****

**ODD**

**Object Design Document**

**HELPSELLER**

Coordinatori del Progetto

| **Top Manager** | **Prof.ssa F. Ferrucci** |
| --- | --- |
| **Project Managers** | **Caccioppoli Mario, De Matteo Antonio** |

Partecipanti

| **Partecipante** | **Matricola** |
| --- | --- |
| Mario Caccioppoli | 0512 |
| Antonio De Matteo | 0512 |
| Daniele Iervolino | 051204975 |
| Francesco Ciampa | 0512103910 |
| Antonio Abate Chechile | 0512103670 |
| Marco Esposito | 0512106946 |
| Giovanni Aquaro | 0512105317 |
| Olexiy Lysytsya | 0512108083 |

# Sommario

**Object Design Document**

**1****Introduzione** [4](https://docs.google.com/document/d/1j9OxbZPlHS0_JElHY68e0pw6zHvcvPsiDosUAe2fiGM/edit#heading=h.wpsvnblkjp8l)

**1.1 Definizioni, acronimi e abbreviazioni** 5

**1**[**.2**](https://docs.google.com/document/d/1j9OxbZPlHS0_JElHY68e0pw6zHvcvPsiDosUAe2fiGM/edit#heading=h.8l8c5vke06wz) **Analisi dei trade-off 6**

Prestazioni VS Cost**i**

Comprensibilità VS Costi

Tempi VS Costi

Scalabilità VS Prestazioni

Tempo di risposta VS Spazio di memoria

Usabilità vs Costi

Sicurezza vs Tempi

**1.3 Componenti off-the-shelf 6**

**1**[**.4**](https://docs.google.com/document/d/1j9OxbZPlHS0_JElHY68e0pw6zHvcvPsiDosUAe2fiGM/edit#heading=h.f0pj4tkx9j6m) **Linee guida per la documentazione dell’interfaccia 7**

Nome dei File

Struttura dei File Sorgente

Formattazione

Organizzazione dei file

**1**[**.5**](https://docs.google.com/document/d/1j9OxbZPlHS0_JElHY68e0pw6zHvcvPsiDosUAe2fiGM/edit#heading=h.5klibphtkrzp) **Design pattern 10**

**2 Packages**

# Introduzione

Help Seller è una piattaforma che si propone di semplificare le interazioni tra le Aziende ed i distributori, semplificando la distribuzione di prodotti di beni di consumo di tipo alimentare e non, nel territorio.

Questo documento, usato come supporto dell’implementazione, ha lo scopo di produrre un modello capace di integrare in modo coerente e preciso tutti i servizi individuati e descritti nelle fasi precedenti. In particolare definisce le interfacce delle classi, le operazioni, i tipi, gli argomenti e la signature dei sottosistemi definiti nel System Design.

In questa prima sezione del documento, verranno descritti i trade-offs e le linee guida per la fase di implementazione, riguardanti la nomenclatura, la documentazione e le convenzioni sui formati.

# 1.1 Definizioni, acronimi e abbreviazioni

| **Termine** | **Definizione** |
| --- | --- |
| **Front-end** | la parte visibile all'utente di un [programma](https://it.wikipedia.org/wiki/Programma_(informatica)) e con cui egli può interagire — tipicamente un'[interfaccia utente](https://it.wikipedia.org/wiki/Interfaccia_utente) |
| **Back-end** | La parte che permette l'effettivo funzionamento di queste interazioni. |
| **Framework** | Un'architettura logica di supporto (spesso un'implementazione logica di un particolare design pattern) sulla quale un software può essere progettato e realizzato, spesso facilitandone lo sviluppo |
| **DB** | Abbreviazione di database, entità facente parte della struttura complessiva del progetto che consente la memorizzazione di informazioni |
| **Enterprise Java Bean** | componenti software che implementano, lato server, la logica di business di un’applicazione web all'interno dell'architettura Java EE espletando servizi a favore della parte di front-end |
| **Java EE** | Java EE è l’acronimo di Java Enterprise Edition, oggi sostituito da Jakarta Enterprise Edition, è un insieme di specifiche le cui implementazioni vengono principalmente sviluppate in [linguaggio di programmazione](https://it.wikipedia.org/wiki/Linguaggio_di_programmazione) [Java](https://it.wikipedia.org/wiki/Linguaggio_Java) e ampiamente utilizzata nella [programmazione Web](https://it.wikipedia.org/wiki/Programmazione_Web). |
| **JPA** | Le ‘Java Persistence API’ sono un framework per il linguaggio di programmazione Java che si occupa della gestione della persistenza dei dati di un DBMS relazionale nelle applicazioni che usano le piattaforme Java Platform, Standard Edition e **Java Enterprise Edition**. |
| **Spring** | In informatica Spring è un framework open source per lo sviluppo di applicazioni su piattaforma Java. Con Spring Boot, è possibile creare una applicazione avente un metodo main che lancia l'intera applicazione web, compreso il web server integrato. |
| **Maven** | In informatica Apache Maven è uno strumento di gestione di progetti software basati su Java e build automation. Maven usa un costrutto conosciuto come Project Object Model (POM); un file XML che descrive le dipendenze fra il progetto e le varie versioni di librerie necessarie nonché le dipendenze fra di esse. In questo modo si separano le librerie dalla directory di progetto utilizzando questo file descrittivo per definirne le relazioni. |

# 1[.2](https://docs.google.com/document/d/1j9OxbZPlHS0_JElHY68e0pw6zHvcvPsiDosUAe2fiGM/edit#heading=h.8l8c5vke06wz) Analisi dei trade-off

**Prestazioni VS Costi**

Considerando il sistema che stiamo realizzando, possiamo dire che il non eccessivo budget a nostra disposizione ci ha spinto a realizzare la piattaforma utilizzando software open source minimizzando così i costi ma al tempo stesso ottenendo una piattaforma stabile e funzionale.

**Comprensibilità VS Costi**

La comprensibilità del codice è un aspetto molto importante soprattutto per la fase di testing. Ogni classe e metodo deve essere facilmente interpretabile anche da chi non ha collaborato al progetto. Si preferisce quindi aggiungere costi per la documentazione al fine di rendere il codice comprensibile anche alle persone non coinvolte nel progetto o che sono subentrate nel team dopo l’implementazione della piattaforma.

**Tempi VS Costi**

Per l'implementazione del sistema si è scelto di utilizzare lo Spring Framework e Maven per il lato back-end, per il front-end invece si è ricorso ad Angular e Bootstrap. Nonostante la poca esperienza del team con queste tecnologie, l'utilizzo di tali specifiche ci permette di risparmiare moltissimo tempo sulla scrittura di codice che verrà generato automaticamente. Questo risparmio è vantaggioso rispetto al tempo necessario all'apprendimento.

**Scalabilità VS Prestazioni**

Data la sua funzione, la scalabilità costituisce uno degli obiettivi chiave durante l’implementazione della piattaforma. Il quantitativo di prodotti offerti dalle aziende può crescere nel tempo ma in particolare la mole di distributori che interagiscono col sistema può crescere rapidamente ed all’improvviso, pertanto è fondamentale che la piattaforma sia in grado di affrontare eventuali picchi di utenza.

**Tempo di risposta VS Spazio di memoria**

La scelta di utilizzare un DB relazionale è scaturita dai diversi vantaggi che se ne derivano: - Gestione consistente dei dati; - Tempo di risposta basso rispetto all’utilizzo del file system; - Accesso veloce e concorrente ai dati.

Lo svantaggio nell’utilizzo di un DB relazionale è che richiede il triplo dello spazio di memoria rispetto ad un file system e si trovano difficoltà nel momento in cui si vuole gestire enormi quantità di dati.

Per questo motivo durante la progettazione del database si è cercato di evitare di memorizzare informazioni ridondanti ma bensì limitarsi a quelle strettamente necessarie per il corretto funzionamento dei servizi offerti.

**Usabilità vs Costi**

L’esperienza dell’utente sulla piattaforma deve essere il più piacevole e meno frustrante possibile, si è deciso quindi di aggiungere costi per rendere il sistema quanto più usabile possibile. La navigazione sarà facile ed intuitiva e le operazioni saranno quanto più immediate possibili.

**Sicurezza vs Tempi**

Per garantire la sicurezza dei dati e per limitare atti di sabotaggio da parte di malintenzionati si è deciso di favorire la sicurezza inserendo form di autenticazione e metodi di verifica da parte degli amministratori del sistema. Questo rende più lenta l'esecuzione di alcune operazioni.

# 1.3 Componenti off-the-shelf

Come già detto nel paragrafo relativo ai trade offs, durante lo sviluppo dell’applicazione si è fatto uso di componenti off-the-shelf: componenti software disponibili sul mercato per facilitare la creazione del progetto. Per il sistema che si vuole realizzare ci interessa un framework per applicazioni web, per il front-end oltre che diverse librerie. Il framework che andremo ad utilizzare lato front-end è Angular, che è un framework open source che contiene una raccolta di strumenti liberi per la creazione di siti e applicazioni per il Web. Angular offre la possibilità di costruire applicazioni garantendo il riuso del codice oltre che a offrire la possibilità di rendere il sistema applicativo multipiattaforma.

Lato back-end invece si è fatto ricorso al framework “Springboot” che agevola lo sviluppo di applicazione stand-alone production-grade e Spring based. Tra le funzionalità principali offerte da Springboot troviamo:

* integrazione di Tomcat, il programmatore non deve preoccuparsi della gestione del container su cui girerà l’applicazione
* creazione di applicazioni stand-alone che quindi non richiedono software di terzi parti per il funzionamento
* offerta di dependencies Maven predefinite semplificando la build configuration
* configurazione automatica di Spring e librerie di terze parti

Spring non è altro che un ulteriore framework open source per lo sviluppo di applicazioni java. È stato largamente riconosciuto all'interno della comunità Java quale valida alternativa al modello basato su Enterprise Java Bean (EJB). Rispetto a quest'ultimo, il framework Spring lascia una maggiore libertà al programmatore fornendo allo stesso tempo un'ampia e ben documentata gamma di soluzioni semplici adatte alle problematiche più comuni.

Anche nello sviluppo della ‘componente intelligente’ sono state utilizzate diverse librerie che hanno fornito validi strumenti per la realizzazione dell’agente, applicando la filosofia del “non reinventare la ruota”. Nello specifico si è fatto ricorso alle seguenti librerie Python:

* Pandas il quale offre diversi metodi volti all’esecuzione di metodi matematici quali ad esempio la decomposizione ai valori singolari
* Numpy per la gestione di strutture quali matrici
* Matplotlib per la generazione di grafici volti all’analisi dei dati trattati

# 1[.4](https://docs.google.com/document/d/1j9OxbZPlHS0_JElHY68e0pw6zHvcvPsiDosUAe2fiGM/edit#heading=h.f0pj4tkx9j6m) Linee guida per la documentazione dell’interfaccia

* Prima dell’implementazione della logica di un qualsiasi indicatore è opportuno sottomettere alle Project Manager, l’algoritmo che si intende seguire per l’implementazione di questi, in modo che eventuali correzioni della logica da seguire possano essere apportate prima di imbattersi nella sintassi degli strumenti scelti.
* Ogni metodo e ogni file devono essere preceduti da un commento, o più precisamente da una documentazione che riporti l’obiettivo che si vuole e deve raggiungere con il nome/i dell’autore/i. Inoltre bisogna commentare, giustificare delle decisioni particolari o dei calcoli.
* La convenzione che deve essere adottata da tutti i team member per quanto riguarda i nomi delle variabili, è la nota **Camel Notation**, eccezion fatta per i nomi delle table e degli attributi presenti nel DB. Ciò è dovuto dal fatto che durante lo sviluppo del backend la connessione al server usufruisce di JPA, il quale necessita di una sua sintassi specifica, ad esempio anziché avere ordineUtente avremo ordine\_utente.
  + 1. **Nome dei File**

La convenzione per quanto riguarda i nomi dei file (ma anche delle operazioni e delle variabili) è quella di avere nomi evocativi, ma soprattutto in lingua italiana. Eccezione fatta per tutti quei termini tipici dello sviluppo nei linguaggi utilizzati e per quei termini ove anche la migliore traduzione in italiano non regge il confronto.

Il nome del file sorgente corrisponde al nome della classe contenuta al suo interno con distinzione tra maiuscole e minuscole. Al nome della classe possono essere aggiunti tre suffissi:

* .java per il file sorgente;
* .class per il Bytecode del file.
* .py per gli script in Python del modulo FIA

**1.4.2 Struttura dei File Sorgente**

Ogni file sorgente contiene esattamente una sola classe pubblica o interfaccia. Ogni sezione del File deve essere separata con una linea vuota. La struttura interna del File è la seguente:

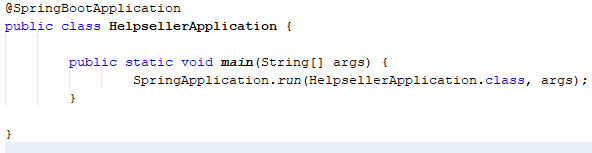
* **Descrizione Classe**: Tutti i file devono iniziare con un commento nel quale è esplicitato il nome della classe, una breve descrizione, l’autore, la versione e la data di creazione e infine le informazioni di Copyright.
* **Dichiarazione del Package e degli Import**: La prima istruzione dopo la descrizione della classe deve essere la dichiarazione del Package, successivamente è possibile dichiarare gli import necessari. Le dichiarazioni del Package e degli Import sono separati da una linea vuota. Infine gli Import sono ordinati come segue:
* ○ Tutti gli Import Statici in un singolo blocco;
* ○ Tutti gli Import Non Statici in un singolo blocco.

Ogni blocco è separato da una linea vuota

* **Dichiarazione della Classe o dell’Interfaccia**: Il file sorgente deve contenere una sola Classe o una sola Interfaccia. Gli elementi contenuti al loro interno devono essere ordinati nel seguente modo:
  + Dichiarazione Variabili Statiche della Classe: Prima le variabili di classe public, poi le variabili protected e infine quelle private;
  + Dichiarazioni Variabili di istanza: Prima le variabili di classe public, poi le variabili protected e infine quelle private;
  + Costruttori;
  + Metodi: Questi metodi devono essere raggruppati in base alla loro funzionalità piuttosto che in base a regole di visibilità o accessibilità. I metodi con nomi simili devono essere vicini tra loro.

**1.4.3 Formattazione**

Non è specificato se l’indentazione del codice deve avvenire tramite l’utilizzo degli spazi o tramite l’utilizzo delle tabulazioni. E’ però importante per la leggibilità del codice, seguire un’opportuna indentazione guidata dal livello di profondità dell’istruzione

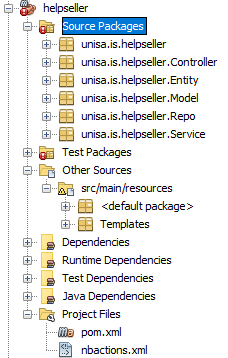


Nell’esempio troviamo al primo livello la definizione della classe, subito dopo la definizione del metodo e successivamente il contenuto di tale metodo.

I tre livelli hanno una profondità di indentazione diversa. Questa pratica deve essere usata soprattutto per le istruzioni FOR, IF dove è buona pratica scendere di livello.

**1.4.4** **Organizzazione dei file**

L’organizzazione generale delle directory dell’applicazione è già fortemente strutturata dai framework utilizzati, ad esempio per quanto riguarda il back-end le varie classi java vengono suddivise secondo la seguente struttura:



In **Source Packages** troviamo sei sotto-directory ognuna popolata da componenti dalla funzionalità ben definita, ad esempio in Entity troveremo la definizione degli oggetti che rappresentano le entità contenute nel DB, in Service troviamo i metodi forniti dalle entità.

Nella directory **Test Packages** troviamo quelli che sono i test eseguiti per assicurare il corretto funzionamento dell’applicazione In **Other Sources** possiamo trovare il file di configurazione “application.properties” fondamentale alla configurazione della connessione al DB. **Dependencies** conterrà tutte le dipendenze gestite automaticamente da Maven ed in **Project Files** possiamo trovare ulteriori file di configurazione.

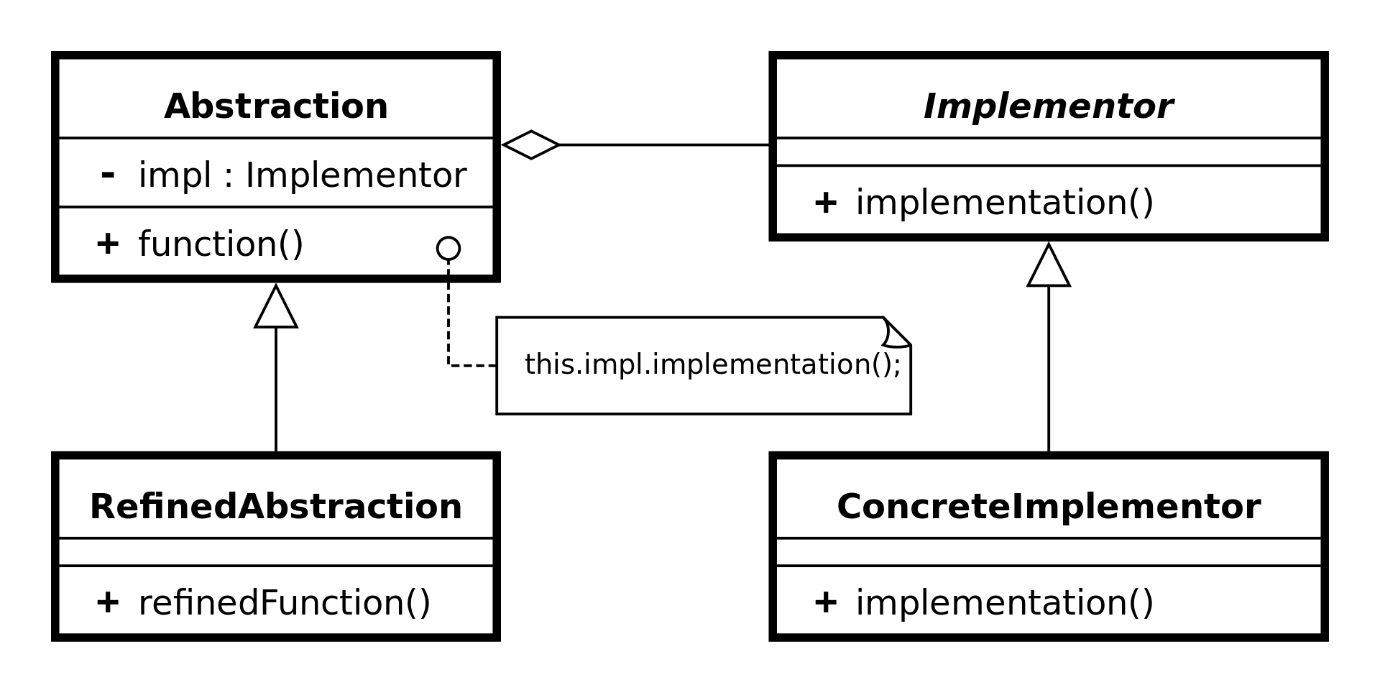
# 1[.5](https://docs.google.com/document/d/1j9OxbZPlHS0_JElHY68e0pw6zHvcvPsiDosUAe2fiGM/edit#heading=h.5klibphtkrzp) Design pattern

“Ogni pattern descrive un problema che ricorre più volte nel nostro ambiente, descrive poi il nucleo della soluzione del problema in modo da poter utilizzare tale soluzione un milione di volte senza mai farlo allo stesso modo”

Nell’ object oriented, i design pattern sono template di soluzioni utilizzabili per risolvere un insieme di problemi ricorrenti. I design pattern vengono classificati in tre famiglie:

* I PATTERN STRUTTURALI consentono di riutilizzare degli oggetti esistenti fornendo agli utilizzatori un'interfaccia più adatta alle loro esigenze.
* I PATTERN COMPORTAMENTALI forniscono soluzione alle più comuni tipologie di interazione tra gli oggetti.
* I PATTERN CREAZIONALI nascondono i costruttori delle classi e mettono dei metodi al loro posto creando un'interfaccia. In questo modo si possono utilizzare oggetti senza sapere come sono implementati.

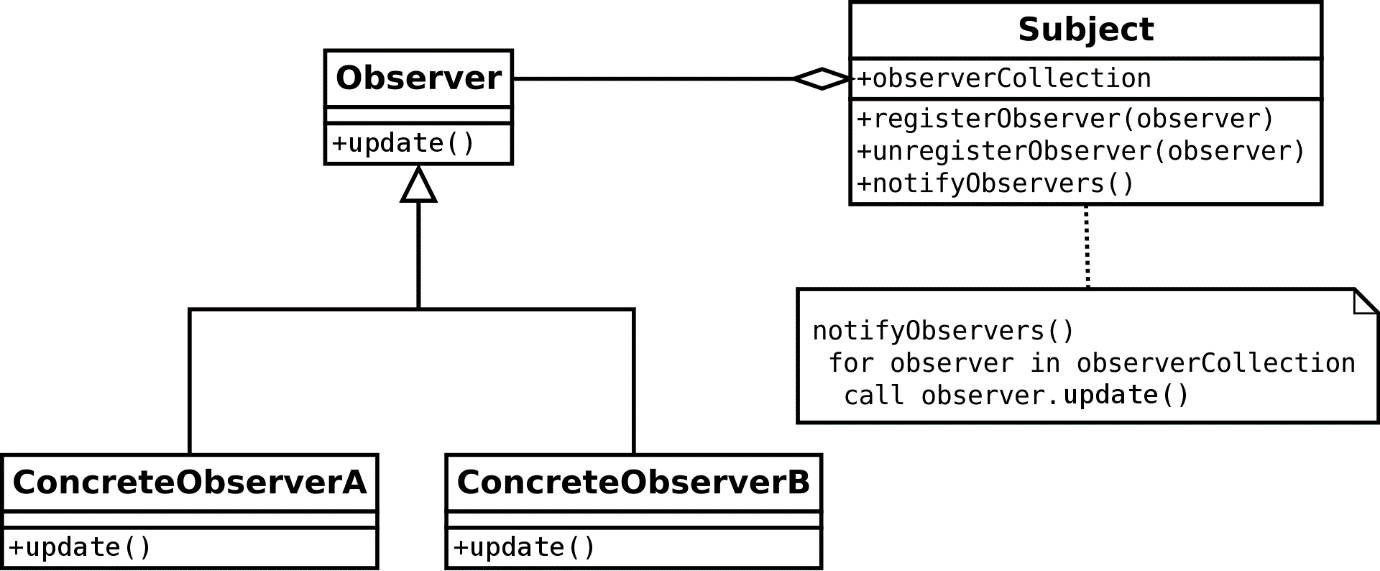
**Bridge Design Pattern**:



Tale pattern permette di separare l'interfaccia di una classe (che cosa si può fare con la classe) dalla sua implementazione (come lo fa). In tal modo si può usare l'ereditarietà per fare evolvere l'interfaccia o l'implementazione in modo separato.

Help Seller fa uso del Bridge Pattern per dare l’opportunità di sviluppare diverse implementazioni dello Storage Layer. Attualmente come sistema di storage è stato scelto di utilizzare un DBMS attraverso la tecnologia JPA. Grazie al Bridge Pattern in futuro sarà utilizzare tecnologie come Hibernate.

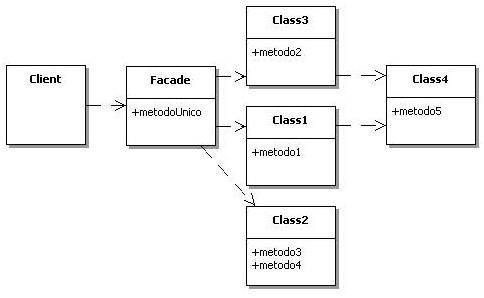
**Observer Design Pattern:**



il pattern si basa su uno o più oggetti, chiamati osservatori o observer, che vengono registrati per gestire un evento che potrebbe essere generato dall'oggetto "osservato", che può essere chiamato soggetto. Uno degli aspetti fondamentali è che tutto il funzionamento dell'observer si basa su meccanismi di [callback](https://it.wikipedia.org/wiki/Callback): in genere, una funzione, o un "blocco di codice" che viene passata come parametro ad un'altra funzione.

La piattaforma utilizza l’Observer Pattern per notificare i cambiamenti di stato tra gli oggetti dipendenti tra loro.

**Facade Pattern:**



Letteralmente façade significa "facciata", ed infatti nella programmazione ad oggetti indica un oggetto che permette, attraverso un'interfaccia più semplice, l'accesso a sottosistemi che espongono interfacce complesse e molto diverse tra loro, nonché a blocchi di codice complessi. Nell’esempio dell’immagine l’interfaccia ‘facade’ consente al client di utilizzare le classi descritte dalle varie classi attraverso una singola implementazione, nascondendo la complessità dell’operazione.

In Help Seller il design pattern del façade è protagonista nella parte beck-end dell’applicazione.

# 3Interfacce delle classi

**3.1 AmministratoreController**

| **NOME CLASSE** | AmministratoreController |
| --- | --- |
| **DESCRIZIONE** | Questa classe fornisce i servizi per l’Amministratore |
| **METODI** | +findAll(): List<Amministratore>  +findId(int i): Amministratore  +deleteId(int i):boolean |

| **NOME CLASSE** | Amministratore |
| --- | --- |
| **NOME METODO** | +findAll(): List<Amministratore> |
| **DESCRIZIONE** | Questo metodo restituisce la lista di amministratori |
| **PRE-CONDIZIONI** | context: Amministratore findAll() pre: List<Amministratore> != null |
| **POST-CONDIZIONI** | / |

| **NOME CLASSE** | Amministratore |
| --- | --- |
| **NOME METODO** | +findId(int i): Amministratore |
| **DESCRIZIONE** | Questo metodo restituisce l’amministratore con id=i |
| **PRE-CONDIZIONI** | context: Amministratore findId() pre: Amministratore!= null |
| **POST-CONDIZIONI** | / |

| **NOME CLASSE** | Amministratore |
| --- | --- |
| **NOME METODO** | +deleteId(int i): boolean |
| **DESCRIZIONE** | Questo metodo elimina l’amministratore con l id=i |
| **PRE-CONDIZIONI** | context: Amministratore deleteId(int i) pre: Amministratore != null |
| **POST-CONDIZIONI** | / |

**3.2 AziendaController**

| **NOME CLASSE** | AziendaController |
| --- | --- |
| **DESCRIZIONE** | Questa classe fornisce i servizi per l’azienda |
| **METODI** | +findAll(): List<Azienda>  +findId(int i): Azienda  +deleteId(int i):boolean |

| **NOME CLASSE** | Azienda |
| --- | --- |
| **NOME METODO** | +findAll(): List<Azienda> |
| **DESCRIZIONE** | Questo metodo trova tutte le aziende |
| **PRE-CONDIZIONI** | context: Azienda findAll() pre: List<Azienda> != null |
| **POST-CONDIZIONI** | / |

| **NOME CLASSE** | Azienda |
| --- | --- |
| **NOME METODO** | +findId(int i): Azienda |
| **DESCRIZIONE** | Questo metodo trova l’azienda con l’id=i |
| **PRE-CONDIZIONI** | context: Azienda findId(int i) pre: Azienda != null |
| **POST-CONDIZIONI** | / |

| **NOME CLASSE** | Azienda |
| --- | --- |
| **NOME METODO** | +deleteId(int i): boolean |
| **DESCRIZIONE** | Questo metodo elimina l’azienda con l’id=i |
| **PRE-CONDIZIONI** | context: Azienda deleteId(int i) pre: Azienda != null |
| **POST-CONDIZIONI** | / |

**3.3 DistributoreController**

| **NOME CLASSE** | DistributoreController |
| --- | --- |
| **DESCRIZIONE** | Questa classe fornisce i servizi per il Distributore |
| **METODI** | +findAll(): List<Distributore>  +findId(int i): Distributore  +deleteId(int i):boolean |

| **NOME CLASSE** | Distributore |
| --- | --- |
| **NOME METODO** | +findAll(): List<Distributore> |
| **DESCRIZIONE** | Questo metodo trova tutti i distributori |
| **PRE-CONDIZIONI** | context: Distributore findAll() pre: List<Distributore> != null |
| **POST-CONDIZIONI** | / |

| **NOME CLASSE** | Distributore |
| --- | --- |
| **NOME METODO** | +findId(int i): Distributore |
| **DESCRIZIONE** | Questo metodo trova il Distributore con id=i |
| **PRE-CONDIZIONI** | context: Distributore findId(int i) pre: Distributore != null |
| **POST-CONDIZIONI** | / |

| **NOME CLASSE** | Distributore |
| --- | --- |
| **NOME METODO** | +deleteId(int i): boolean |
| **DESCRIZIONE** | Questo metodo elimina il Distributore con id=i |
| **PRE-CONDIZIONI** | context: Distributore deleteId(int i) pre: Distributore != null |
| **POST-CONDIZIONI** | / |

**3.4 DocumentoController**

| **NOME CLASSE** | DocumentoController |
| --- | --- |
| **DESCRIZIONE** | Questa classe fornisce i servizi per Documento |
| **METODI** | +findAll(): List<Documento>  +findId(int i): Documento  +deleteId(int i):boolean |

| **NOME CLASSE** | Documento |
| --- | --- |
| **NOME METODO** | +findAll(): List<Documento> |
| **DESCRIZIONE** | Questo metodo trova tutti i documenti |
| **PRE-CONDIZIONI** | context: Documento findAll() pre: List<Documento> != null |
| **POST-CONDIZIONI** | / |

| **NOME CLASSE** | Documento |
| --- | --- |
| **NOME METODO** | +findId(int i): Documento |
| **DESCRIZIONE** | Questo metodo trova il documento con id=i |
| **PRE-CONDIZIONI** | context: Documento findId() pre: Documento != null |
| **POST-CONDIZIONI** | / |

| **NOME CLASSE** | Documento |
| --- | --- |
| **NOME METODO** | +findId(int i): boolean |
| **DESCRIZIONE** | Questo metodo elimina il documento con id=i |
| **PRE-CONDIZIONI** | context: Documento deleteId() pre: Documento != null |
| **POST-CONDIZIONI** | / |

**3.5 OrdineController**

| **NOME CLASSE** | OrdineController |
| --- | --- |
| **DESCRIZIONE** | Questa classe fornisce i servizi per Ordine |
| **METODI** | +findAll(): List<Ordine>  +findId(int i): Ordine  +deleteId(int i):boolean |

| **NOME CLASSE** | Ordine |
| --- | --- |
| **NOME METODO** | +findAll(): List<Ordine> |
| **DESCRIZIONE** | Questo metodo trova tutti gli Ordini |
| **PRE-CONDIZIONI** | context: Ordine findAll() pre: List<Ordine> != null |
| **POST-CONDIZIONI** | / |

| **NOME CLASSE** | Ordine |
| --- | --- |
| **NOME METODO** | +findId(int i): Ordine |
| **DESCRIZIONE** | Questo metodo trova l’Ordine con id=i |
| **PRE-CONDIZIONI** | context: Ordine findId(int i) pre: Ordine != null |
| **POST-CONDIZIONI** | / |

| **NOME CLASSE** | Ordine |
| --- | --- |
| **NOME METODO** | +deleteId(int i): boolean |
| **DESCRIZIONE** | Questo metodo elimina l’Ordine con id=i |
| **PRE-CONDIZIONI** | context: Ordine deleteId(int i) pre: Ordine != null |
| **POST-CONDIZIONI** | / |

**3.6 OrdineProdottoController**

| **NOME CLASSE** | OrdineProdottoController |
| --- | --- |
| **DESCRIZIONE** | Questa classe fornisce i servizi per OrdineProdotto |
| **METODI** | +findAll(): List<OrdineProdotto>  +findId(int i): OrdineProdotto  +deleteId(int i):boolean |

| **NOME CLASSE** | OrdineProdotto |
| --- | --- |
| **NOME METODO** | +findAll(): List<OrdineProdotto> |
| **DESCRIZIONE** | Questo metodo trova tutti gli “OrdineProdotto” |
| **PRE-CONDIZIONI** | context: OrdineProdotto findAll() pre: List<OrdineProdotto> != null |
| **POST-CONDIZIONI** | / |

| **NOME CLASSE** | OrdineProdotto |
| --- | --- |
| **NOME METODO** | +findId(int i): OrdineProdotto |
| **DESCRIZIONE** | Questo metodo trova l’OrdineProdotto con id=i |
| **PRE-CONDIZIONI** | context: OrdineProdotto findId() pre: OrdineProdotto != null |
| **POST-CONDIZIONI** | / |

| **NOME CLASSE** | OrdineProdotto |
| --- | --- |
| **NOME METODO** | +deleteId(int i): boolean |
| **DESCRIZIONE** | Questo metodo elimina l’OrdineProdotto con id=i |
| **PRE-CONDIZIONI** | context: OrdineProdotto findId() pre: OrdineProdotto != null |
| **POST-CONDIZIONI** | / |

**3.7 ProdottoController**

| **NOME CLASSE** | ProdottoController |
| --- | --- |
| **DESCRIZIONE** | Questa classe fornisce i servizi per Prodotto |
| **METODI** | +findAll(): List<Prodotto>  +findId(int i): Prodotto  +deleteId(int i):boolean |

| **NOME CLASSE** | Prodotto |
| --- | --- |
| **NOME METODO** | +findAll(): List<Prodotto> |
| **DESCRIZIONE** | Questo metodo trova tutti i Prodotti |
| **PRE-CONDIZIONI** | context: Prodotto findAll() pre: List<Prodotto> != null |
| **POST-CONDIZIONI** | / |

| **NOME CLASSE** | Prodotto |
| --- | --- |
| **NOME METODO** | +findId(int i): Prodotto |
| **DESCRIZIONE** | Questo metodo trova il Prodotto con id=i |
| **PRE-CONDIZIONI** | context: Prodotto findId(int i) pre: Prodotto != null |
| **POST-CONDIZIONI** | / |

| **NOME CLASSE** | Prodotto |
| --- | --- |
| **NOME METODO** | +deleteId(int i): boolean |
| **DESCRIZIONE** | Questo metodo elimina il Prodotto con id=i |
| **PRE-CONDIZIONI** | context: Prodotto deleteId(int i) pre: Prodotto != null |
| **POST-CONDIZIONI** | / |

**3.8 RecensioneController**

| **NOME CLASSE** | RecensioneController |
| --- | --- |
| **DESCRIZIONE** | Questa classe fornisce i metodi per le Recensioni |
| **METODI** | +findAll(): List<Recensione>  +findId(int i): Recensione  +deleteId(int i):boolean |

| **NOME CLASSE** | Recensione |
| --- | --- |
| **NOME METODO** | +findAll(): List<Recensione> |
| **DESCRIZIONE** | Questo metodo trova tutte le Recensioni |
| **PRE-CONDIZIONI** | context: Recensione findAll() pre: List<Recensione> != null |
| **POST-CONDIZIONI** | / |

| **NOME CLASSE** | Recensione |
| --- | --- |
| **NOME METODO** | +findId(int i): Recensione |
| **DESCRIZIONE** | Questo metodo trova la Recensione con id=i |
| **PRE-CONDIZIONI** | context: Recensione findId(int i) pre: Recensione != null |
| **POST-CONDIZIONI** | / |

| **NOME CLASSE** | Recensione |
| --- | --- |
| **NOME METODO** | +deleteId(int i): boolean |
| **DESCRIZIONE** | Questo metodo elimina la Recensione con id=i |
| **PRE-CONDIZIONI** | context: Recensione deleteId(int i) pre: Recensione != null |
| **POST-CONDIZIONI** | / |

**3.9 ScontoController**

| **NOME CLASSE** | ScontoController |
| --- | --- |
| **DESCRIZIONE** | Questa classe fornisce i servizi per Sconto |
| **METODI** | +findAll(): List<Sconto>  +findId(int i): Sconto  +deleteId(int i):boolean |

| **NOME CLASSE** | Sconto |
| --- | --- |
| **NOME METODO** | +findAll(): List<Sconto> |
| **DESCRIZIONE** | Questo metodo trova tutti gli Sconti |
| **PRE-CONDIZIONI** | context: Sconto findAll() pre: List<Sconto> != null |
| **POST-CONDIZIONI** | / |

| **NOME CLASSE** | Sconto |
| --- | --- |
| **NOME METODO** | +findId(int i): Sconto |
| **DESCRIZIONE** | Questo metodo trova lo Sconto con id=i |
| **PRE-CONDIZIONI** | context: Sconto findId(int i) pre: Sconto != null |
| **POST-CONDIZIONI** | / |

| **NOME CLASSE** | Sconto |
| --- | --- |
| **NOME METODO** | +deleteId(int i): boolean |
| **DESCRIZIONE** | Questo metodo elimina lo Sconto con id=i |
| **PRE-CONDIZIONI** | context: Sconto deleteId(int i) pre: Sconto != null |
| **POST-CONDIZIONI** | / |

**3.10 ScontoProdottoController**

| **NOME CLASSE** | ScontoProdottoController |
| --- | --- |
| **DESCRIZIONE** | Questa classe fornisce i servizi per ScontoProdotto |
| **METODI** | +findAll(): List<ScontoProdotto>  +findId(int i): ScontoProdotto  +deleteId(int i):boolean |

| **NOME CLASSE** | ScontoProdotto |
| --- | --- |
| **NOME METODO** | +findAll(): List<ScontoProdotto> |
| **DESCRIZIONE** | Questo metodo trova tutti gli ScontiProdotto |
| **PRE-CONDIZIONI** | context: ScontoProdotto findAll() pre: List<ScontoProdotto> != null |
| **POST-CONDIZIONI** | / |

| **NOME CLASSE** | ScontoProdotto |
| --- | --- |
| **NOME METODO** | +findId(int i): ScontoProdotto |
| **DESCRIZIONE** | Questo metodo trova lo ScontoProdotto con id=i |
| **PRE-CONDIZIONI** | context: ScontoProdotto findId(int i) pre: ScontoProdotto != null |
| **POST-CONDIZIONI** | / |

| **NOME CLASSE** | ScontoProdotto |
| --- | --- |
| **NOME METODO** | +deleteId(int i): ScontoProdotto |
| **DESCRIZIONE** | Questo metodo elimina lo ScontoProdotto con id=i |
| **PRE-CONDIZIONI** | context: ScontoProdotto deleteId(int i) pre: ScontoProdotto != null |
| **POST-CONDIZIONI** | / |

**3.11 TrasportoController**

| **NOME CLASSE** | TrasportoController |
| --- | --- |
| **DESCRIZIONE** | Questa classe fornisce i servizi per Trasporto |
| **METODI** | +findAll(): List<Trasporto>  +findId(int i): Trasporto  +deleteId(int i):boolean |

| **NOME CLASSE** | Trasporto |
| --- | --- |
| **NOME METODO** | +findAll(): List<Trasporto> |
| **DESCRIZIONE** | Questo metodo trova tutti i Trasporti |
| **PRE-CONDIZIONI** | context: Trasporto findAll() pre: List<Trasporto> != null |
| **POST-CONDIZIONI** | / |

| **NOME CLASSE** | Trasporto |
| --- | --- |
| **NOME METODO** | +findId(int i): Trasporto |
| **DESCRIZIONE** | Questo metodo trova il Trasporto con id=i |
| **PRE-CONDIZIONI** | context: Trasporto findId(int i) pre: Trasporto != null |
| **POST-CONDIZIONI** | / |

| **NOME CLASSE** | Trasporto |
| --- | --- |
| **NOME METODO** | +deleteId(int i): Trasporto |
| **DESCRIZIONE** | Questo metodo elimina il Trasporto con id=i |
| **PRE-CONDIZIONI** | context: Trasporto deleteId(int i) pre: Trasporto != null |
| **POST-CONDIZIONI** | / |

**3.12 Utente Controller**

| **NOME CLASSE** | UtenteController |
| --- | --- |
| **DESCRIZIONE** | Questa classe fornisce i servizi per l’autenticazione e la registrazione degli users |
| **METODI** | +UtenteModel **auth**(String email, String password, String tipo)  +void **regAdmin**(String username, String password, String email)  +void **registrazione**(String email, String password, String nome\_azienda, String indirizzo, String vat, String descrizione, String logo)  +void **registrazione**(String username, String email, String password, String nome, String cognome, String telefono, String indirizzo\_sede, int id\_ordine\_prova, String vat) |

| **NOME CLASSE** | Utente |
| --- | --- |
| **NOME METODO** | **auth**(String email, String password, String tipo): UtenteModel |
| **DESCRIZIONE** | Questo metodo controlla che i dati inseriti lato frontend per il login siano validi o meno |
| **PRE-CONDIZIONI** | **context**: UtentService.auth(email, password, tipo) **pre**: email != null && password != null && tipo == ‘Amministratore’ || ‘Distributore’ || ‘Azienda’ |
| **POST-CONDIZIONI** | / |

| **NOME CLASSE** | Utente |
| --- | --- |
| **NOME METODO** | **regAdmin**(String username, String password, String email): void |
| **DESCRIZIONE** | Questo metodo effettua con la registrazione di un utente di tipo **Amministratore** e quindi l’inserimento di un nuovo utente nel DB della piattaforma |
| **PRE-CONDIZIONI** | **context**: UtentService.**regAdmin**(email, password, username) **pre**: email != null && password != null && username != null |
| **POST-CONDIZIONI** | / |

| **NOME CLASSE** | Utente |
| --- | --- |
| **NOME METODO** | **registrazione**(String email, String password, String nome\_azienda, String indirizzo, String vat, String descrizione, String logo): void |
| **DESCRIZIONE** | Questo metodo effettua con la registrazione di un utente di tipo **Azienda** e quindi l’inserimento di un nuovo utente nel DB della piattaforma |
| **PRE-CONDIZIONI** | **context**: UtenteService.registrazione **pre**: email != null && password != null && nome\_azienda != null && indirizzo != null && vat != null && descrizione != null && |
| **POST-CONDIZIONI** | / |

| **NOME CLASSE** | Utente |
| --- | --- |
| **NOME METODO** | **registrazione**(String email, String password, String nome\_azienda, String indirizzo, String vat, String descrizione, String logo): void |
| **DESCRIZIONE** | Questo metodo effettua con la registrazione di un utente di tipo **Distributore** e quindi l’inserimento di un nuovo utente nel DB della piattaforma |
| **PRE-CONDIZIONI** | **context**: UtenteService.registrazione **pre**: email != null && password != null && nome\_azienda != null && indirizzo != null && vat != null && descrizione != null && |
| **POST-CONDIZIONI** | / |