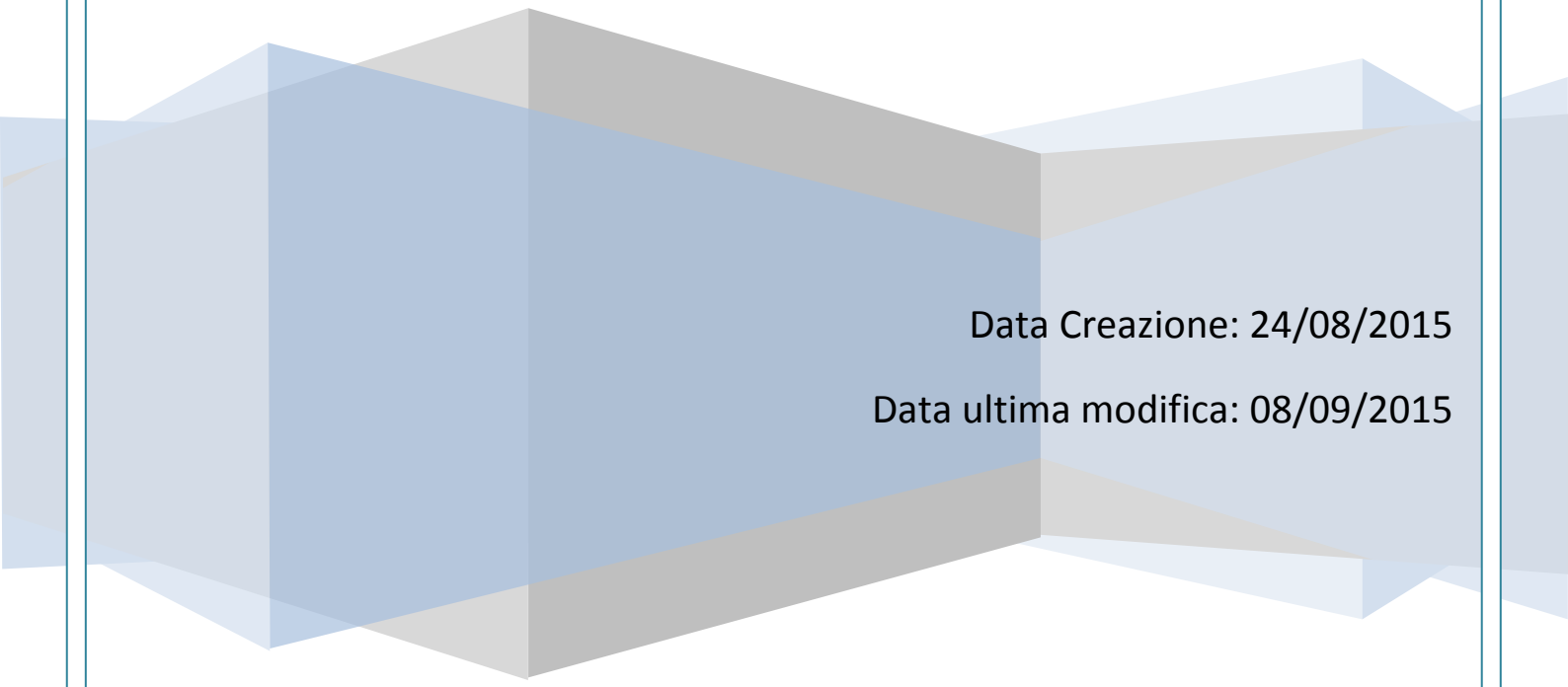
Documento di Progetto

Versione 1.2

Elisa Antolli

Alice Culaon

Diego Pillon



Data Creazione: 24/08/2015

Data ultima modifica: 08/09/2015

INDICE

1. MODELLO TABELLA DELLE REVISIONI

Rev./Ver.	Data	Descrizione	Autore
1/1.0	24-08-2015	Creazione del documento, paragrafo 5	Elisa Antolli
2/1.1	03-09-2015	Aggiunta paragrafi paragrafo 6, 7, 9: tecnologie	Elisa Antolli
3/1.2	07-09-2015	Diagramma di flusso, modifiche al testo; paragrafi 8, 10	Elisa Antolli

Tot. Rev. 3 **Versione corrente 1.2**

2. SCOPO DEL DOCUMENTO

Lo scopo di questo documento è descrivere e definire il Sistema in modo più specifico degli altri documenti già presentati; il documento è rivolto al team di sviluppo ed è la base per l'implementazione del Sistema.

3. AUTORI DEL DOCUMENTO

1. *Antolli Elisa;*
2. *Culaon Alice;*
3. *Pillon Diego;*

4. GLOSSARIO

Termine	Descrizione	Paragrafo
Back Office (BO)	(letteralmente retro-ufficio) è quella parte di un'azienda (o di un'organizzazione) che comprende tutte le attività proprie dell'azienda, come il sistema di produzione o la gestione. In pratica, il back office è tutto ciò che il cliente (o l'utente) non vede, ma che consente la realizzazione dei prodotti o dei servizi a lui destinati.	3, 10, 12
Front Office (FO)	Nell'organizzazione aziendale il termine front office (letteralmente fronte-ufficio), indica l'insieme delle strutture di un'organizzazione che gestiscono l'interazione con il cliente. Il Front office è il reparto amministrativo dell'organizzazione che si occupa del ciclo cliente.	3, 7, 8, 10, 12
HTML	HyperText Markup Language (HTML) , traduzione letterale: linguaggio a marcatori per ipertesti, è il linguaggio di markup solitamente usato per la formattazione e impaginazione di documenti ipertestuali disponibili nel World Wide Web sotto forma di pagine web	3, 4, 10, 11
Http	HyperText Transfer Protocol (HTTP), protocollo di trasferimento di un ipertesto, è usato come principale sistema per la trasmissione d'informazioni sul web ovvero in un'architettura tipica client-server	3, 4, 11
Sistema	Il prodotto software in oggetto che ha la funzione di gestione delle prenotazioni di una fumetteria	2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 14

Termini del Glossario: 5

5. DEFINIZIONE DELL' ARCHITETTURA DEL SISTEMA

In questa sezione l'architettura del Sistema viene spiegata più in dettaglio, specificando come le tecnologie scelte vengono utilizzate e dove vengono utilizzate.

5.1. L' architettura di base

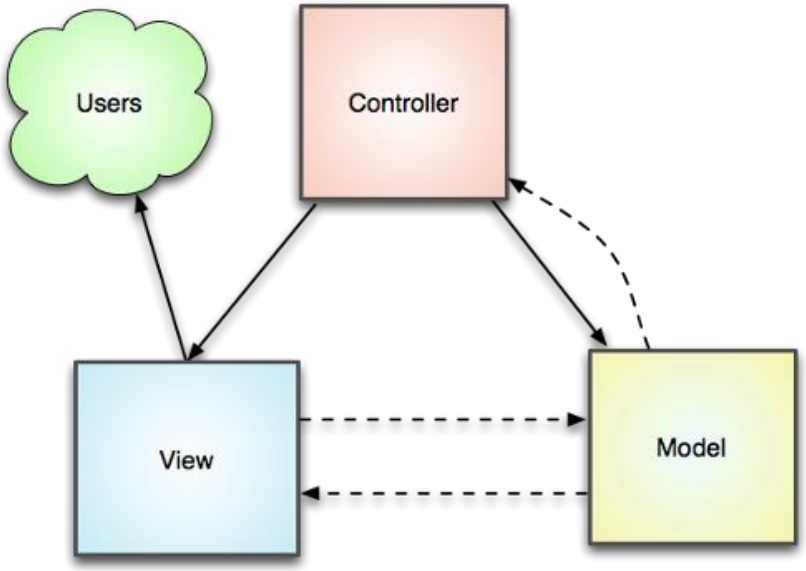
Il Sistema viene sviluppato con tecnologia web: con il termine applicazione web, in generale, si indicano sistemi informatici progettati per operare attraverso internet e tramite l'utilizzo di un browser; applicazioni sviluppate utilizzando tecnologie web, HTML, JavaScript e CSS. Può essere eseguito da un server HTTP (*Host web*) o localmente sul dispositivo dell'utente. La funzione di *server web* è quello di ricevere una richiesta (richiesta) e ritorno (risposta) al *browser* dell'utente. Il *browser* consente all'utente di richiedere risorse, e quando il server risponde a una richiesta si ottengono pagine HTML, immagini e documenti PDF che vengono poi visualizzati per l'utente. Di solito i server inviano istruzioni al *browser* in formato HTML. Queste istruzioni dicono al *browser* come presentare i contenuti per l'utente web. Il Sistema sarà depositato su un web server, progettato per servire le richieste di ogni cliente simultaneamente. In generale i dati vengono salvati ed estratti con il supporto di un Database Server, mentre per la costruzione delle pagine dinamiche, viene utilizzato un'applicazione di supporto basata su *Servlets*.

Per questo abbiamo scelto di utilizzare il modello di architettura MVC – *Model View Controller;* Il MVC è stato creato negli anni '70, ed è ancora un modello applicabile in varie applicazioni, in particolare le applicazioni web. Con la crescente complessità delle applicazioni sviluppate, diventa essenziale la separazione tra i dati (*Model*) e il layout (*View*). Pertanto, le modifiche apportate al layout non influenzano la manipolazione dei dati, e possono essere riorganizzati senza modificare il layout.

1. Il *Model* è la parte dove vengono specificate le rappresentazione di "dominio" – che dicono come vengono strutturati i dati dentro il Sistema – ad esempio una classe "Author", con i suoi determinati attributi e metodi *get/set*. In poche parole sono la rappresentazione degli elementi del dominio e interazione con gli strumenti di persistenza;
2. Il *View* è la parte dove il *Model* viene mostrato all' utente, in una determinata forma permettendo l'interazione tra utente e Sistema. Tecnicamente, sono le interfacce di risposta in HTML (JSP,ASP, PHP);
3. Il *Controller* è costituito dai processi e dalle risposte agli eventi; tipicamente azioni degli utenti che possono cambiare il Model. Nel controller viene fatta la validazione dei dati ed è anche il "luogo" dove i valori inseriti dagli utenti vengono "filtrati".

Una semplice dimostrazione grafica del modello MVC viene riportata in figura 1.

Figura 1: Rappresentazione generica Modello MVC

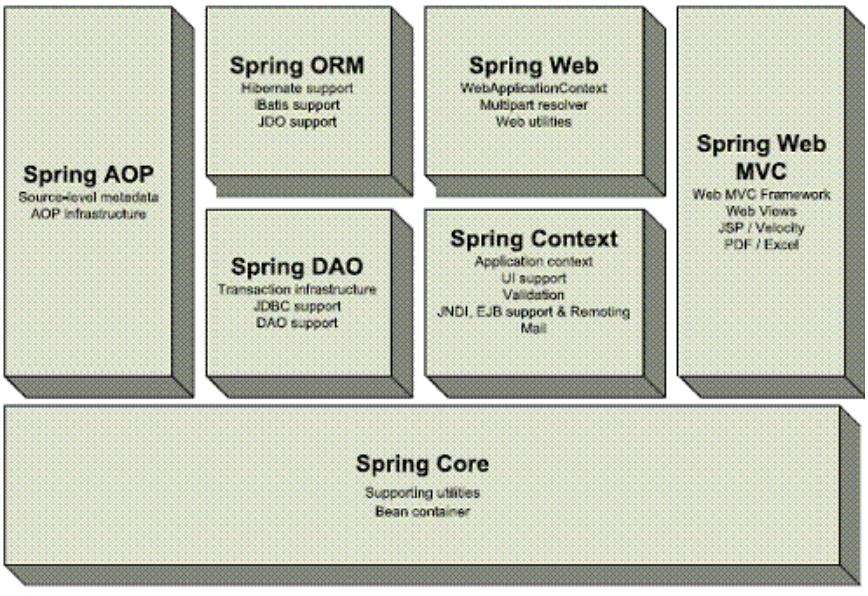


Per lo sviluppo di questa architettura, abbiamo utilizzato un *framework* di supporto chiamato *Spring Framework* - un *framework* open source che dà la possibilità di creare e gestire un progetto diviso in una struttura MVC, facilitando lo sviluppo attraverso della fornitura di innumerevole *library* e risorse che saranno spiegate in più dettaglio nella prossima sezione.

5.2. La struttura Spring Framework

Il Sistema è diviso in forma tale da seguire lo standard del Spring Framework. L'adozione di questo standard permette allo sviluppatore di concentrarsi sull'attuazione delle caratteristiche specifiche dell'applicazione, delegando al Framework, come lo Spring, il compito di specificare la dipendenza di alcuni oggetti. La struttura dello Spring è presentata nella figura 2.

Figura 2: Struttura Framework



Il modulo Spring Core rappresenta le caratteristiche principali dello Spring, in cui l'elemento principale è il *BeanFactory*; si tratta di una implementazione del pattern *Factory*, incaricato di

applicare *l'Inversion of Control,* che consente l'accoppiamento tra la configurazione e la specificazione delle dipendenze e la loro logica di programmazione.

Il modulo *Spring Context* dello Spring si trova sopra il pacchetto principale e fornisce un modo per accedere il *beans* con uno stile di struttura. Lo *Spring Context* fornisce il supporto per i servizi aziendali, come *Java Naming* and Directory Interface ™ (JNDI), Enterprise JavaBeans (EJB), e-mail, convalidazione, internazionalizzazione, propagazione di eventi, carico di risorse e creazione trasparente di contesti.

Il modulo Spring AOP è il modulo che permette l'implementazione di programmazione orientata a aspetti, che in questo progetto non usiamo, quindi non viene approfondita.

Il modulo Spring DAO fornisce un livello di astrazione per JDBC, eliminando gran parte del codice necessario per interagire con un database. Inoltre fornisce un modo semplice per gestire la manipolazione delle eccezioni e codici di errore emessi da diversi fornitori di database. Fornisce anche un modo per eseguire la gestione di transazioni programmatiche e dichiarative, non solo per le classi che implementano le interfacce speciali, ma per tutti gli oggetti Java (POJO). Il progetto Spring Framework consente l'integrazione diretta in alcuni dei popolari Api's ORM, come JDO, Hibernate e iBatis.

Il modulo ORM, tuttavia, prevede l'integrazione dello Spring con altri framework di persistenza di oggetti, come Hibernate e iBatis. Gli strumenti ORM consentono agli sviluppatori di realizzare il principio fondamentale di progettazione orientata agli oggetti: l'incapsulamento. Questo consente a un client di interagire con un oggetto senza conoscerne i dettagli di implementazione.

Per fornire funzionalità specifiche per progetti Web, esiste il modulo Spring Web. Sono fornite caratteristiche come i componenti per il caricamento dei file e il supporto per l'utilizzo di *Inversion of Control*. Fornisce anche altre caratteristiche, come l'inizializzazione dei contesti, utilizzando *listeners* di *servlet* ed un contesto di applicazioni orientata per il *Web*.

Il modulo Spring Web MVC fornisce un'implementazione del modello Model-View-Controller per le applicazioni Web. L'implementazione MVC non è solo un'implementazione comune: fornisce una separazione pulita tra il codice del modello, il dominio e i moduli web. Consente anche che tutte le altre caratteristiche dello Spring Framework, come la validazione, vengano utilizzate.

5.3. La divisione del Sistema

Il Sistema è suddiviso in pacchetti che seguono la struttura MVC dello Spring Framework:

1. **com.progetto.fumetteria.configuration**

   Qua sono realizzate le configurazioni base del Sistema.

   Nella classe AppConfig.java sono realizzati gli "*import*" dei file xml di impostazione; ad esempio: la configurazione *Hibernate*. In questa classe sono realizzate anche le configurazioni del tipo di *encoding*, luogo dei file (cartelle) ed anche i tipi di *views* che sono utilizzati.

Nella classe Initializer.java sono realizzate alcune impostazioni di inizializzazione dell'applicazione; ad esempio: l'impostazione del cammino iniziale dell'applicazione. Vengono realizzate anche le impostazioni dei tipi di "*annotation*", che il progetto utilizza. Vengono altresì definiti quali sono i *listeners* del Sistema.

2. com.progetto.fumetteria.controller

Il pacchetto controller è responsabile della manipolazione del cambiamento di informazione tra la *view* ed i servizi che sono a disposizione dell'applicazione. Il controller è anche responsabile del *mapping* dei cammini e del "reindirizzamento" degli utenti alle pagine richieste.

Le classi che appartengono a questo pacchetto sono:

2.1. AdminController: Questa è il controller che crea il cammino della pagina principale che contengono tutti i cammini per le funzioni disponibili solamente all'amministratore di sistema;

2.2. LoginController: Questo controller contiene le valutazioni degli utenti del Sistema; tanto un utente finale, quanto un utente amministratore;

2.3. AuthorController: Questo controller è responsabile delle azioni d'intermediazione tra le *view* ed il *model* della "Gestione degli autori";

2.4. ComicController: Questo controller è responsabile delle azioni d'intermediazione tra le *view* ed il *model* della "Gestione dei fumetti" (compreso anche il requisito [RFBO09.Gestione Stock]);

2.5. ErrorController: Questo controller dovrebbe gestire tutte le *exception* e fare un *redirect* ad una pagina di default per gli errori;

2.6. GenreController: Questo controller è responsabile delle azioni d'intermediazione tra la *view* ed il *model* per la "Gestione dei generi";

2.7. HomeController: Questo controller gestisce le funzionalità e i cammini alla "Pagina principale" [con lista (*menubar*) dei generi, lista dei fumetti, filtraggi fumetti, ecc.], del percorso *Front Office*;

2.8. NoteController: Questo controller è responsabile delle azioni d'intermediazione tra la *view* ed il *model* della "Gestione delle note" (commenti);

2.9. NoticeController: Questo controller è responsabile delle azioni d'intermediazione tra la *view* ed il *model* della "Creazione di avviso di disponibilità dei fumetti";

2.10. PublishingHouseController: Questo controller è responsabile delle azioni d'intermediazione tra la *view* ed il *model* della "Gestione di case editrice";

2.11. ReserveController: Questo controller è responsabile delle azioni d'intermediazione tra la *view* ed il *model* della "Gestione delle prenotazioni";

2.12. SuggestionController: Questo controller è responsabile delle azioni d'intermediazione tra la *view* ed il *model* della "Gestione dei suggerimenti";

2.13. UserController: Questo controller è responsabile delle azioni d'intermediazione tra la *view* ed il *model* della "Gestione di tutti i tipi di utenti";

2.14. ComicFrontController: Questo controller gestisce le funzionalità e cammini della pagina intermediaria "Visualizzare un singolo Fumetto", del percorso *Front Office*;

2.15. GenreFrontController: Questo controller gestisce le funzionalità e cammini della pagina intermediaria di "Visualizzazione dei fumetti per un determinato genere", del percorso *Front Office*;

2.16. ReserveFrontController: Questo controller gestisce le funzionalità e i cammini di "creazione di una prenotazione";

2.17. SuggestionFrontController: Questo controller gestisce le funzionalità e i cammini di "creazione di un suggerimento";

2.18. UserFrontController: Questo controller è responsabile delle azioni d'intermediazione tra la *view* ed il *model* della "gestione di utenti clienti" (utenti non amministratori).

3. com.progetto.fumetteria.dao

Il DAO è responsabile della comunicazioni tra l'applicazione e la base dei dati. In esso sono realizzate le connessioni con il data base: inserimenti, cancellazioni, ecc. Queste operazioni sono fatte utilizzando un *framework* ORM, denominato *Hibernate*; questo *framework* ha l'obbiettivo di facilitare l'interazione tra l'applicazione e la base di dati, in maniera tale che questo lavoro sia più trasparente al programmatore.

Le classi che appartengono al DAO sono:

3.1. GenericDao: Questa è la classe più importante del pacchetto DAO, perché questa è la classe astratta, che contiene tutta la logica per realizzare le chiamate operative del database: alcuni tipi diversi di filtraggio, inserimenti, cancellazioni ed aggiornamenti. Tutte le altre classi sono responsabili solo di estendere questa classe e passare il tipo di oggetto su cui si deve lavorare;

3.2. AuthorDao: Questa classe è responsabile del tipo "Autore";

3.3. ComicAuthorDao: Questa classe è responsabile del tipo "Relazione tra Autori e Fumetti";

3.4. ComicDao: Questa classe è responsabile del tipo "Fumetto";

3.5. GenreDao: Questa classe è responsabile del tipo "Genere";

3.6. NoteDao: Questa classe è responsabile del tipo "Commento" (nota);

3.7. PublishingHouseDao: Questa classe è responsabile del tipo "Casa Editrice";

3.8. UserDao: Questa classe è responsabile del tipo "Utente";

3.9. ReserveDao: Questa classe è responsabile del tipo "Prenotazione";

3.10. NoticeDao: Questa classe è responsabile del tipo "Notifica";

3.11. UserRoleDao: Questa classe è responsabile del tipo "Ruolo dell'utente";

3.12. PhoneDao: Questa classe è responsabile del tipo "Telefono".

4. com.progetto.fumetteria.model

Le classi "model" rappresentano un *mapping* oggetto-relazione della base di dati del Sistema. È utilizzato per il passaggio di informazioni tra i livelli del modello MVC (*Model-View-Controller*) ed anche è utilizzato per il *framework Hibernate* per fare il *mapping* della base dei dati.

Le classi che appartengono alla "model" sono:

4.1. Author: Questa classe contiene tutte le informazioni dell'autore; ad esempio: la tabella "*author*" della base dei dati contiene la colonna "*name*"; questa classe avrà un attributo "*name*", con i suoi rispettivi get/set;

4.2. ComicAuthor: Questa classe contiene tutte le informazioni di quello che riguarda la relazione tra le due classi "*Author*" e "*Comic*"; ad esempio: la tabella "*author*" è in relazione con la tabella "*comic*" della base dei dati, e loro contengono una classe intermediaria che gestisce il loro rapporto, quindi la classe *ComicAuthor* serve a realizzare il mapping di questa tabella intermediaria, come già spiegato nell'esempio precedente;

4.3. ComicAuthorId: Questa classe contiene l'insieme di attributi che compongono la chiave primarie della relazione comic – author, Poiché la chiave primaria della classe ComicAuthor è una chiave composta,;

4.4. Comic: Questa classe contiene tutte le informazioni del fumetto; ad esempio: la tabella "*comic*" della base dei dati contiene la colonna "*title*", quindi questa classe avrà un attributo "*title*", con i suoi rispettivi get/set;

4.5. Genre: Questa classe contiene tutte le informazioni relative al genere; ad esempio: la tabella "*genre*" della base dei dati contiene la colonna "*name*", quindi questa classe avrà un attributo "*name*", con i suoi rispettivi get/set;

4.6. Note: Questa classe contiene tutte le informazioni relative alle note (commenti); ad esempio: la tabella "*note*" della base dei dati contiene la colona "*status*", quindi questa classe avrà un attributo "*status*" , con i suoi rispettivi get/set;

4.7. PublishingHouse: Questa classe contiene tutte le informazioni della casa editrice; ad esempio: la tabella "*publishing_house*" della base dei dati contiene la colonna "*name*", quindi questa classe avrà un attributo "*name*", con i suoi rispettivi get/set;

4.8. User: Questa classe contiene tutte le informazioni dell'utente; ad esempio: la tabella "*user*" della base dei dati contiene la colonna "*email*", quindi questa classe avrà un attributo "*email*", con i suoi rispettivi get/set;

4.9. Reserve: Questa classe contiene tutte le informazioni relative alla prenotazione; ad esempio: la tabella "*reserve*" della base dei dati contiene la colonna "*total*", quindi questa classe avrà un attributo "*total*" , con i suoi rispettivi get/set;

4.10. Notice: Questa classe contiene tutte le informazioni relative all'avviso; ad esempio: la tabella "*notice*" della base dei dati contiene la colonna "*email*", quindi questa classe avrà un attributo "*email*", con i suoi rispettivi get/set;

4.11. Phone: Questa classe contiene tutte le informazioni relative al telefono che un utente può avere; ad esempio: la tabella "*phone*" della base dei dati contiene la colonna "*id_user*", che serve ad indicare che il telefono "*xxx xxxxxx*" ha come id l'utente "*x*", laddove un utente può avere più di un numero di telefono. Quindi, questa classe avrà un attributo "*idUser*" , con i suoi rispettivi get/set.

5. com.progetto.fumetteria.service

Nel pacchetto service ci sono tutti i servizi di cui il DAO ci dà la possibilità di usufruire. Questo modulo del Sistema è indipendente dalla programmazione realizzata per effettuare la connessione con la base dei dati. La sua utilità principale è "modularizzare" in maniera più indipendente i livelli del modello MVC; ad esempio: se un protocollo, come la comunicazione con la base dei dati, fosse cambiato, i livelli "Controller" e "View" non ne sarebbero

influenzati. (Considerando che il cambiamento effettuato utilizza lo stesso '*standard*' proposto);

Le classi che appartengono al pacchetto service sono:

5.1. AuthorService: Questa classe è responsabile dei servizi offerti dal tipo "Autore";

5.2. ComicAuthorService: Questa classe è responsabile dei servizi offerti dalla relazione tra i tipi "autori – fumetti";

5.3. ComicService: Questa classe è responsabile dei servizi offerti dal tipo "Fumetto";

5.4. GenericService: Questa classe è una classe astratta, con tutte le operazioni che vanno ad influenzare la base di dati. E' scritta in maniera più comprensibile al programmatore, utilizzando gli oggetti e le sue relazioni con altri oggetti, piuttosto che il linguaggio per *database*. Le altre classi devono solo estendere questa e passare come parametro il suo tipo di oggetto;

5.5. GenreService: Questa classe è responsabile dei servizi offerti del tipo "Genere";

5.6. NoteService: Questa classe è responsabile dei servizi offerti del tipo "Commento" (nota);

5.7. PublishingHouseService: Questa classe è responsabile dei servizi offerti dal tipo "casa editrice";

5.8. UserService: Questa classe è responsabile dei servizi offerti del tipo "Utente;

5.9. ReserveService: Questa classe è responsabile dei servizi offerti dal tipo "Prenotazione";

5.10. NoticeService: Questa classe è responsabile dei servizi offerti dal tipo "Avviso";

5.11. PhoneService: Questa classe è responsabile dei servizi offerti dal tipo "Telefono".

6. com.progetto.fumetteria.util

In questo pacchetto ci sono alcune classi per l'ausilio alla programmazione, ad esempio la definizione dei formati numerici, le classi con valori statici, ecc...

7. src.main.webapp.WEB-INF.jsp

Questo è il pacchetto dove ci sono le schermate del Sistema: suddiviso in *backoffice* e *frontoffice*.

6. LE TECNOLOGIE

6.1. Le tecnologie web – client side

Le tecnologie di base sono le tecnologie principali ed essenziali per creare un sistema web.

1. HTML – l'*HTML* è un abbreviazione per l'espressione inglese *Hyper Text Markup Language*, che significa "Linguaggio di formattazione di ipertesto". E' un linguaggio utilizzato per produrre pagine web, perché documenti scritti in html possono essere interpretati per i browser;
2. CSS – *Cascading Style Sheets*, oppure CSS, è un linguaggio di fogli di stile, utilizzato per definire la presentazione di documenti scritti in un linguaggio di formattazione, come l'HTML. Il suo principale beneficio è promuovere la separazione tra il formato e il contenuto di un documento: invece di mettere la formattazione nel documento, lo sviluppatore crea un *link* che punta alla pagina che

contiene lo stile. Quindi quando si vuole modificare il layout del sito, basta modificare il file del link. Il progetto è basato anche su tecnologia *CSS3*, che contiene delle risorse in più delle precedenti. *Mozilla Firefox* e *Google Chrome*, sono i browser che hanno maggiore supporto per questa tecnologia, infatti i test vengono eseguiti su questi browser;

3. JavaScript – *Javascript* è un linguaggio di programmazione *client side*. E' utilizzato per controllare l'HTML ed il CSS, per manipolare i comportamenti nella pagina, senza necessariamente passare per il server, controllando il browser, realizzando comunicazione asincrona e cambiando il contenuto del documento. E' stato concepito per essere un linguaggio di script *orientato agli oggetti* basato in prototipi, debolmente tipato e dinamico con funzioni di prima classe;
4. jQuery – *Jquery* è una library *javascript cross-browser*, sviluppata per semplificare gli script client-side che interagiscono con l'HTML. E' opensource e la sintassi è stata sviluppata per semplificare la navigazione del documento HTML; permette la selezione di elementi *DOM* (*Document Object Model* – rappresentazione e interazione con oggetti in documenti di formattazione), creazione di animazioni, manipolazione di eventi e sviluppo di applicazioni *AJAX*; questa library semplifica lo sviluppo di applicazioni web dinamiche di grande complessità;
5. Bootstrap – *Bootstrap* è una raccolta di strumenti liberi per la creazione di siti e applicazioni per il Web. Questa contiene alcuni modelli di progettazione basati su HTML e CSS, sia per la tipografia, che per le componenti dell'interfaccia, come moduli, bottoni e navigazione, e altri componenti dell'interfaccia, così come alcune estensioni opzionali di *JavaScript*;
6. Ajax – *Ajax* è una metodologia che utilizza altre tecnologie, come ad ed esempio XML e JavaScript per fare richieste asincrone al server; con le informazioni ritornate dal server cambia la pagina già caricata utilizzando DOM, per aggiornarla senza ricaricare nuovamente tutto il suo contenuto. Questa pratica riduce il traffico e migliora l'esperienza dell'utente.

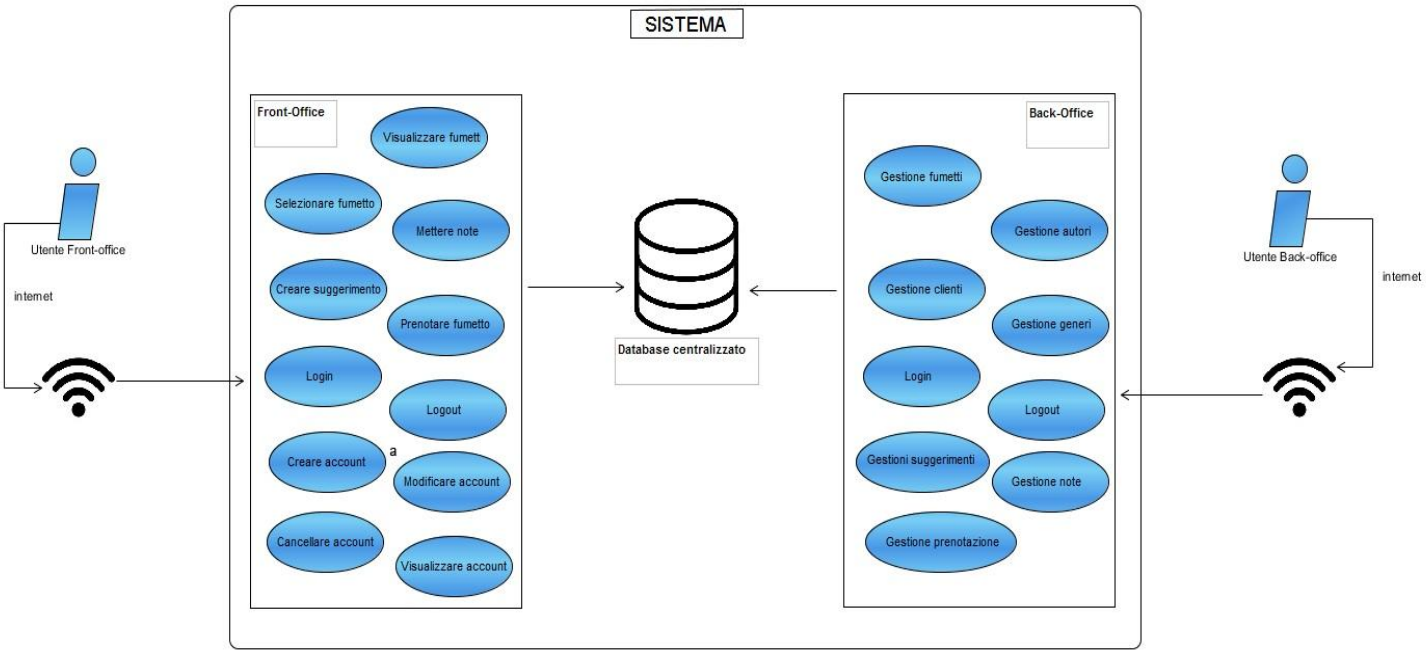
6.2. Le tecnologie base – server side

1. Java – *Java* è un linguaggio di programmazione orientato agli oggetti. A differenza dei linguaggi convenzionali, che sono compilati in codice nativo, il linguaggio *Java* viene compilato in un *bytecode* che viene eseguito (interpretato) in una macchina virtuale;
2. Servlet – I *Servlet* sono classi Java usate per estendere le funzionalità di un Server, sviluppate secondo una struttura ben definita che, quando installate e configurate su un *Server* che abbia l'implementazione di un *Servlet Container*, sono in grado di gestire le richieste ricevute dai *client Web*, come, ad esempio, i *browsers*;
3. MySql – *MySql* è un sistema per gestire le basi dei dati (SGBD), che utilizza il linguaggio *SQL* (*Structured Query Language*) come interfaccia. E' un sistema *open source*;
4. JSP – *Java Server Pages* (JSP) è una tecnologia che aiuta gli sviluppatori a creare pagine web dinamiche basate su HTML, XML o altri tipi di documenti. Per eseguire *JavaServer Pages*, è necessario un web server compatibile con un Servlet Container, ad esempio *Apache Tomcat*;
5. Apache Tomcat - *Tomcat* è un web server *Java*, più precisamente, un servlet container. *Tomcat* permette l'implementazione delle tecnologie *Java Servlet* e *JavaServer Pages* (JSP); come *web* server, fornisce un server Web HTTP puramente scritto in Java;
6. Hibernate - *Hibernate* è un *framework* per il *mapping object-relational*, scritto nel linguaggio *Java*. *Hibernate* facilita la mappatura degli attributi tra una base dati tradizionale ed il modello degli oggetti di un'applicazione, utilizzando file (XML) o annotation *Java*.

7. MODULI DEL SISTEMA

7.1. Diagramma di contesto

Figura 3: Diagramma di Contesto



7.2. Front-Office

Il modulo front-office del Sistema ha lo scopo di raggruppare le operazione che sono nell'interesse dell'utente finale (del tipo '*customer*'); ad esempio: fare la presentazione dei fumetti registrati nel Sistema, oppure eseguire una prenotazione di un determinato fumetto; La lista di tutte le possibili operazioni del modulo front-office, si incontra nel documento "Analisi dei Requisiti 4.4".

7.3. Back-Office

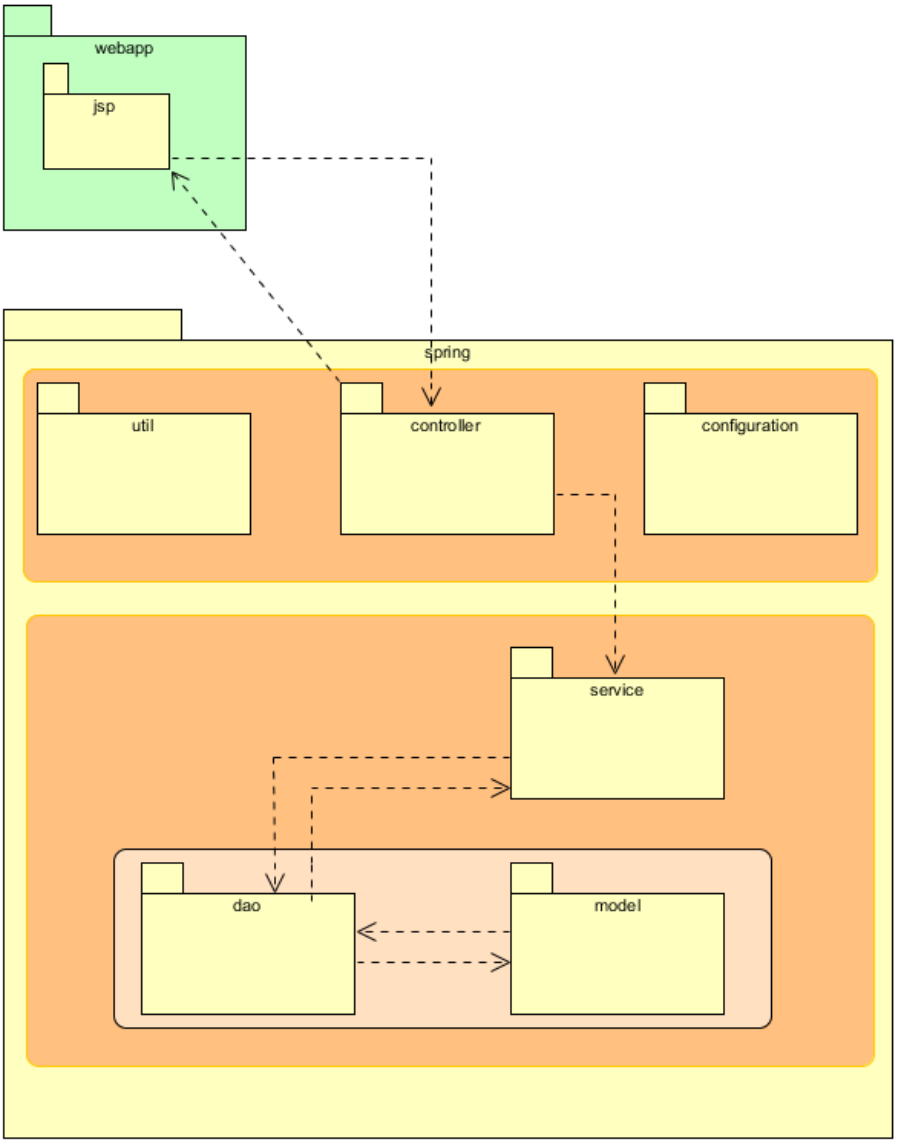
Il modulo back-office del Sistema ha lo scopo di raggruppare le operazione che un utente del tipo 'amministratore' può eseguire, ad esempio fare l'inserimento di un nuovo fumetto oppure cambiare lo *status* di una prenotazione. Il back-office del sistema funziona come la parte amministrativa del negozio, cioè, tutto ciò che serve per gestire il negozio.

7.4. Database centralizzato

Il database si occupa di salvare le informazioni utili al sistema; è centralizzato, e per quello garantisce che sia possibile gestire il sistema da diversi computer avendo sempre l'accesso agli stessi dati.

7.5. Diagramma della suddivisione del progetto

Figura 1 Suddivisione del progetto



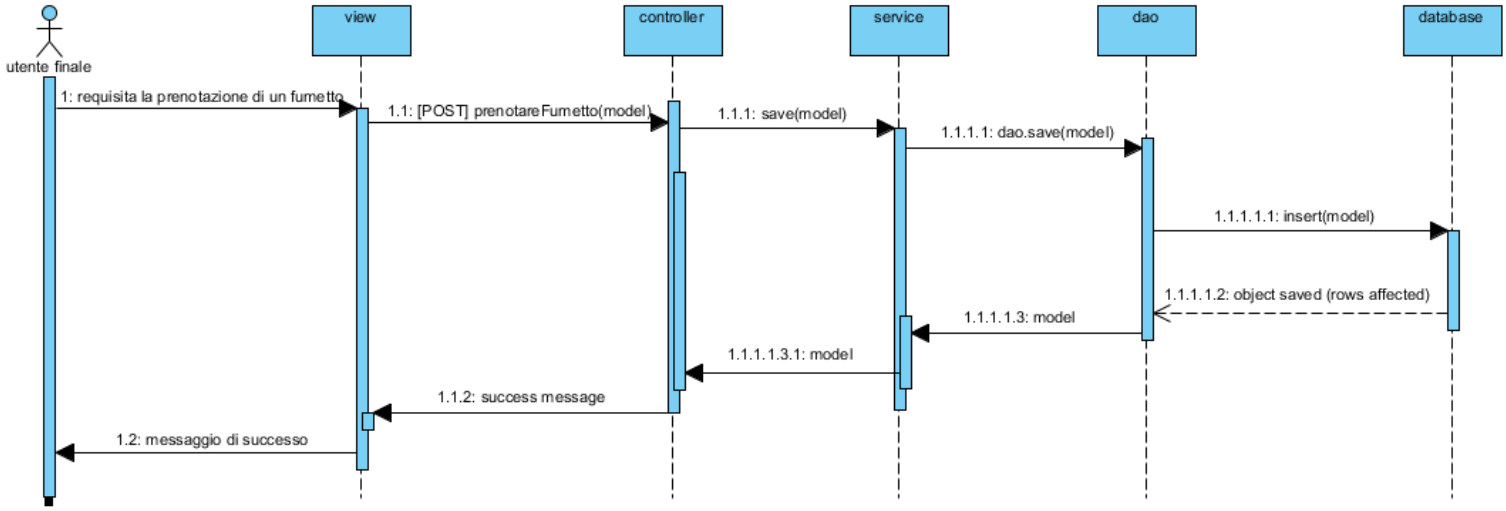
8. DINAMICA DEL SISTEMA

8.1. *Flussi principali*

In questa sezione saranno presentati come esempio due dei principali digrammi di sequenza del Sistema: “Prenotare” e “Accesso all’area amministrativa”.

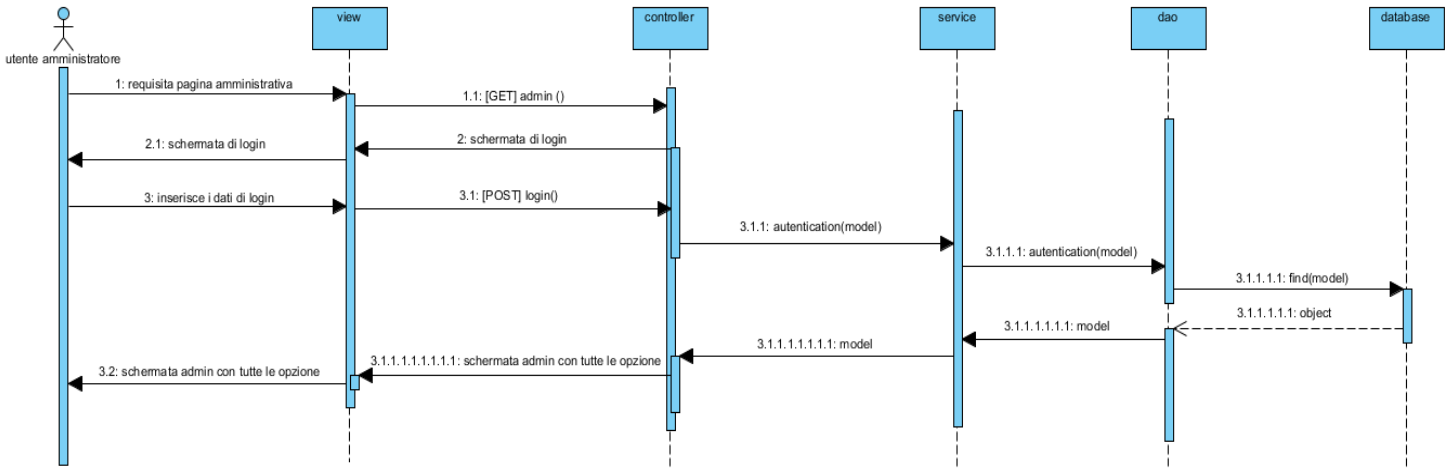
Nel diagramma “Prenotare”, abbiamo considerato come pre-requisito che l’utente finale sia loggato nel Sistema.

Figura 5: Diagramma di sequenza - prenotare



Il diagramma “acesso all’area amministrativa” mostra il flusso di requisizione dell’area amministrativa del Sistema e come viene eseguito il login.

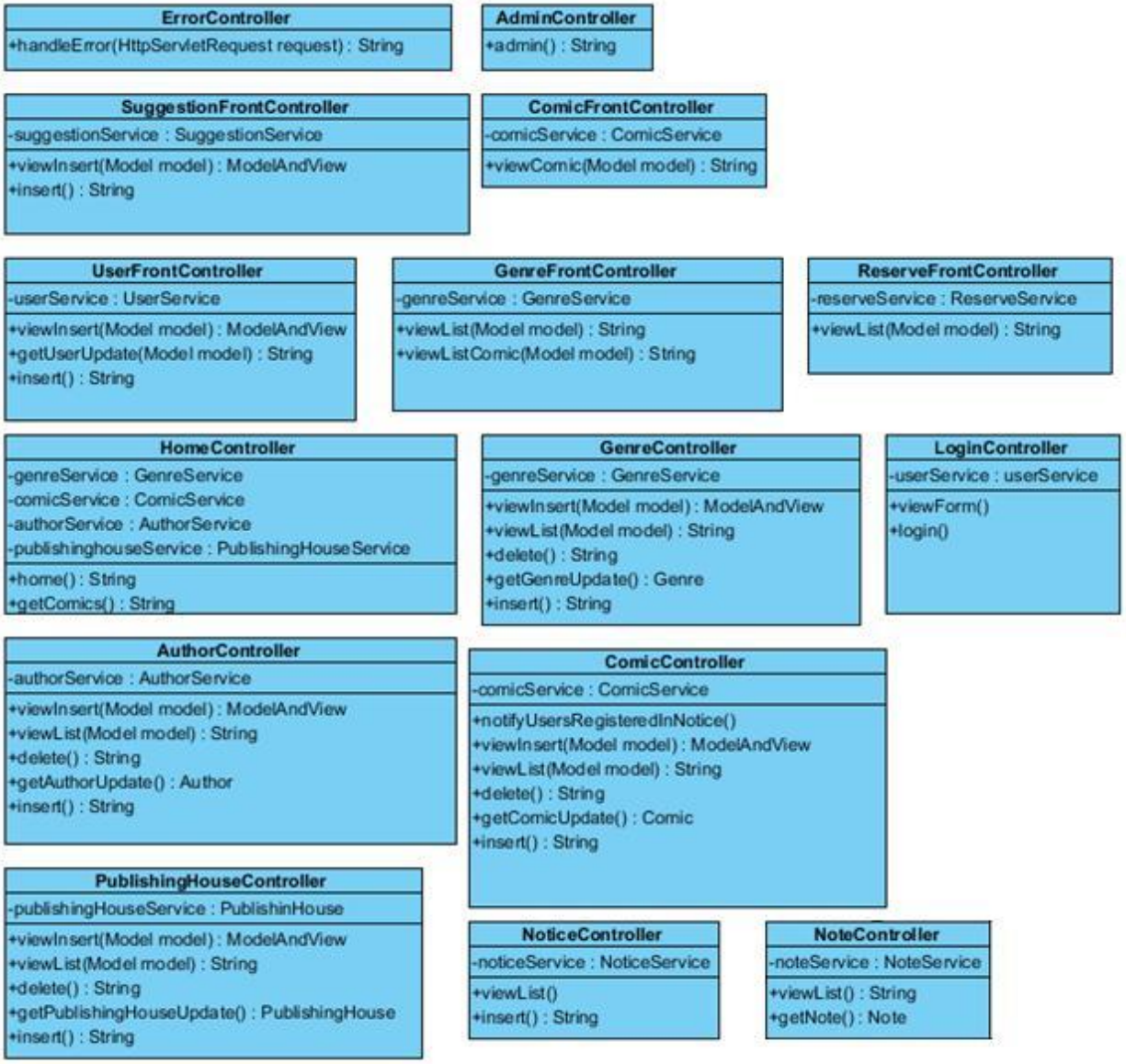
Figura 6: Diagramma di sequenza - Accesso all'area Amministrativa



9. DIAGRAMMA DELLE CLASSI

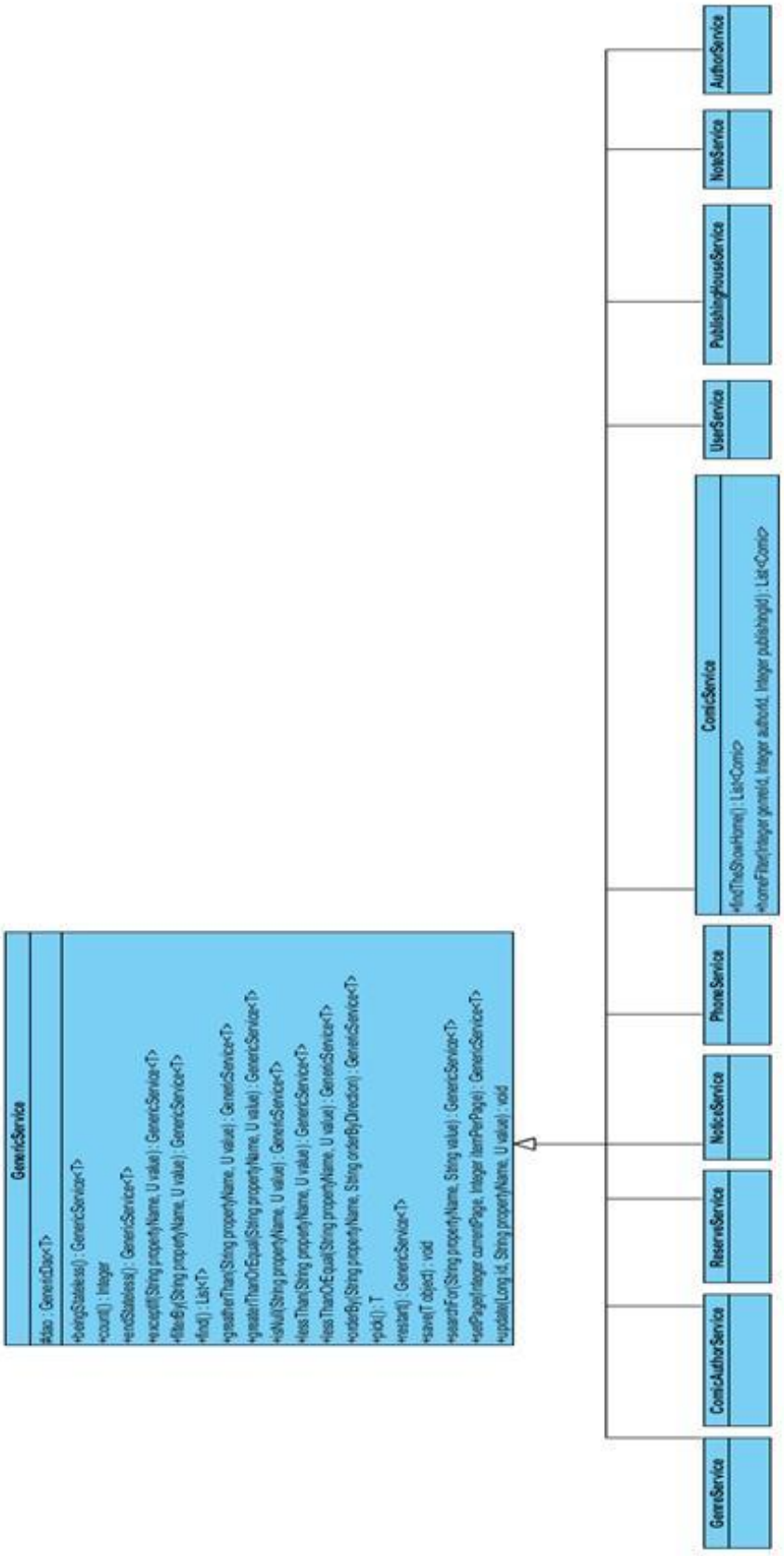
9.1. Pacchetto controller

Figura 2:Classi del Pacchetto Controller



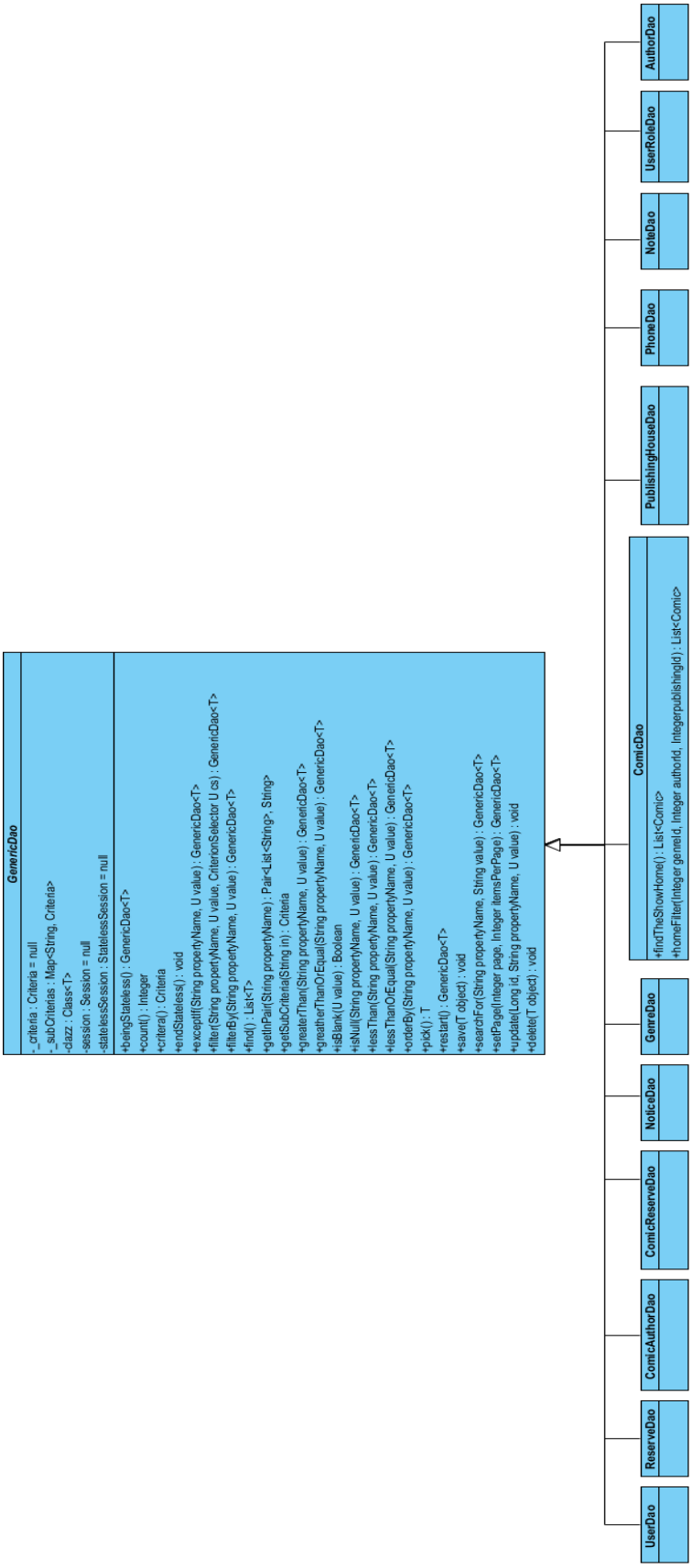
9.2. Pacchetto Service

Figura 8:Classi del Pacchetto Service



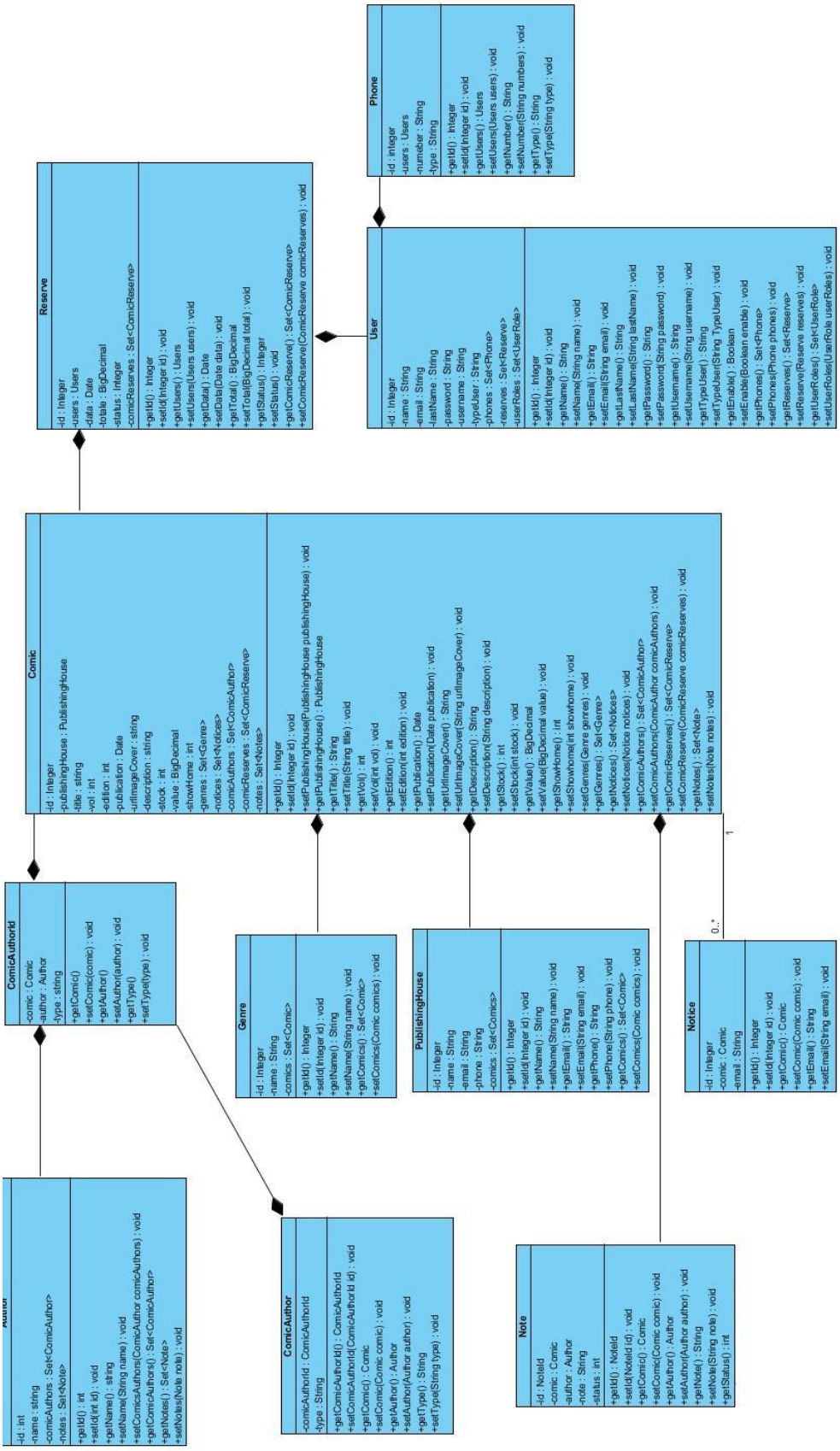
9.3. Pacchetto DAO

Figura 9:Classi del Pacchetto DAO



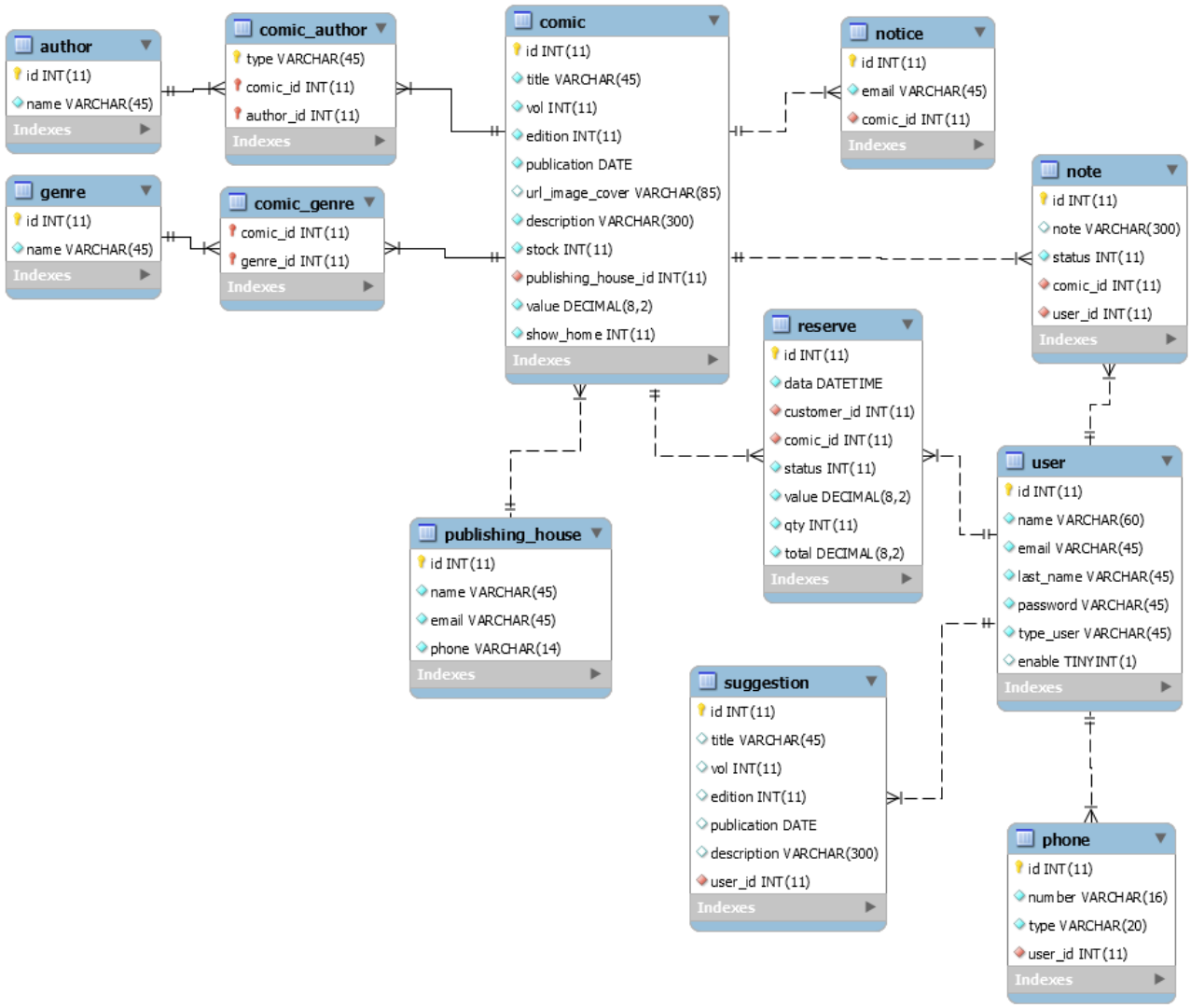
9.4. Pacchetto Model

Figura 10:Classi del Pacchetto Model



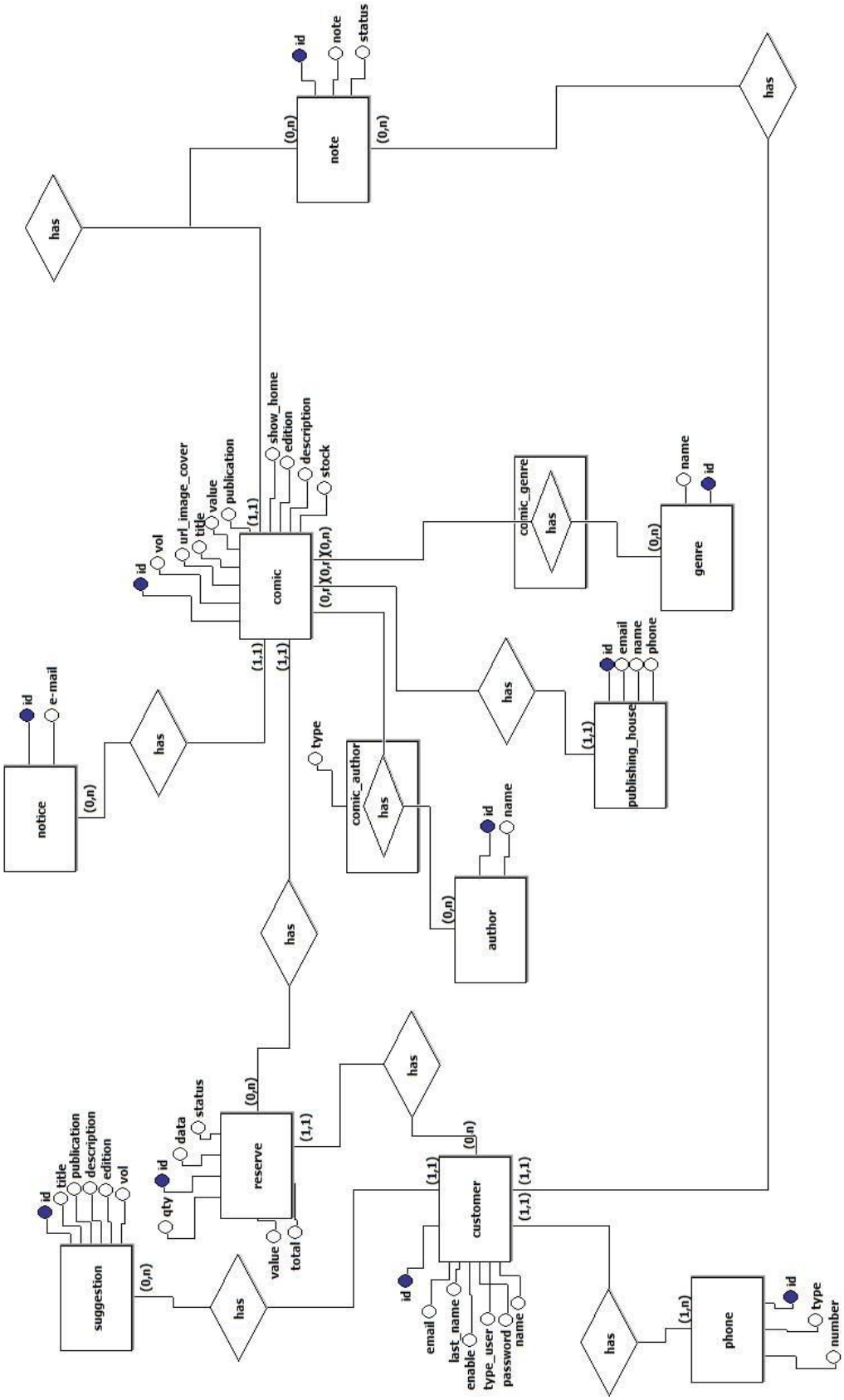
9.5. Diagramma entita' relazione (DER) e ER

Figura 11 Diagramma entità relazione (DER)



9.6. Diagramma entita' relazione (DER) e ER

Figura 12 Modello entità relazione



10. DIZIONARIO DEI DATI

10.1. Tabella author

Attributo	Tipo	Descrizione	Chiave Primaria	Chiave Esterna	*Not Null*	*Unique*
Id	INT(11)	Numero identificatore	Si		Si	Si
Name	VARCHAR(45)	Nome dell'autore	No		Si	No

10.2. Tabella comic_author

Attributo	Tipo	Descrizione	Chiave Primaria	Chiave Esterna	*Not Null*	*Unique*
Type	VARCHAR(45)	Tipo dell'authore	Si		Si	No
Author_id	INT(11)	Identificatore dell'autore	Si	Si	Si	No
Comic_id	INT(11)	Identificatore del fumetto	Si	Si	Si	No

10.3. Tabella comic

Attributo	Tipo	Descrizione	Chiave Primaria	Chiave Esterna	*Not Null*	*Unique*
Id	INT(11)	Identificatore dell'autore	Si		Si	Si
Title	VARCHAR(45)	Titolo	No		Si	No
Vol	INT(11)	Volume	No		Si	No
Edition	INT(11)	Num. Edizione	No		Si	No
Publication	DATE	Data della publicazione	No		Si	No
Url_image_cover	VARCHAR(45)	Url dell' immagine di copertina	No		No	No
Description	VARCHAR(300)	Descrizione	No		Si	No
Stock	INT(11)	Quantità di fumetti presenti	No		Si	No
Publishing_house_id	INT(11)	Chiave esterna - casa editrice	No	Si	Si	No
Value	DECIMAL(8,2)	Valore del fumetto	No		Si	No
Show_home	INT(11)	Flag per sapere se è prodotto di vitrine	No		Si	No

10.4. Tabella genre

Attributo	Tipo	Descrizione	Chiave Primaria	Chiave Esterna	*Not Null*	*Unique*
Id	INT(11)	Numero identificatore	Si		Si	Si
Name	VARCHAR(45)	Nome del genere	No		Si	Si

10.5. Tabella comic_genre

Attributo	Tipo	Descrizione	Chiave Primaria	Chiave Esterna	Not Null	Unique
Genre_id	INT(11)	Identificatore del genere	Si	Si	Si	No
Comic_id	INT(11)	Identificatore del fumetto	Si	Si	Si	No

10.6. Tabella notice

Attributo	Tipo	Descrizione	Chiave Primaria	Chiave Esterna	Not Null	Unique
Id	INT(11)	Numero identificatore	Si		Si	Si
Email	VARCHAR(45)	Email dell'utente	No		Si	No
Comic_id	INT(11)	Chiave esterna - fumetto	No	Si	Si	No

10.7. Tabella reserve

Attributo	Tipo	Descrizione	Chiave Primaria	Chiave Esterna	Not Null	Unique
Id	INT(11)	Numero identificatore	Si		Si	Si
Data	DATETIME	Data della prenotazione	No		Si	No
Total	DECIMAL(8,2)	Valore della prenotazione	No		Si	No
Customer_id	INT(11)	Chiave esterna dell'utente	No	Si	Si	No
Comic_id	INT(11)	Numero identificatore del fumetto	No	Si	Si	No
Status	INT(11)	Stato della prenotazione	No		Si	No
Value	DECIMAL(8,2)	Valore totale della prenotazione	No		Si	No
Qty	INT(11)	Quantità di fumetti della prenotazione	No		Si	No

10.8. Tabella publishing_house

Attributo	Tipo	Descrizione	Chiave Primaria	Chiave Esterna	Not Null	Unique
Id	INT(11)	Numero identificatore	Si		Si	Si
Name	VARCHAR(45)	Nome della casa editrice	No		Si	No
Email	VARCHAR(45)	Email della casa editrice	No		Si	Si
Phone	VARCHAR(14)	Telefono della casa editrice	No		Si	Si

10.9. Tabella user

Attributo	Tipo	Descrizione	Chiave Primaria	Chiave Esterna	Not Null	Unique
Id	INT(11)	Numero identificatore	Si		Si	Si
Name	VARCHAR(45)	Nome dell'utente	No		Si	No
Email	VARCHAR(45)	Email dell'utente	No		Si	Si
Last_name	VARCHAR(45)	Cognome dell'utente	No		Si	No
Password	VARCHAR(45)	Password dell'utente	No		Si	No
Type_user	VARCHAR(45)	Tipo di utente	No		Si	No
Enable	TYNYINT	L'utente è attivo o no	No		Si	No

10.10. Tabella phone

Attributo	Tipo	Descrizione	Chiave Primaria	Chiave Esterna	Not Null	Unique
Id	INT(11)	Numero identificatore	Si		Si	Si
Number	VARCHAR(16)	Numero del telefono	No		Si	No
Type	VARCHAR(20)	Tipo di telefono	No		Si	No
User_id	INT(11)	Chiave esterna – utente	No	Si	Si	No

10.11. Tabella note

Attributo	Tipo	Descrizione	Chiave Primaria	Chiave Esterna	Not Null	Unique
Id	INT(11)	Numero identificatore	Si		Si	Si
Note	VARCHAR(300)	Commentario dell'utente	No		Si	No
Status	INT(11)	Status (approvato o no)	No		Si	No
Comic_id	INT(11)	Chiave esterna – comic	No		Si	No
User_id	INT(11)	Chiave esterna – user	No		Si	No

10.12. Tabella suggestion

Attributo	Tipo	Descrizione	Chiave Primaria	Chiave Esterna	Not Null	Unique
Id	INT(11)	Identificatore dell'autore	Si		Si	Si
Title	VARCHAR(45)	Titolo	No		Si	No
Vol	INT(11)	Volume	No		Si	No
Edition	INT(11)	Num. Edizione	No		Si	No
Publication	DATE	Data della publicazione	No		Si	No
Description	VARCHAR(300)	Descrizione	No		Si	No
User_id	INT(11)	Chiave esterna – utente	No	Si	Si	No