

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

21 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	8
3.3.1	Mockups	12
3.4	Domain Model	12
3.5	Architettura Implementativa	14
3.5.1	Domain Logic	14
3.5.2	Domain Model	15
3.5.3	DAO	16
3.5.4	Business Logic	17
4	Test	24
5	Interfaccia Web	26
5.1	Home Page	26
5.2	Pagine di gestione della Waybill	27
5.3	Pagine di gestione dei Customer	29
6	Considerazioni finali e sviluppi futuri	31
A	Mockup	32

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 (**waybill**) da parte degli operatori del corriere (**operator**), dei guidatori dei camion (**driver**) e dei clienti (**customer**). 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 e 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 assegnate.

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 informazioni.

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. È richiesta inoltre una parte di interfaccia grafica che permetta quantomeno di testare le principali annotazioni CDI presenti.

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.

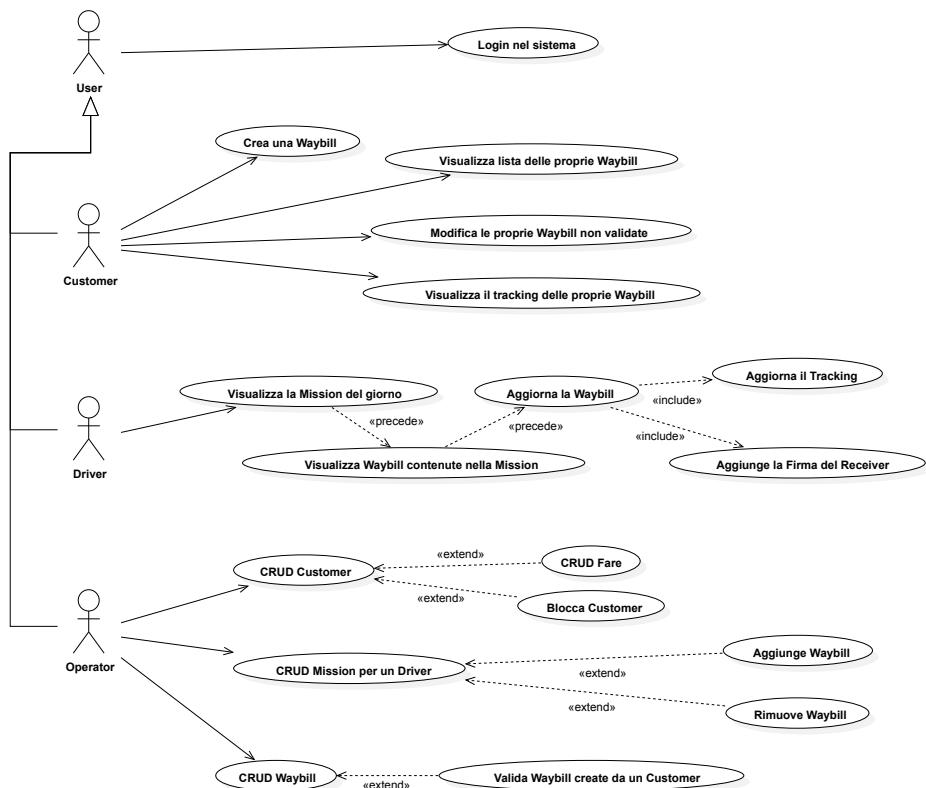


Figura 3.1: Use Case Diagram

Sono stati individuati 3 attori all'interno del sistema: **Operator**, **Customer** e **Driver**, che derivano dal ruolo generico **User**. Non possono esistere utenti senza ruolo.

Data la natura enterprise dell'applicativo, non è presente un caso d'uso di registrazione dei nuovi User. Gli utenti Operator e Driver sono inseriti dall'amministratore del sistema, mentre gli utenti Customer possono essere inseriti anche dagli Operator.

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.

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, nonché 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. Dove si sono omesse le frecce per il tasto back e quello home (presenti in ogni pagina).

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

- Un operator visualizzerà tutte le waybill da validare relative alla sua agency e tutte quelle da assegnare ad un driver, sempre relative alla sua agency.
- Un customer visualizzerà tutte le waybill proposte, ancora modificabili, e tutte quelle già validate, per controllarne il tracking.
- Un driver visualizzerà tutte le waybill presenti nella mission di quel giorno, e potrà mettere la firma e aggiornare il tracking di quelle non ancora consegnate.

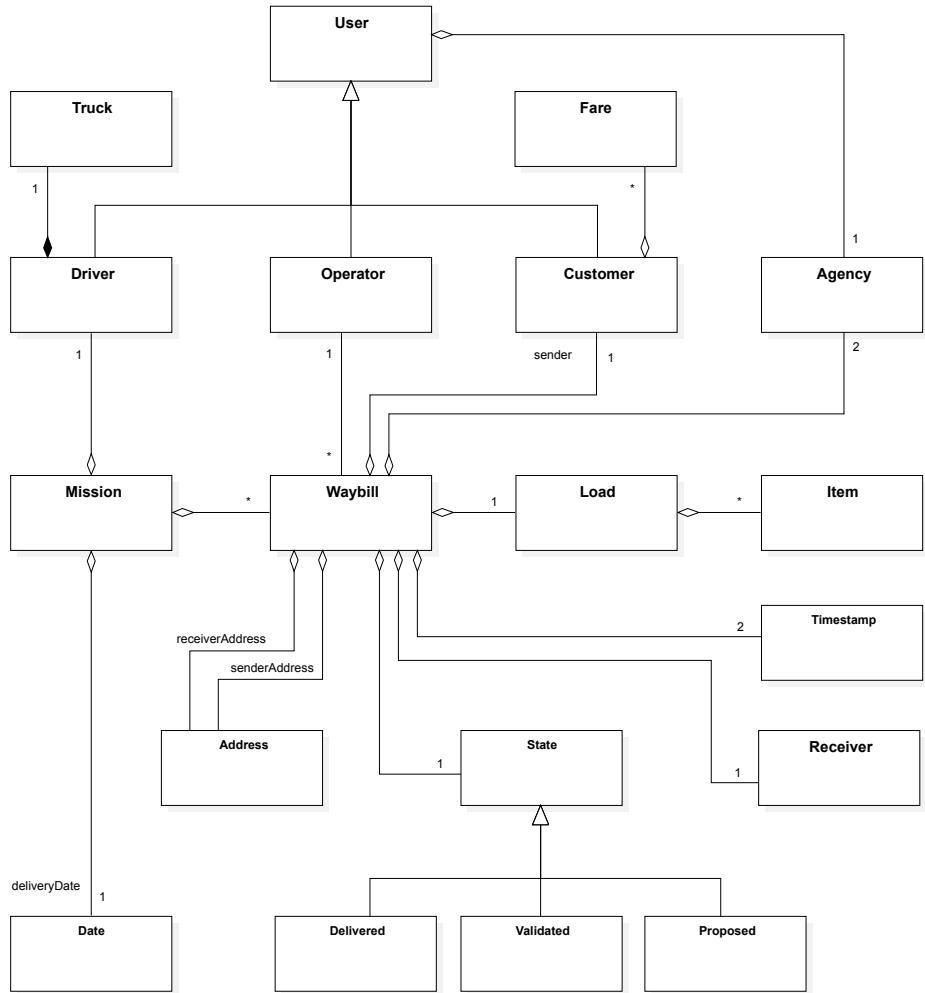


Figura 3.2: Conceptual Model

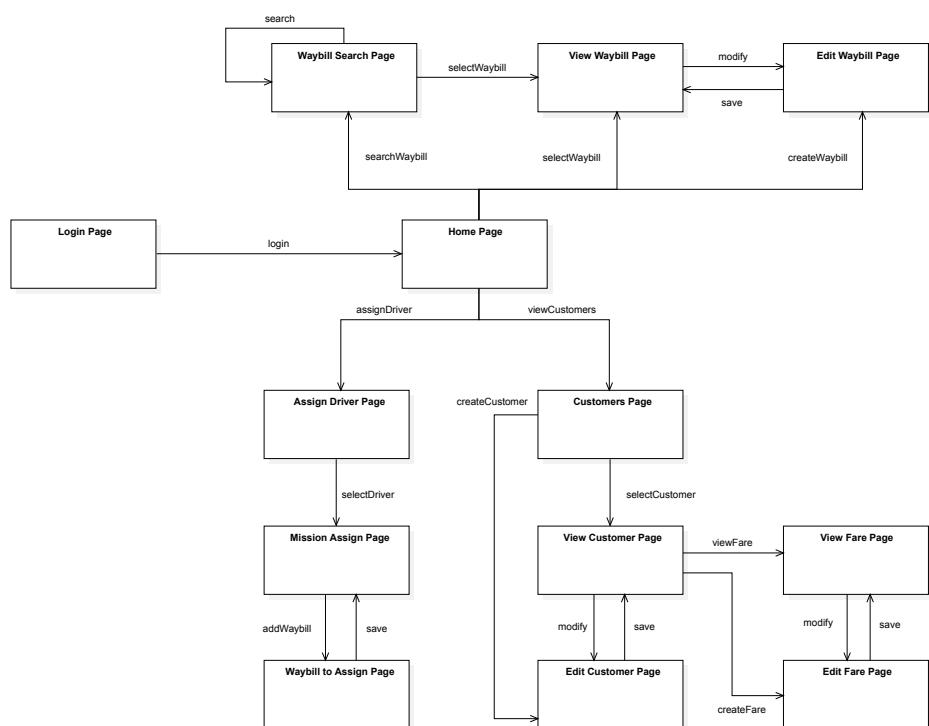


Figura 3.3: Page Navigation Diagram

- 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 che nel diagramma sono sotto la Home, per la **gestione della mission** dei driver e per la **gestione dei clienti** sono accessibili soltanto dall'**Operator**.

Per quanto riguarda le altre pagine:

- La **Assign Driver Page** contiene sia i driver disponibili quel giorno, che la lista di waybill non ancora assegnate.
- La **Mission Assign Page** contiene tutte le waybill della mission odier- na del driver selezionato nella Assign Driver Page, con i pulsanti per l'aggiunta di una nuova waybill.
- La **Waybill to assign Page** contiene tutte le waybill non assegnate che hanno peso e volumi che possono entrare nella mission corrente, dati peso e volume rimanenti nel truck del driver.
- La **Customers Page** contiene la lista di tutti i customer di un opera- tor, con i tasti di visualizzazione, di modifica e di aggiunta di un nuovo customer.
- Le pagine **View Customer Page** e **Edit Customer Page** permet- tono la visualizzazione e l'editing delle informazioni relative ad un customer.
- Le pagine **View Fare Page** e **Edit Customer Page** consentono di vedere e modificare le informazioni relative ad una Fare (tariffa) di un customer.
- Le pagine **View Waybill Page** e **Edit Waybill Page** si occupano della visualizzazione e modifica di una waybill, sono accessibili da tutti i ruoli ma con comportamenti radicalmente diversi.
- La pagina **Waybill Search Page** contiene un form di ricerca avanzata per le waybill, è accessibile solo dagli operator e dai customer ed ha anch'essa funzioni diverse a seconda del ruolo.

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	
A056789B3	Prods	43 €	5 kg	8	3	
A0HTWZB	Company	420 €	145 kg	76	102	
AU7Y3NL	ARC	50 €	13 kg	18	11	

(b) View/Edit Waybill page

Waybill ID: #123456789		Welcome, OP356357	
FROM: Mario Rossi Via Bracciali 232 50045 Firenze (FI) Customer ID: 345654456573	TO: Riccardo Bianchi Via Giorgio 45 50045 Montecatini (AR)	Create By: OP356357 Creation Date: 1/04/2017	
Total Weight: 19 kg Total Volume: 104 m³ Packages: 3	Cost: 50,63€ Fare: 118,68€		
Departure Date: 12/12/2012 Arrival Date: TBC Driver Assigned: DR644362	Departure Agency: Firenze Arrival Agency: Arezzo		
Tracking: IDLE			
		Signature Receiver:	Book Modify

(c) Search Waybill page

Search Waybill						
Extended Search		Departure Date	Arrived Date			
Search by ID:	<input type="text"/>	<input type="button"/>	<input type="button"/>			
Search by Customer:	<input type="button"/>	<input type="button"/>	<input type="button"/>	Sort by:	Cost	
Search by Departure Agency:	<input type="button"/>	<input type="button"/>	<input type="button"/>			
Search						
ID	Customer	Cost	Weight	Volume	# Items	Actions
A056789B3	Prods	120 €	10 kg	2	10	
A056789B3	Prods	43 €	5 kg	8	3	
A0HTWZB	Company	420 €	145 kg	76	102	
AU7Y3NL	ARC	50 €	13 kg	18	11	

(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	
A0HTWZB	Company	420 €	145 kg	76	102	
AU7Y3NL	ARC	50 €	13 kg	18	11	

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.

12

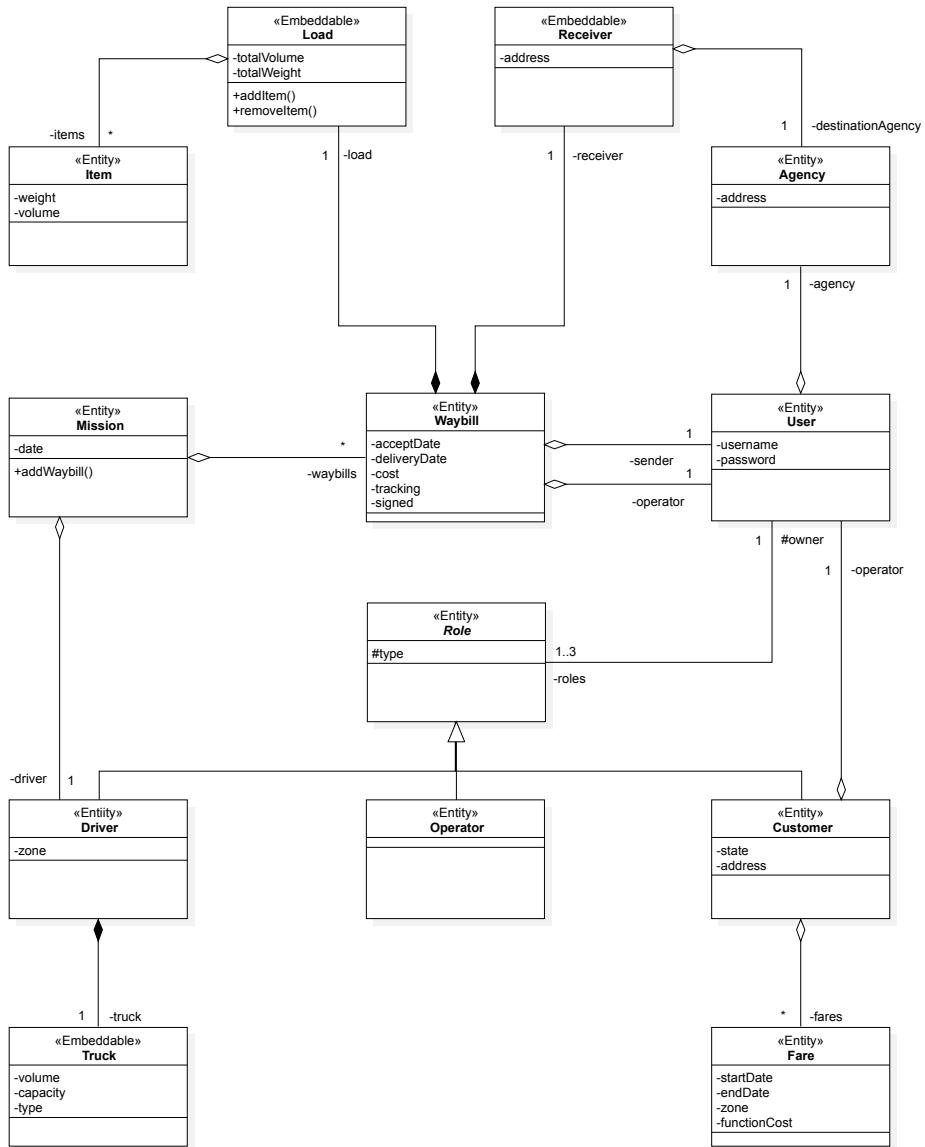


Figura 3.5: Domain Model

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**) inserendo una relazione di composizione: ogni **User** può contenere fino a 3 ruoli. In questo modo è stato possibile creare **utenti multiruolo**, caso particolare che ha introdotto una sfida a livello di implementazione.

I constraints 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 type dentro Role (enum con valori CUSTOMER, DRIVER e OPERATOR) rappresenta il **PartyType**. L'**AccountabilityType** non è stato realizzato, in quanto non deve essere possibile fare nuovi tipi di 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, semplificando il modello.

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 ed implementate 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.le funzioni dei controller operano sulle entità e sono alla base dei casi d'uso.

3.5.2 Domain Model

L'implementazione del modello di dominio è stata fatta seguendo l'uml prodotto, dove le relazioni di aggregazione con il diamante vuoto e la molteplicità a molti dall'altra parte si sono tradotte nell'annotazione `@OneToMany`. Ad esempio, l'aggregazione tra Mission e Waybill è annotata come `@OneToMany` nella classe Mission (una mission può avere più waybill). L'associazione tra User e Role è stata annotata con `@OneToMany` nella classe User e `@ManyToOne` in quella Role. In aggiunta, 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.

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 **sender** e **receiver** della classe Waybill.

Un'altra notazione usata in Role è `@Transient`. Questa è stata applicata all'attributo **type**, che non viene salvato sul database poichè l'oggetto type riflette il tipo sottoclasse di Role dell'oggetto che lo contiene.

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. Questo è stato fatto per evitare di avere un'unica tabella con molti attributi nulli e permettere l'utilizzo di query polimorfiche efficienti.

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 di accountability sul ruolo dell'utente. 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 chiamata **BaseDao**, successivamente estesa da tutti i DAO, che implementa i metodi `save`, `delete` e `findById`, e contiene il riferimento all'**Entity Manager**. Per farlo, questa classe sfrutta i generics di Java, dove il tipo generico rappresenta la classe del dominio corrispondente ad un particolare DAO. Per ogni entità è stato quindi implementato un suo particolare DAO:

```
// Inizio della dichiarazione di BaseDao
public abstract class BaseDao<E extends BaseEntity> implements
    Serializable

// Inizio della dichiarazione di WaybillDao.
public class WaybillDao extends BaseDao<Waybill>
```

I metodi sono stati implementati a seconda delle necessità dei controller della Business Logic. Degna di nota è la funzione di ricerca avanzata su **WaybillDao**: `advancedSearch`. 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> findProposedByAgency(Agency agency) {
    return entityManager
        .createQuery("SELECT w FROM Waybill w WHERE
            w.sender.agency = :agency AND w.operator IS NULL",
            Waybill.class).setParameter("agency", agency)
        .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. Una piccola parte dei controller è annotata come `@Model`, ovvero vengono distrutti non appena la pagina JSF viene generata. Tuttavia, i controller che devono modificare e persistere entità sul database utilizzano l'annotazione `@ViewScoped`, che li tiene in vita fintanto che la pagina web rimane attiva. Anche quelli che contengono parametri http utilizzano la notazione `@ViewScoped`, aggiunta per far funzionare i tasti di logout e home. Per queste pagine avere uno scope di tipo Request (come con `@Model`) non è sufficiente, poiché vengono ricaricate ogni volta che l'utente preme su home o logout, perdendo i parametri http.

Beans

Per contenere informazioni che durano più pagine abbiamo fatto 3 Bean: **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 **Truck** associato.

HomePageController

Il controller della Home Page implementa tutte le funzioni dei 3 ruoli. Ciascuna recupera una lista di Waybill attraverso i DAO e, dopo aver effettuato il controllo sul ruolo, la rende disponibile alla pagina web con un metodo getter. Un utente nella Home assume tutti i suoi ruoli: Un utente che è sia un Driver, sia un Customer, sia un Operator visualizzerà 3 sezioni e userà tutti i metodi dell'**HomePageController**.

Di seguito è riportato il metodo che fa vedere tutte le waybill proposte a un operatore:

```
// Metodo presente in HomePageController.
public List<Waybill> getProposedWaybillOperator() {
    if (getUser().isOperator())
        return waybillDao.findProposedByAgency(getUser().getAgency());
    else
        return null;
}
```

ViewWaybillPageController, EditWaybillPageController e il Pattern Strategy

Questi controller a differenza di quello dell'home, hanno il ruolo dettato dalla pagina precedente: un utente multiruolo che accede ad una waybill dalla sezione operator avrà un accesso alla waybill come operator, e così per gli altri due ruoli.

Poichè le pagine di visualizzazione e modifica della Waybill cambiano comportamento a seconda del ruolo dell'utente, per l'implementazione dei controller si è deciso di adottare il pattern Strategy, come mostrato in Figura 3.6. Così facendo possiamo implementare i controlli di accesso rendendo i controller unaware del ruolo dell'utente. Un controllo sul ruolo è 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, settano la variabile `CurrentRole` e inizializzano l'oggetto `strategy` adeguato, con una delle 3 classi concrete figlie di `RoleStrategy`. Queste definiscono alcuni

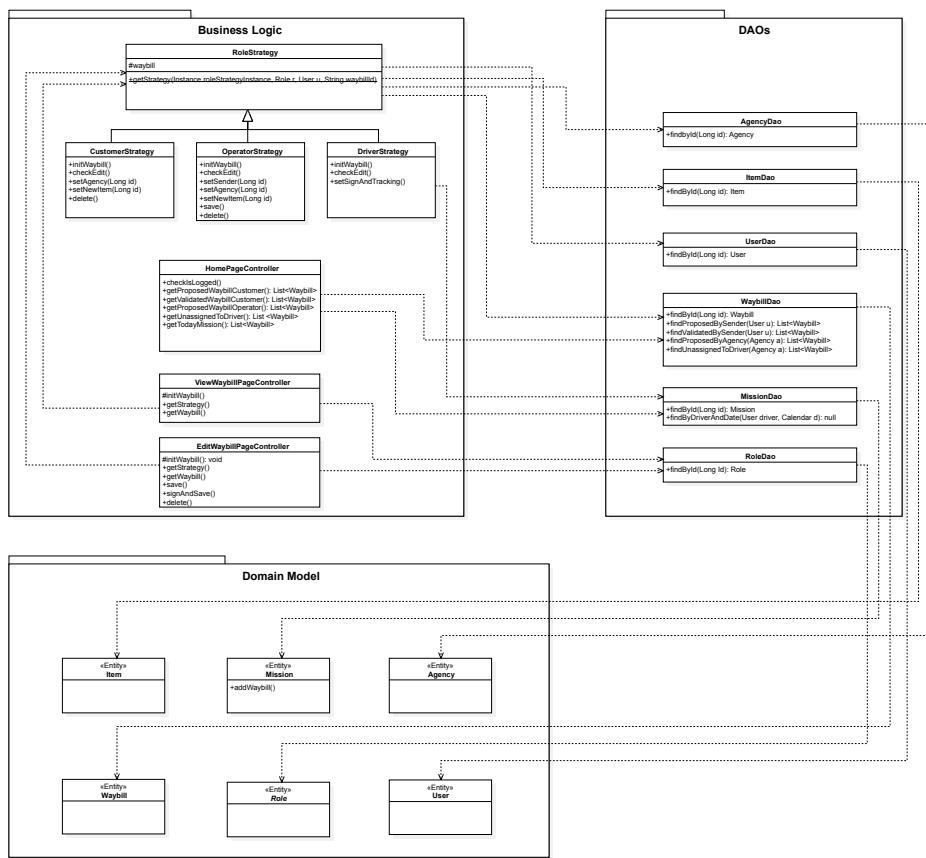


Figura 3.6: 3-Tier Model per le pagine ViewWaybill, EditWaybill e Home

metodi setter e getter che operano sulla waybill. In particolare quelli che collegano la waybill ad altre entità. I setter sono necessari per effettuare ul controllo sul database sulle entità da collegare alla waybill.

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

Gestendo le sottoclassi di RoleStrategy come Bean di CDI è stato possibile spostare l'injection dei dao, dai controller agli strategy. Tutte le Annotazioni CDI sono quindi utilizzabili sugli strategy concreti.

Per farlo è stata fatta l'injection a runtime dello strategy concreto. Nei controller è presente infatti l'oggetto **roleStrategyInstance** che contiene i bean per **OperatorStrategy** **CustomerStrategy** e **DriverStrategy**. Questo viene passato alla funzione **getStrategy** di RoleStrategy per la scelta.

```
// Membro di ViewWaybillPageController e EditWaybillPageController.
@Inject Instance<RoleStrategy> roleStrategyInstance;

// Metodo presente in RoleStrategy
public static RoleStrategy getStrategy(Instance<RoleStrategy>
    roleStrategyInstance, Role r, User u, String waybillId) {
    RoleStrategy rs;
    if (r.isCustomer())
        rs = roleStrategyInstance.select(CustomerStrategy.class).get();
    else if (r.isOperator())
        rs = roleStrategyInstance.select(OperatorStrategy.class).get();
    else if (r.isDriver())
        rs = roleStrategyInstance.select(DriverStrategy.class).get();
    else
        throw new IllegalArgumentException("role not found");
    rs.init(u, waybillId);
    return rs;
}
```

Di seguito, alcuni metodi rilevanti:

```
// Metodo setSender in OperatorStrategy.
@Override
public void setSender(Long id) {
    User sender = userDao.findById(id);
    if (sender == null)
        throw new IllegalArgumentException("id not found");
```

```

    else if (!sender.isCustomer())
        throw new IllegalArgumentException("the sender is not a
            customer");
    else if (!sender.getCustomerRole().isActive())
        throw new IllegalArgumentException("the sender is blocked!");
    waybill.setSender(sender);
}

```

```

// Metodo save in EditWaybillPageController.
@Transactional
public String save() {
    strategy.save();
    return "waybill-view?id=" + getWaybill().getId() + "&roleId=" +
        currentRole.getId() + "&faces-redirect=true";
}

// Metodo save in OperatorStrategy.
@Override
public void save() {
    waybill.setOperator(user);
    waybill.setAcceptDate(Calendar.getInstance());
    super.save();
}

```

EditCustomerPageController e il CustomerBean

Un esempio di utilizzo di CustomerBean è presente nel controller **EditCustomerPageController**, come mostrato in Figura 3.7. 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");
}

```

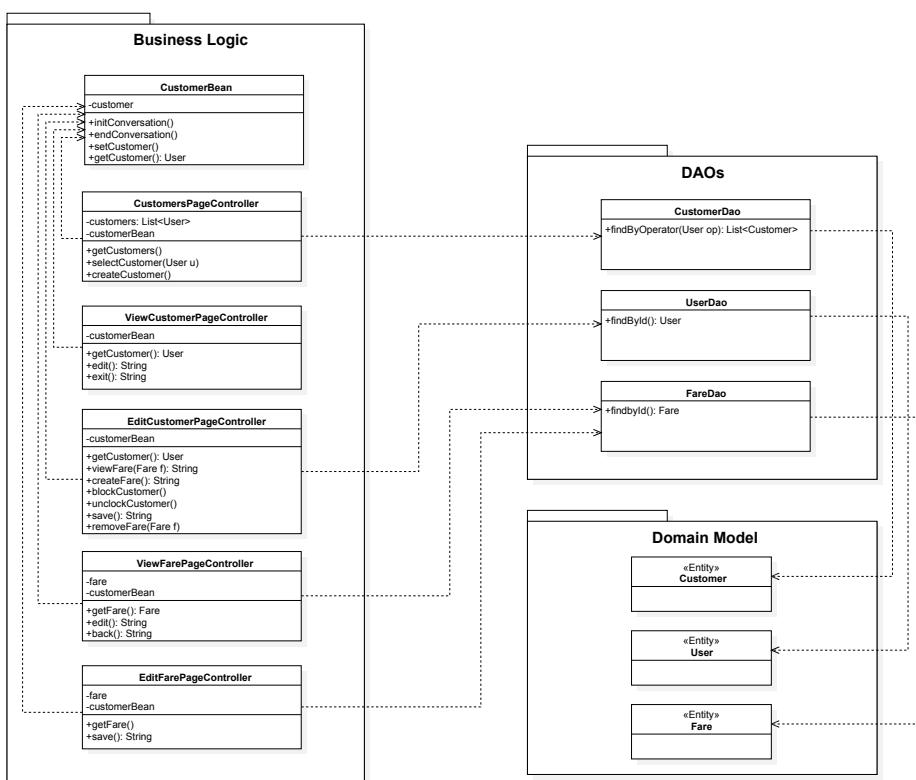


Figura 3.7: 3-Tier Model con CustomerBean

```

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

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.

Di seguito alcuni metodi degni di nota:

```

// Metodo selectCustomer in CustomersPageController.
public String selectCustomer(User u) {
    customerBean.initConversation();
    customerBean.setCustomer(u);
    return "customer-view?faces-redirect=true";
}

// Metodo removeFare in EditCustomerPageController.
@Transactional
public void removeFare(Fare f) {
    customerBean.getCustomer().getCustomerRole().getFares().remove(f);
}

```

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'89% delle istruzioni. Grazie al programma **JaCoCo** è stato possibile avere una misura quantitativa della coverage, come si può vedere in Figura 4.1.

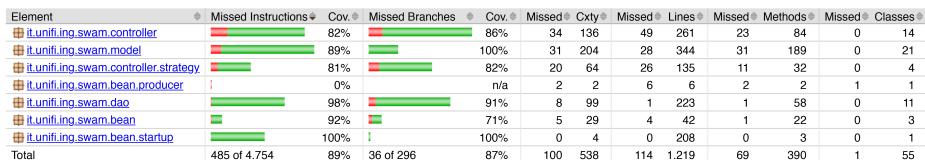


Figura 4.1: Report della coverage

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.

```
//part of init of WaybillDaoJpaTest
protected void init() throws InitializationError {
    ...
    proposedWaybill = ModelFactory.generateWaybill();
    User otherSender = ModelFactory.generateUser();
    otherSender.addRole(ModelFactory.generateCustomer());
    otherSender.setAgency(departureAgency);
    proposedWaybill.setSender(otherSender);
    ...
}

// Metodi presenti in WaybillDaoJpaTest.
@Test
public void testFindProposedByAgency() {
    List<Waybill> result =
        waybillDao.findProposedByAgency(proposedWaybill.getSender()
            .getAgency());
    assertEquals(1, result.size());
    assertEquals(proposedWaybill, result.get(0));
}

// 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 dell’interfaccia web.

Lo sviluppo dell’interfaccia ha permesso il test di tutto il sistema e, in particolare, delle annotazioni CDI, che non era stato effettuato nei test dei controller. Questo ha permesso di correggere alcuni errori (nei controller, nei dao e nel modello) molto difficili da individuare con i semplici test di unità.

Nelle Figura 5.1 è riportata la pagina di login.

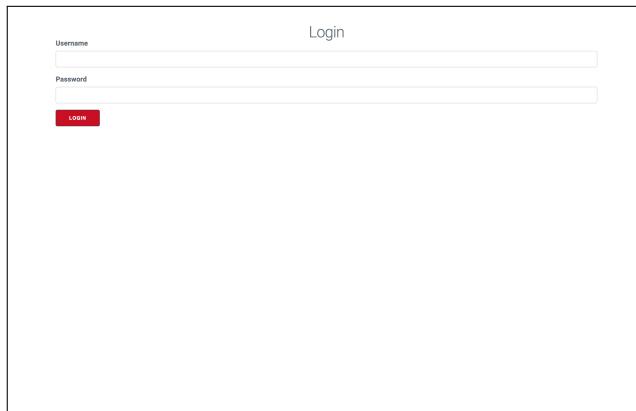
A screenshot of a web-based login form. The title bar says "Login". There are two input fields: "Username" and "Password", both containing placeholder text. Below the password field is a red "LOGIN" button.

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.

Id	Sender Name	Departure Agency Id	Receiver Name	Receiver Address	Destination Agency Id	Cost	Tracking		
								VIEW	EDIT
126	Primo Cliente	101	Primo Ricevitore	Via Giorgio Arizzo, Italia 52100	120	10.0	IDLE	<button>VIEW</button>	<button>EDIT</button>
127	Tutto Tutti	101	Secondo Ricevitore	Via Santa Marta Firenze, Italia 50100	121	20.0	IDLE	<button>VIEW</button>	<button>EDIT</button>

Id	Sender Name	Departure Agency Id	Receiver Name	Receiver Address	Destination Agency Id	Cost	Tracking		
								VIEW	EDIT
								<button>VIEW</button>	<button>EDIT</button>

Customer Home Page

Figura 5.2: Homepage dell'Operator

Id	Sender Name	Departure Agency Id	Receiver Name	Receiver Address	Destination Agency Id	Cost	Tracking		
								VIEW	EDIT
127	Tutto Tutti	101	Secondo Ricevitore	Via Santa Marta Firenze, Italia 50100	121	20.0	IDLE	<button>VIEW</button>	<button>EDIT</button>

Id	Sender Name	Departure Agency Id	Receiver Name	Receiver Address	Destination Agency Id	Cost	Tracking		
								VIEW	EDIT
131	Tutto Tutti	101	Quarto Ricevitore	Via Trento Firenze, Italia 50100	123	60.0	DELIVERED	<button>VIEW</button>	

Customer Home Page

Figura 5.3: Homepage del Customer

5.2 Pagine di gestione della Waybill

Le pagine di modifica e visualizzazione della waybill hanno permesso di testare l'uso dei parametri http e l'annotazione @ViewScoped. La pagina di modifica di una waybill è mostrata in Figura 5.5.

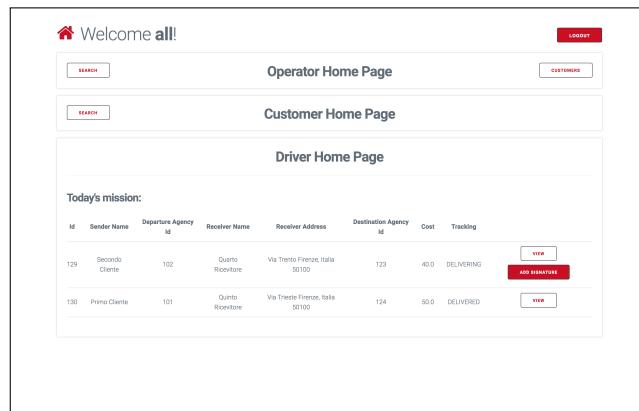


Figura 5.4: Homepage del Driver

The screenshot shows the 'Waybill Edit as Operator:' page with the following sections:

- Sender:** Fields for ID (108), Name (Primo Cliente), and Address (Via menchetti 24 Arezzo, 52100).
- Receiver:** Fields for Name, Phone, Email, Street, City, CAP, and State.
- Departure Agency** and **Destination Agency** tables with columns: ID, name, Address, and Description.
- Cost:** Text field containing '10.0'.

Figura 5.5: Pagina di modifica della Waybill

5.3 Pagine di gestione dei Customer

Le pagine relative alla gestione dei Customer sono 5:

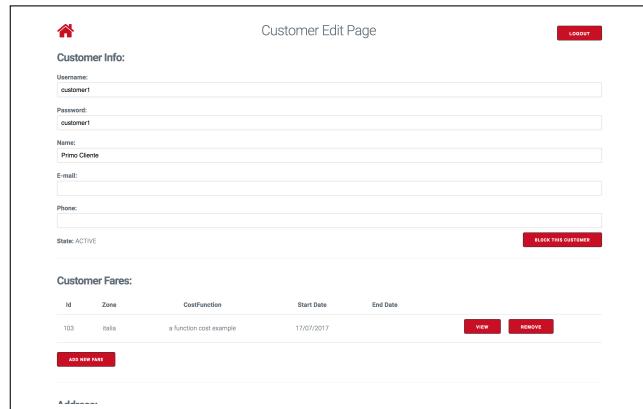
1. la pagina di visualizzazione dell'elenco dei Customer
2. la pagina di visualizzazione del Customer
3. la pagina di modifica del Customer
4. la pagina di visualizzazione della Fare
5. la pagina di modifica della Fare

Fare queste pagine ha permesso di testare il corretto funzionamento dell'annotazione `@ConversationScoped`, usata in `CustomerBean`. Questo bean, infatti, è iniettato da CDI in ciascuno dei loro controller.

Alcune di queste pagine sono mostrate nelle Figure 5.6, 5.7 e 5.8.

Customers:					
ID	Username	Name	Email	Phone	Status
108	customer1	Primo Cliente			ACTIVE <button>VIEW</button>
116	ali	Tutto Tutti			ACTIVE <button>VIEW</button>

Figura 5.6: Pagina di visualizzazione dei Customer



The screenshot shows the 'Customer Edit Page'. At the top right is a red 'Logout' button. Below it, the title 'Customer Edit Page' is centered. On the left, there's a small house icon. The main area is divided into sections: 'Customer Info:' and 'Customer Fares:'.

Customer Info:

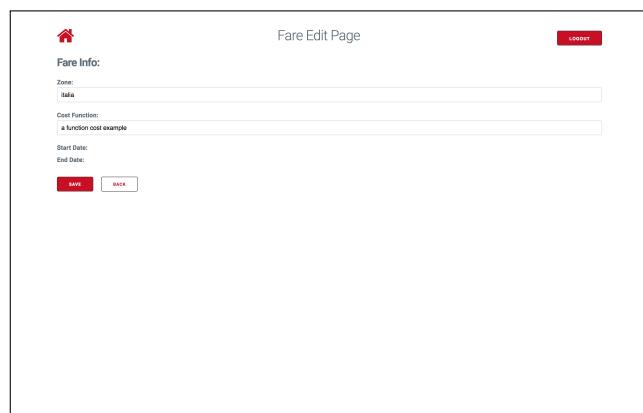
- Username: customer1
- Password: customer1
- Name: Primo Cliente
- E-mail: (empty input field)
- Phone: (empty input field)
- State: ACTIVE

Customer Fares:

ID	Zone	CostFunction	Start Date	End Date	Action
103	Italia	a function cost example	17/07/2017		VIEW REMOVE

ADD NEW FARE

Figura 5.7: Pagina di modifica del Customer



The screenshot shows the 'Fare Edit Page'. At the top right is a red 'Logout' button. Below it, the title 'Fare Edit Page' is centered. On the left, there's a small house icon. The main area is divided into sections: 'Fare Info:'.

Fare Info:

- Zone: Italia
- Cost Function: a function cost example
- Start Date: (empty input field)
- End Date: (empty input field)

SAVE BACK

Figura 5.8: Pagina di modifica della Fare

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, il sistema dovrebbe avere un comportamento prevedibile in ogni situazione.

Il naturale sviluppo futuro del progetto è il completamento del layer di interfaccia e una fase più approfondita di usability test che verifichi come un vero utente interagisce con il sistema.

Appendice A

Mockup

In questa Appendice sono presenti tutti i mockup elaborati.

The screenshot shows the Operator Homepage interface. At the top right, it says "Welcome, OP356357" with a user icon. Below that is a horizontal line. Underneath, there are two tables:

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	

On the right side of the interface, there are several 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:

Search by Customer: Sort by:

Search by Departure Agency: State: Search by Arrived Agency: Cost:

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

Figura A.10: Modify and View Fare