Case Diagram

Dai casi d'uso dell'operator, nasce il concetto di **Mission**. Una Mission corrisponde alle consegne che il driver dovrà effettuare in un determinato giorno.

3.2 Modello Concettuale

Dopo lo studio dei requisiti sui dati è stato costruito il modello concettuale mostrato in Figura 3.2.

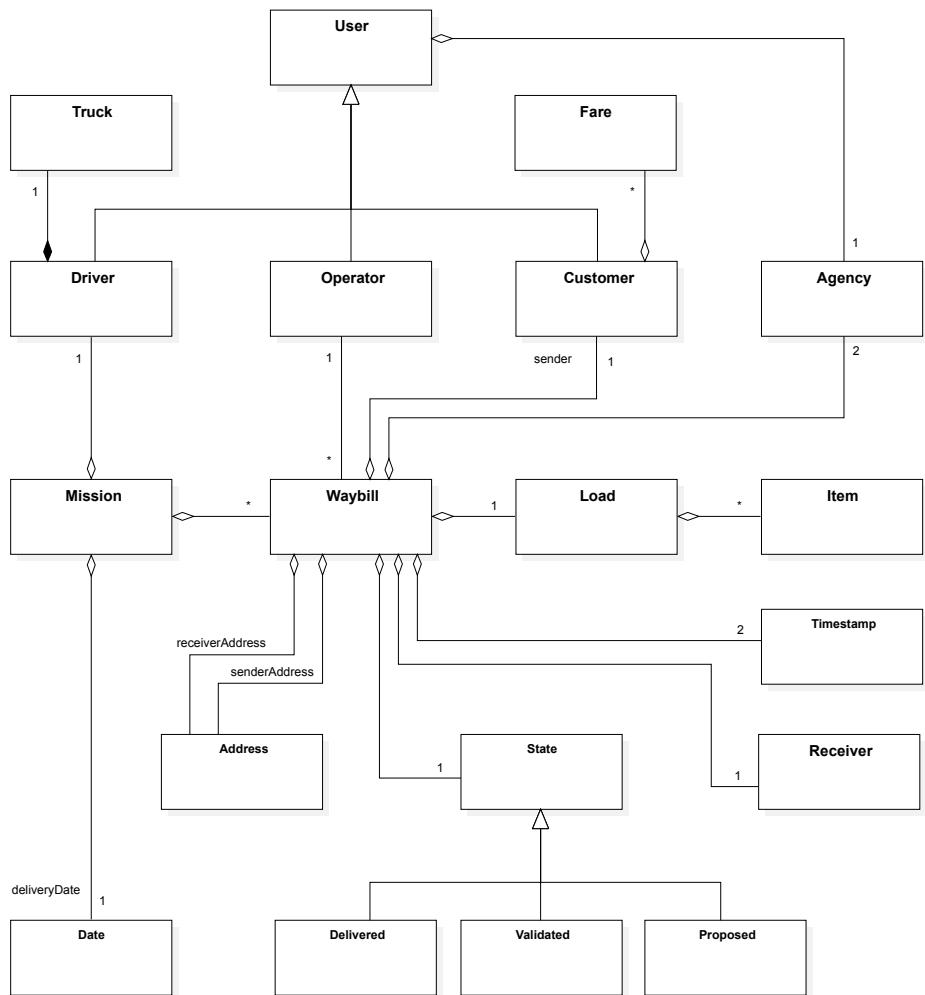


Figura 3.2: Conceptual Model

Oltre alle entità descritte precedentemente se ne sono aggiunte delle altre: **Mission** e **Load**. Alla Mission è associata una data, una lista di waybills e un driver, mentre Load contiene informazioni sul carico di una waybill quali

peso e volume totali, nonchè la lista di Items a questa associata. Un **Item** corrisponde ad un collo, unità base della spedizione, che deve avere un id proprio.

3.3 Page Navigation Diagram

Espandendo ulteriormente i casi d'uso è stato elaborato il Page Navigation Diagram mostrato in figura 3.3.

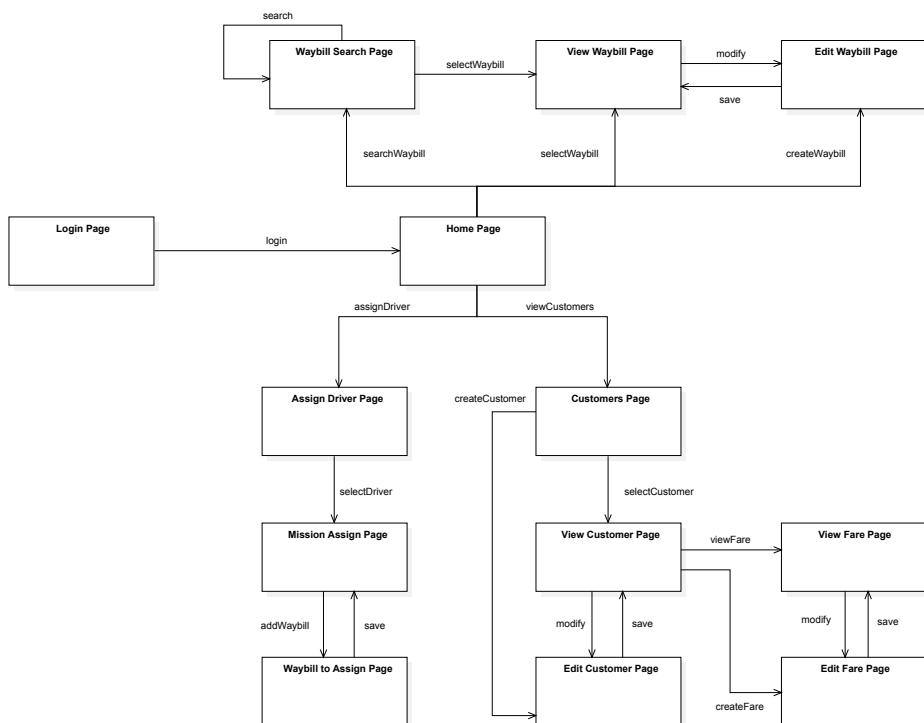


Figura 3.3: Page Navigation Diagram

La **Home Page** avrà diverse funzionalità a seconda dei ruoli dei singoli utenti, meglio esemplificate nei mockup. In particolare:

- La **Waybill Search Page** sarà accessibile solo dagli **Operator** e dai **Customer**.
- Le pagine di **View** e **Edit** della **Waybill** sono accessibili da tutti.
- Tutte le pagine sotto la **Home**, per la **gestione della mission** dei driver e per la **gestione dei clienti** sono accessibili soltanto dall'**Operator**.

3.3.1 Mockups

Dal Page Navigation Diagram sono state progettate le interfacce delle singole pagine con dei mockup, alcuni dei quali sono mostrati in Figura 3.4. In Appendice A sono presenti tutti quelli elaborati.

(a) Operator Homepage

ID	Customer	Cost	Weight	Volume	# Items	Actions
A056789B3	Prods	120 €	10 kg	2	10	
A056789C2	Company	60 €	10 kg	2	7	
A056789E8	Just Company	40 €	200 kg	94	45	
A056789L2	AHC	50 €	10 kg	2	11	

ID	Customer	Cost	Weight	Volume	# Items	Actions
A056789B3	Prods	120 €	10 kg	2	10	
A056789B3	Prods	43 €	5 kg	8	3	
A056789E8	Company	420 €	145 kg	76	102	
A056789L2	AHC	50 €	10 kg	2	11	

(b) View/Edit Waybill page

Waybill ID: #123456789

FROM: Mario Rossi
Via Bracciali 232
50045 Firenze (FI)
Customer ID: 345654456573

TO: Riccardo Bianchi
Via Giorgio 45
50045 Montecatini (AR)

Total Weight (kg): 50,636
Total Volume (m³): 0,3
Packages: 3

Cost: 50,636
Fare: 118,684

Departure Date: 12/12/2012
Arrival Date: TBC
Driver Assigned: DR644362

Departure Agency: Firenze
Arrival Agency: Arezzo

Tracking: IDLE

Signature Receiver:

Back Modify

(c) Search Waybill page

Search Waybill

Extended Search

Search by ID: Departure Date: Arrived Date:

Search by Customer: Sort by: Cost

Search by Departure Agency: Search by Arrived Agency:

Search

ID	Customer	Cost	Weight	Volume	# Items	Actions
A056789B3	Prods	120 €	10 kg	2	10	
A056789B3	Prods	43 €	5 kg	8	3	
A056789E8	Company	420 €	145 kg	76	102	
A056789L2	AHC	50 €	10 kg	2	11	

Back

(d) Mission Assign page

Mission Assign Page

Mission for Driver: Mario

Available Weight: 173 / 500 Kg Available Volume: 104 / 300 m³

ID	Customer	Cost	Weight	Volume	# Items	Actions
A056789B3	Prods	120 €	10 kg	2	10	
A056789B3	Prods	43 €	5 kg	8	3	
A056789E8	Company	420 €	145 kg	76	102	
A056789L2	AHC	50 €	10 kg	2	11	

Back Add Waybill

Figura 3.4: Mockup di alcune pagine

3.4 Domain Model

Il modello concettuale è stato quindi ripensato considerando le caratteristiche di Java EE e delle annotazioni JPA giungendo al **modello di dominio** mostrato in Figura 3.5.

È stata fatta una distinzione tra classi di tipo **@Entity** ed **@Embeddable** in cui le prime vengono mappate come singole tabelle nel database, mentre le seconde come colonne interne all'entità che le possiedono. Un esempio è la classe **Truck** che è mappata internamente come colonne della tabella drivers. Si noti come le classi annotate come **@Embeddable** abbiano una relazione di composizione nel Domain Model.

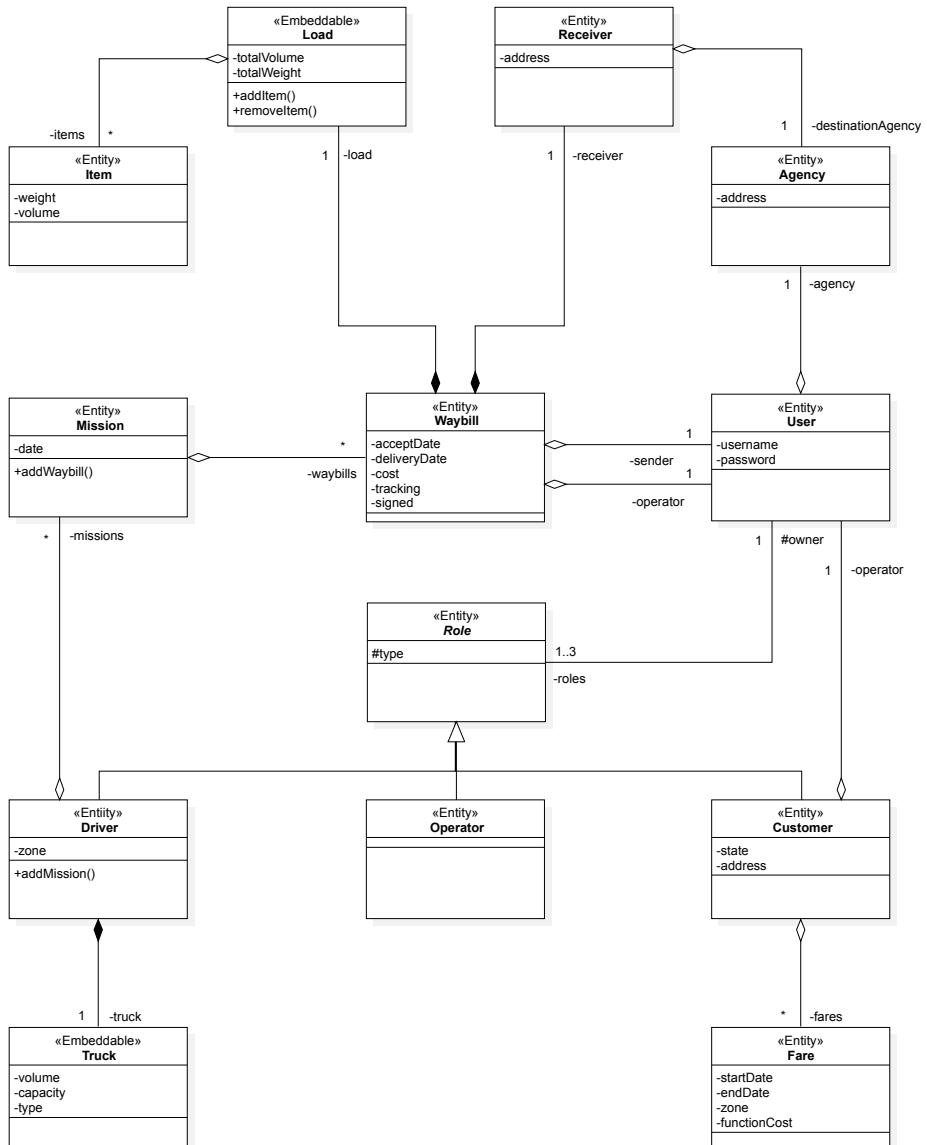


Figura 3.5: Domain Model

Inoltre nel modello di dominio sono stati trasformati in attributi tutti i concetti enumerabili, come lo stato del cliente, e quelli per i quali non c'è la necessità di creare una classe apposita, come le date e i timestamps.

Sono state effettuate anche le seguenti modifiche:

- Si sono separati gli utenti (**User**) dai ruoli (**Role**). In questo modo è stato possibile creare utenti con più di un ruolo. I constraint sulla waybill sono stati quindi realizzati con un pattern **Accountability**. La waybill rappresenta una relazione di accountability tra 2 User: Sender e Operator. Quindi, User è un **Party**, mentre Role, con il suo type (enum con valori CUSTOMER, DRIVER e OPERATOR), rappresenta il **PartyType**. L'**AccountabilityType** non è stato realizzato, in quanto non deve essere possibile fare nuove accountability a run-time.
- Considerando che la filiale (Agency) di partenza è sempre la stessa per ciascun cliente, e quella di arrivo è sempre la stessa per ciascun destinatario, si sono tolte le aggregazioni dirette tra queste e la waybill.

3.5 Architettura Implementativa

3.5.1 Domain Logic

Il modello usato per la logica di dominio è di tipo 3-Tier, come mostrato in Figura 3.6:

- **Domain Model**: contiene tutte le entità del sistema con i relativi mapping su database. Questa parte è implementata utilizzando la specifica JPA tramite il framework **Hibernate**.
- **DAO**: ogni entità nel Domain Model possiede un suo DAO, che gestisce le comunicazioni con il database per il recupero e la persistenza delle entità attraverso l'**Entity Manager**. I DAO forniscono un ulteriore livello di astrazione tra Dominio e Database. In essi sono contenute tutte le query JPQL utilizzate dal sistema.
- **Business Logic**: implementa la logica di controllo delle pagine web attraverso i **Controller**. Questi utilizzano i DAO per il recupero e la persistenza delle entità del dominio. Inoltre implementano le funzioni che operano sulle entità e sono alla base dei casi d'uso.

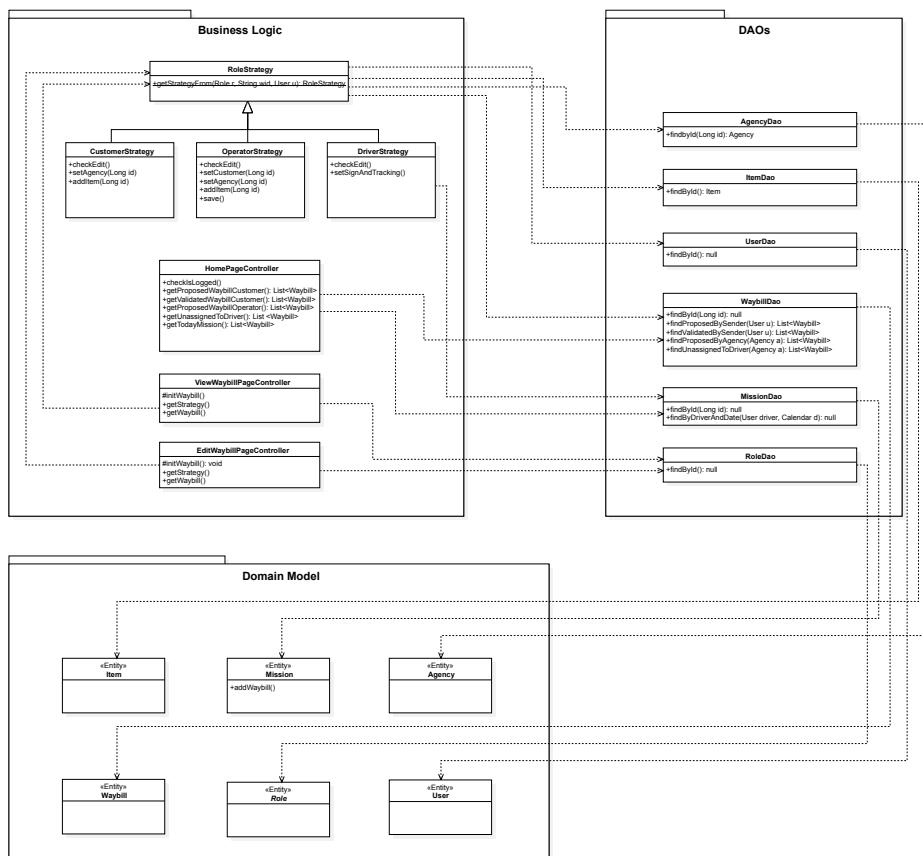


Figura 3.6: 3-Tier Model

3.5.2 Domain Model

Per l'implementazione del modello di dominio si è fatto uso di una classe astratta chiamata **BaseEntity**, che viene estesa da tutte le entità del modello e permette una migliore gestione degli **id**, degli **uuid** (Universally Unique IDentifiers) e del metodo **equals**, che confronta gli uuid.

Per vincolare alcuni attributi a non essere nulli quando vengono persistiti nel database, è stata usata la notazione JPA **@NotNull**. Alcuni esempi sono gli attributi **username** e **password** della classe User, e gli attributi **operator** e **receiver** della classe Waybill.

Per modellare le relazioni tra le varie entità sono state usate le notazioni **@ManyToOne** e **@OneToMany**. In particolare vengono utilizzate singolarmente nel caso di una relazione direzionale o entrambe nel caso di una bidirezionale. Ad esempio, l'aggregazione tra Mission e Waybill è annotata come **@OneToMany** nella classe Mission (una mission può avere più waybill) e l'associazione tra User e Role come **@OneToOne** nella classe User e **@ManyToOne** in quella Role. In questo modo, attraverso l'attributo **mappedBy**, nel database la tabella roles avrà una colonna chiamata **owner_id** che rimanderà all'id dello user che lo possiede, senza aver bisogno di una tabella aggiuntiva.

Un'altra notazione usata in Role è **@Transient**. Questa è stata applicata all'attributo **type**, che non viene salvato sul database poiché l'informazione sul type è contenuta nel tipo dell'oggetto stesso.

Per modellare l'ereditarietà delle classi dei ruoli è stata usata la notazione **@Inheritance(strategy=InheritanceType.JOINED)**. Con questa strategia i field specifici di una sottoclasse sono mappati in una tabella separata e i campi in comune, invece, sono inseriti nella tabella della classe padre. Così facendo, al contrario dalla strategia **TABLE_PER_CLASS**, in cui viene creata una tabella per ogni classe, si può recuperare un ruolo senza conoscere prima la sua classe concreta, utilizzando **RoleDAO**.

Sono state infine utilizzati dei tipi **@Enumerated** per descrivere i vari stati degli utenti e delle bolle ma anche il tipo di truck e la firma.

I metodi delle classi del domain model sono per la quasi totalità metodi getter e setter. Tra le poche eccezioni troviamo i metodi **setSender** e **setOperator** della classe **Waybill** che devono effettuare i controlli sul ruolo. Di seguito è mostrato il metodo **setSender**:

```
// Metodo setSender in Waybill.  
public void setSender (User sender) throws IllegalArgumentException  
{  
    if (!sender.hasRole(Role.Type.CUSTOMER))
```

```

        throw new IllegalArgumentException("sender has not the
            Customer role");
    this.sender = sender;
}

```

3.5.3 DAO

Per lo sviluppo dei DAO è stata costruita una classe generica chiamata **BaseDao**, successivamente estesa da tutti i DAO, che implementa il metodo save e contiene il riferimento all'**Entity Manager**. Per ogni entità è stato implementato un suo particolare DAO. I metodi stati implementati a seconda delle necessità dei controller della business logic.

Degna di nota è la funzione di ricerca avanzata (`advancedSearch`) su **WaybillDao**. Questa permette di ricevere molti parametri per la ricerca delle waybill dove in caso di parametri uguali a `null` non viene inserita la corrispondente clausola nella query. Questo metodo in teoria potrebbe sostituire tutti i metodi di recupero della classe `WaybillDao` ma, la complessità del metodo lo rende molto difficile da testare e inadatto ad un utilizzo esteso. Per questo motivo il metodo viene usato solo sulla pagina di ricerca delle waybill e si fa riferimento a metodi più semplici per gli altri usi più specifici.

A seguire alcuni metodi degni di nota presenti nei diversi DAO:

```

// Metodo presente in WaybillDao.
public List<Waybill> findProposedBySender(User sender) {
    return entityManager.createQuery("SELECT w FROM Waybill w WHERE
        w.sender = :sender AND w.operator IS NULL",
        Waybill.class).setParameter("sender", sender).getResultList();
}

```

```

// Metodo presente in MissionDao.
public Mission findByDriverAndDate(User driver, Calendar date) {
    List<Mission> result = entityManager.createQuery("FROM Mission
        WHERE driver_id = :driver_id AND date = :date",
        Mission.class).setParameter("driver_id",
        driver.getDriverRole().getId()).setParameter("date", date,
        TemporalType.DATE).getResultList();
    if (!result.isEmpty())
        return result.get(0);
    return null;
}

```

```
}
```

3.5.4 Business Logic

Per l'implementazione di questa parte è stato utilizzato Java e le annotazioni CDI che permettono l'injection dei DAO e dei vari Bean nei controller. Si è realizzato un controller per pagina, per un totale di 14 controller. La maggior parte dei controller è annotata come `@Model`. In questo modo i controller vengono distrutti non appena la pagina JSF viene generata. Tuttavia, i controller che devono modificare, e quindi persistere entità sul database, utilizzano l'annotazione `@ViewScoped` che li tiene in vita fintanto che la pagina web rimane attiva.

Beans

I Bean utilizzati nella business logic sono 3: **UserSessionBean**, **CustomerBean** e **MissionBean**.

- **UserSessionBean** è annotato come `@SessionScoped` e contiene un riferimento a `User`, che se non è `null` indica l'utente loggato.
- **CustomerBean** è annotato come `@ConversationScoped` e mantiene il riferimento al cliente selezionato nella Customer Page. Questo permette di non dover passare ogni volta l'id del cliente nelle pagine di visualizzazione e modifica del customer e in quelle di visualizzazione e modifica delle tariffe.
- **MissionBean** è annotato come `@ConversationScoped` e mantiene il riferimento alla mission del driver selezionato precedentemente in **Assign Driver Page**, insieme alle informazioni sul camion di quest'ultimo.

HomePageController

Il controllo sui ruoli dell'utente viene effettuato dalla pagina web che implementa l'interfaccia. Quindi il controller della Home Page si limita ad implementare tutte le funzioni dei 3 ruoli. Ciascuna di queste non fa altro che recuperare attraverso i dao una lista di Waybill e renderla disponibile alla pagina web con un metodo getter.

ViewWaybillPageController, EditWaybillPageController e il Pattern Strategy

Per l'implementazione dei controller di visualizzazione e di modifica della waybill si è deciso di adottare il pattern Strategy come mostrato in Figura 3.6, in quanto queste due pagine cambiano comportamento a seconda del ruolo dell'utente. Un controllo sul ruolo dovrà necessariamente essere fatto anche nella pagina web. Ma rinforzarlo sui controller permette di evitare errori nella scrittura della pagina.

Questi controller prendono l'id del ruolo passato dalla home page e inizializzano l'oggetto **strategy** adeguato, con una delle 3 classi concrete figlie di **RoleStrategy**. Queste definiscono alcuni metodi setter che operano sulla waybill. In particolare quelli che collegano la waybill ad altre entità per aggiungere il controllo sulla loro presenza sul database.

La pagina web utilizzerà quindi direttamente l'oggetto **RoleStrategy** per chiamare i suddetti metodi, mentre per gli altri verrà utilizzata la **Waybill**.

EditCustomerPageController e il CustomerBean

Un esempio di utilizzo di CustomerBean è presente nel controller **EditCustomerPageController**. In particolare, nel metodo presentato di seguito, viene usato sia il Bean relativo alla sessione dell'utente loggato, sia il CustomerBean relativo al customer selezionato. La “conversazione” viene aperta nella pagina precedente (**CustomersPage**) al momento della selezione del customer.

```
// Metodo initEditCustomerPage in EditCustomerPageController.
@PostConstruct
protected void initEditCustomerPage(){

    if(!userSession.getUser().isOperator())
        throw new IllegalArgumentException("you cant view this page");

    if(conversationBean.getCustomer() == null) {
        conversationBean.setCustomer(ModelFactory.generateUser());
        conversationBean.getCustomer().addRole(ModelFactory
            .generateCustomer());
        conversationBean.getCustomer().setAgency(userSession
            .getUser().getAgency());
    }

    if (!conversationBean.getCustomer().getAgency())

```

```
        .equals(userSession.getUser().getAgency()))
    throw new IllegalStateException("customer not Editable");
}
```

Da notare l'utilizzo dell'annotazione `@PostConstruct` la quale assicura che il metodo venga eseguito subito dopo l'injection dei due Bean.

Capitolo 4

Test

Sebbene il testing sia stato presentato come momento finale nella progettazione, è stato eseguito iterativamente durante tutta l'implementazione del software, in modo da assicurare che ogni nuova classe Java inserita si comportasse come atteso.

In tutti i test eseguiti è stato verificato il corretto funzionamento dei metodi appartenenti alle varie classi. In particolare sono stati testati i Controller, i DAO e le classi del modello, attraverso i framework di **JUnit** e **Mockito**. Quest'ultimo, in combinazione con i **FieldUtils** di Apache per l'uso della Reflection, ha permesso di disaccoppiare il testing dei controller dal funzionamento dei DAO. In questo modo viene garantita la proprietà di test d'unità dei vari package.

Nonostante i test non coprano ogni singolo metodo presente nelle classi (sono stati esclusi ad esempio i metodi getter e setter), è stata ottenuta una copertura del codice scritto pari all'82% delle istruzioni. Grazie al programma **JaCoCo** è stato possibile avere una misura quantitativa della coverage, come si può vedere in Figura 4.1.

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
it.unifi.ing.swam.model	84%	100%	42	205	56	341	42	190	0	21		
it.unifi.ing.swam.bean.startup	0%	n/a	2	2	48	48	2	2	1	1		
it.unifi.ing.swam.controller.strategy	66%	58%	22	49	32	104	7	25	0	4		
it.unifi.ing.swam.dao	92%	92%	17	103	11	213	11	65	1	11		
it.unifi.ing.swam.controller	92%	94%	17	110	23	210	12	68	0	14		
it.unifi.ing.swam.bean	82%	57%	6	27	8	39	2	20	0	3		
it.unifi.ing.swam.bean.producer	0%	n/a	2	2	6	6	2	2	1	1		
Total	682 of 3.866	82%	37 of 252	85%	108	498	184	961	78	372	3	55

Figura 4.1: Report della coverage

Ovviamente il software considera anche le istruzioni e le classi che abbiamo scelto di non testare. Per una migliore gestione delle eccezioni in fase di test è stata usata la libreria **AssertJ** che permette di trattare le eccezioni in maniera analoga ai normali output con **Assert**. Di seguito un esempio di test

relativo al controller della pagina di ricerca delle Waybill in cui è possibile notare anche l'utilizzo di Mockito.

```
// Metodi presenti in SearchPageControllerTest.

@Test
public void testSetOperator() {
    when(userDao.findById(operatorId)).thenReturn(operator);
    searchPageController.initSearchPage();
    searchPageController.setOperator(operatorId);
    assertEquals(operator,
        searchPageController.getWaybillQuery().getOperator());
}

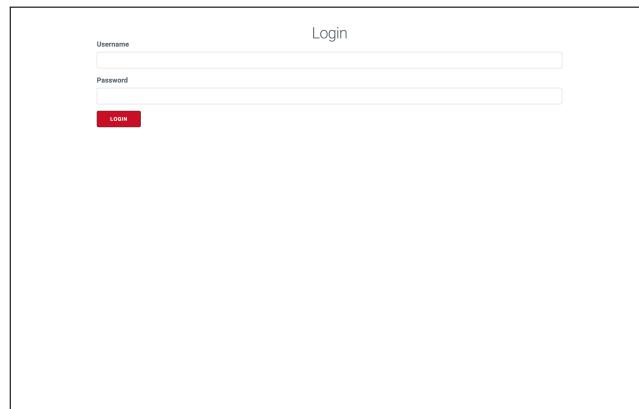
@Test
public void testSetOperatorThrowsIllegalArgumentException() {
    when(userDao.findById(operatorId)).thenReturn(customer);
    searchPageController.initSearchPage();

    assertThatExceptionOfType(IllegalArgumentException.class)
        .isThrownBy(() -> {
            searchPageController.setOperator(operatorId);
        });
}
```

Capitolo 5

Interfaccia Web

Durante lo svolgimento del lavoro, per valutare il corretto funzionamento dei Controller e dei DAO, sono state sviluppate alcune pagine JSF relative a un’ipotetica interfaccia web. Nelle Figure 5.1, 5.2, 5.3 e 5.4 sono riportate le due pagine progettate.



The image shows a clean, minimalist login interface. At the top center, the word "Login" is written in a small, sans-serif font. Below it are two horizontal input fields. The first field is labeled "Username" and contains the placeholder text "Username". The second field is labeled "Password" and also contains the placeholder text "Password". At the bottom of the form is a single, solid red rectangular button with the word "LOGIN" written in white capital letters.

Figura 5.1: Pagina di Login

5.1 Home Page

L’interfaccia per la home page è mostrata nelle Figure 5.2, 5.3 e 5.4.

Considerando che un utente può assumere fino a 3 ruoli diversi, la home page è strutturata in modo da contenere 3 sezioni, una per ciascun ruolo. Questa pagina è l’unica in cui un utente può assumere 3 ruoli in contemporanea. Infatti nelle pagine per la gestione delle waybill, l’utente assume il ruolo indicato dalla sezione della home page nella quale ha selezionato la waybill.

Welcome all!

Operator Home Page

To be validated:

ID	Sender Name	Departure Agency Id	Receiver Name	Receiver Address	Destination Agency Id	Cost	Tracking
24	mario	2	giorgio	via gorgini arezzo, italia 52100	22	10.0	IDLE

Customer Home Page

Operator Home Page

Figura 5.2: Homepage dell'Operator

Welcome all!

Operator Home Page

Customer Home Page

Proposed:

ID	Sender Name	Departure Agency Id	Receiver Name	Receiver Address	Destination Agency Id	Cost	Tracking
24	mario	2	giorgio	via gorgini arezzo, italia 52100	22	10.0	IDLE

Customer Home Page

Figura 5.3: Homepage del Customer

Welcome all!

Operator Home Page

Customer Home Page

Driver Home Page

Today's mission:

ID	Sender Name	Departure Agency Id	Receiver Name	Receiver Address	Destination Agency Id	Cost	Tracking
----	-------------	---------------------	---------------	------------------	-----------------------	------	----------

Figura 5.4: Homepage del Driver

Capitolo 6

Considerazioni finali e sviluppi futuri

In questo elaborato, è stata modellata, sviluppata e testata una Web Application per la gestione di spedizioni di un'azienda di logistica.

Particolare attenzione è stata posta nella realizzazione dei vari controlli, che sono spesso ridondanti su diversi livelli (ad esempio sui Controller e sulle pagine JSF). In questo modo, pur mancando gran parte l'interfaccia web, il sistema sottostante dovrebbe avere un comportamento prevedibile in ogni situazione.

Il naturale sviluppo futuro del progetto è il completamento del layer di interfaccia e una fase di usability test. In questo modo sarà possibile verificare come un vero utente interagisce con il sistema.

Appendice A

Mockup

In questa Appendice sono presenti tutti i mockup elaborati.

Operator Homepage

Welcome, OP356357 

Waybills to Validate

ID	Customer	Cost	Weight	Volume	# items	Actions
AG567RB3	Prada	500 €	40 kg	7	22	 
AG4357H52	Company	60 €	10 kg	2	7	 
AH8372PQE	Just Company	450 €	200 kg	89	45	 

Waybills to Assign

ID	Customer	Cost	Weight	Volume	# items	Actions
AG567RB3	Prada	270 €	10 kg	2	10	 
AG567RB3	Prada	43 €	5 kg	8	3	 
AGHTW28	Company	420 €	145 kg	76	102	 
AGTY342L	ARC	50 €	13 kg	18	11	 

Buttons:

- Search Waybills
- Create Waybill
- Customers
- Assign Driver

Figura A.1: Operator Homepage

Customer Homepage																																															
Welcome, CU567RB3 ➔																																															
Waybill Validated																																															
<table border="1"> <thead> <tr> <th>ID</th> <th>Receiver</th> <th>Cost</th> <th>Weight</th> <th>Volume</th> <th># items</th> <th>Actions</th> <th>⋮</th> </tr> </thead> <tbody> <tr> <td>AG567RB3</td> <td>CU4HDT52</td> <td>500 €</td> <td>40 kg</td> <td>7</td> <td>22</td> <td></td> <td></td> </tr> <tr> <td>AG4357HS2</td> <td>CU34543</td> <td>60 €</td> <td>10 kg</td> <td>2</td> <td>7</td> <td></td> <td></td> </tr> <tr> <td>AH8372P9E</td> <td>CU576345</td> <td>950 €</td> <td>200 kg</td> <td>89</td> <td>45</td> <td></td> <td></td> </tr> </tbody> </table>								ID	Receiver	Cost	Weight	Volume	# items	Actions	⋮	AG567RB3	CU4HDT52	500 €	40 kg	7	22			AG4357HS2	CU34543	60 €	10 kg	2	7			AH8372P9E	CU576345	950 €	200 kg	89	45										
ID	Receiver	Cost	Weight	Volume	# items	Actions	⋮																																								
AG567RB3	CU4HDT52	500 €	40 kg	7	22																																										
AG4357HS2	CU34543	60 €	10 kg	2	7																																										
AH8372P9E	CU576345	950 €	200 kg	89	45																																										
<input type="button" value="Search Waybills"/> <input type="button" value="Create Waybill"/>																																															
Waybill Proposed																																															
<table border="1"> <thead> <tr> <th>ID</th> <th>Receiver</th> <th>Cost</th> <th>Weight</th> <th>Volume</th> <th># items</th> <th>Actions</th> <th>⋮</th> </tr> </thead> <tbody> <tr> <td>AG567RB3</td> <td>CU576345</td> <td>500 €</td> <td>10 kg</td> <td>2</td> <td>10</td> <td></td> <td></td> </tr> <tr> <td>AG567RB3</td> <td>CU576345</td> <td>500 €</td> <td>10 kg</td> <td>2</td> <td>10</td> <td></td> <td></td> </tr> <tr> <td>AG567RB3</td> <td>CU576345</td> <td>500 €</td> <td>10 kg</td> <td>2</td> <td>10</td> <td></td> <td></td> </tr> <tr> <td>AG567RB3</td> <td>CU576345</td> <td>500 €</td> <td>10 kg</td> <td>2</td> <td>10</td> <td></td> <td></td> </tr> </tbody> </table>								ID	Receiver	Cost	Weight	Volume	# items	Actions	⋮	AG567RB3	CU576345	500 €	10 kg	2	10			AG567RB3	CU576345	500 €	10 kg	2	10			AG567RB3	CU576345	500 €	10 kg	2	10			AG567RB3	CU576345	500 €	10 kg	2	10		
ID	Receiver	Cost	Weight	Volume	# items	Actions	⋮																																								
AG567RB3	CU576345	500 €	10 kg	2	10																																										
AG567RB3	CU576345	500 €	10 kg	2	10																																										
AG567RB3	CU576345	500 €	10 kg	2	10																																										
AG567RB3	CU576345	500 €	10 kg	2	10																																										

Figura A.2: Customer Homepage

Driver Homepage																																																																															
Welcome, DR435352 ➔																																																																															
Waybills to Deliver Today																																																																															
<table border="1"> <thead> <tr> <th>ID</th> <th>Receiver</th> <th>Cost</th> <th>Weight</th> <th>Volume</th> <th># items</th> <th>Status</th> <th>Actions</th> <th>⋮</th> </tr> </thead> <tbody> <tr> <td>AG567RB3</td> <td>Prada</td> <td>500 €</td> <td>40 kg</td> <td>7</td> <td>22</td> <td>Delivered</td> <td></td> <td></td> </tr> <tr> <td>AG4357HS2</td> <td>Company</td> <td>60 €</td> <td>10 kg</td> <td>2</td> <td>7</td> <td>Delivered</td> <td></td> <td></td> </tr> <tr> <td>AH8372P9E</td> <td>Just Company</td> <td>950 €</td> <td>200 kg</td> <td>89</td> <td>45</td> <td>Delivering</td> <td></td> <td></td> </tr> <tr> <td>AG567RB3</td> <td>Prada</td> <td>270 €</td> <td>10 kg</td> <td>2</td> <td>10</td> <td>Delivering</td> <td></td> <td></td> </tr> <tr> <td>AG567RB3</td> <td>Prada</td> <td>43 €</td> <td>5 kg</td> <td>8</td> <td>3</td> <td>Delivering</td> <td></td> <td></td> </tr> <tr> <td>AGHTW28</td> <td>Company</td> <td>420 €</td> <td>145 kg</td> <td>76</td> <td>102</td> <td>Delivering</td> <td></td> <td></td> </tr> <tr> <td>AGTY392L</td> <td>ARC</td> <td>50 €</td> <td>13 kg</td> <td>18</td> <td>11</td> <td>Delivering</td> <td></td> <td></td> </tr> </tbody> </table>								ID	Receiver	Cost	Weight	Volume	# items	Status	Actions	⋮	AG567RB3	Prada	500 €	40 kg	7	22	Delivered			AG4357HS2	Company	60 €	10 kg	2	7	Delivered			AH8372P9E	Just Company	950 €	200 kg	89	45	Delivering			AG567RB3	Prada	270 €	10 kg	2	10	Delivering			AG567RB3	Prada	43 €	5 kg	8	3	Delivering			AGHTW28	Company	420 €	145 kg	76	102	Delivering			AGTY392L	ARC	50 €	13 kg	18	11	Delivering		
ID	Receiver	Cost	Weight	Volume	# items	Status	Actions	⋮																																																																							
AG567RB3	Prada	500 €	40 kg	7	22	Delivered																																																																									
AG4357HS2	Company	60 €	10 kg	2	7	Delivered																																																																									
AH8372P9E	Just Company	950 €	200 kg	89	45	Delivering																																																																									
AG567RB3	Prada	270 €	10 kg	2	10	Delivering																																																																									
AG567RB3	Prada	43 €	5 kg	8	3	Delivering																																																																									
AGHTW28	Company	420 €	145 kg	76	102	Delivering																																																																									
AGTY392L	ARC	50 €	13 kg	18	11	Delivering																																																																									

Figura A.3: Driver Homepage

Assign Driver Page																																															
Welcome, OP356357 ➔																																															
Drivers Available																																															
<table border="1"> <thead> <tr> <th>ID</th> <th>Driver</th> <th>Truck Weight</th> <th>Truck Volume</th> <th>Type</th> <th>Actions</th> <th>⋮</th> </tr> </thead> <tbody> <tr> <td>DR212RB3</td> <td>Mario</td> <td>500 kg</td> <td>300</td> <td>VAN</td> <td></td> <td></td> </tr> <tr> <td>DR456HS2</td> <td>Luigi</td> <td>1000 kg</td> <td>500</td> <td>VAN</td> <td></td> <td></td> </tr> <tr> <td>DR835462P9E</td> <td>Piero</td> <td>2500 kg</td> <td>900</td> <td>TRUCK</td> <td></td> <td></td> </tr> </tbody> </table>								ID	Driver	Truck Weight	Truck Volume	Type	Actions	⋮	DR212RB3	Mario	500 kg	300	VAN			DR456HS2	Luigi	1000 kg	500	VAN			DR835462P9E	Piero	2500 kg	900	TRUCK														
ID	Driver	Truck Weight	Truck Volume	Type	Actions	⋮																																									
DR212RB3	Mario	500 kg	300	VAN																																											
DR456HS2	Luigi	1000 kg	500	VAN																																											
DR835462P9E	Piero	2500 kg	900	TRUCK																																											
Waybills to Assign																																															
<table border="1"> <thead> <tr> <th>ID</th> <th>Customer</th> <th>Cost</th> <th>Weight</th> <th>Volume</th> <th># items</th> <th>Actions</th> <th>⋮</th> </tr> </thead> <tbody> <tr> <td>AG567RB3</td> <td>Prada</td> <td>270 €</td> <td>10 kg</td> <td>2</td> <td>10</td> <td></td> <td></td> </tr> <tr> <td>AG567RB3</td> <td>Prada</td> <td>43 €</td> <td>5 kg</td> <td>8</td> <td>3</td> <td></td> <td></td> </tr> <tr> <td>AGHTW28</td> <td>Company</td> <td>420 €</td> <td>145 kg</td> <td>76</td> <td>102</td> <td></td> <td></td> </tr> <tr> <td>AGTY392L</td> <td>ARC</td> <td>50 €</td> <td>13 kg</td> <td>18</td> <td>11</td> <td></td> <td></td> </tr> </tbody> </table>								ID	Customer	Cost	Weight	Volume	# items	Actions	⋮	AG567RB3	Prada	270 €	10 kg	2	10			AG567RB3	Prada	43 €	5 kg	8	3			AGHTW28	Company	420 €	145 kg	76	102			AGTY392L	ARC	50 €	13 kg	18	11		
ID	Customer	Cost	Weight	Volume	# items	Actions	⋮																																								
AG567RB3	Prada	270 €	10 kg	2	10																																										
AG567RB3	Prada	43 €	5 kg	8	3																																										
AGHTW28	Company	420 €	145 kg	76	102																																										
AGTY392L	ARC	50 €	13 kg	18	11																																										
<input type="button" value="Back"/>																																															

Figura A.4: Assign Driver Page

Mission Assign Page

Welcome, OP356357

Mission for Driver: Mario

Available Weight: 173 / 500 Kg	Available Volume: 104 / 300 m3
--------------------------------	--------------------------------

Id	Customer	Cost	Weight	Volume	# items	Actions
AG567RB3	Prado	270 €	10 kg	2	10	
AG567RB3	Prado	43 €	5 kg	8	3	
AGHTW28	Company	420 €	145 kg	76	102	
AGTY392L	ARC	50 €	13 kg	18	11	

Back **Add Waybills**

Figura A.5: Mission Assign Page

Search Waybill

Welcome, OP356357

Extended Search

Search by ID:	Departure Date	Arrived Date
<input type="text"/>	<input type="text"/> / /	<input type="text"/> / /
Search by Customer:	Sort by:	
<input type="text"/>	State	Cost
Search by Departure Agency:	Search by Arrived Agency:	Search

Id	Customer	Cost	Weight	Volume	# items	Actions
AG567RB3	Prado	270 €	10 kg	2	10	
AG567RB3	Prado	43 €	5 kg	8	3	
AGHTW28	Company	420 €	145 kg	76	102	
AGTY392L	ARC	50 €	13 kg	18	11	

Back

Figura A.6: Search Waybill Page

Modify and View Waybill

Welcome, OP356357

Waybill ID: #123456789 State: Validated

FROM: Mario Rossi Via Bianchi 232 50063 Figline Valdarno (FI) Customer ID: 34565465673	TO: Riccardo Bianchi Via Giorgini 45 50045 Montevarchi (AR)	Create By: OP23432355 Creation Date: 1/04/2017
-------------------------------------------------------------------------------------------------	-------------------------------------------------------------------	---------------------------------------------------

Total Weight: 2kg	Cost: 50,63€	
Total Volume: 0,3	Fore: IT858364	
Packages: 3		

Departure Date: 23/12/2012	Departure Agency: Firenze
Arrival Date: TDB	Arrival Agency: Arezzo
Driver Assigned: DR649362	

Tracking: IDLE	Signature Receiver:
----------------	---------------------

Back **Modify**

Figura A.7: Modify and View Waybill

Figura A.8: Customer List

Customer Page

Welcome, OP356357 

Customer ID: #69574846364 State: Active 

Company Name: Prada S.P.A.	Create By: OP23432355
Address: Via de Levone 50034 Terranuova Bracciolini (AR) Italy	Creation Date: 1/04/2017
Phone: +34 0657483736	

Fares List:

Identifier	Zone	Cost	From	To	Action
IT5752342	Italy	23	23/12/2014	23/12/2019	              
EU234243	Europe	131	21/7/2014	21/7/2016	              
ME756756	Mexico	300	14/5/2014	7/4/2018	              

Add new Fare 

Back **Modify**

Figura A.9: Customer Page

Modify and View Fare

Welcome, OP356357 

Fare ID: #IT065679 State: Active 

Customer ID: CUR3474382
Customer Name: Prado S.P.A.

Zone: Italy

Valid From: 23/4/2016
Valid To: 8/3/2018

Cost: 54,4€

Create By: OP23432355
Creation Date: 1/04/2017

Back **Modify**

Figura A.10: Modify and View Fare