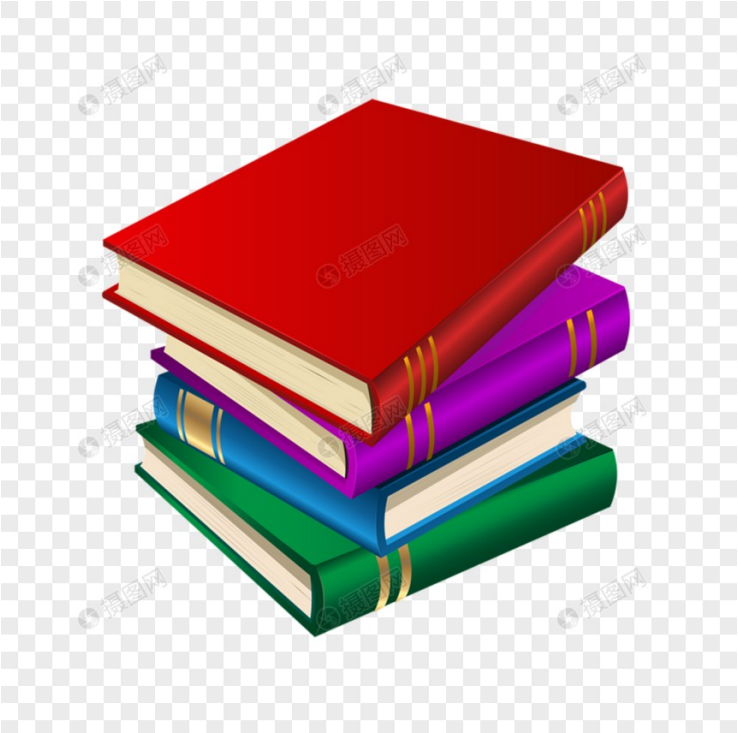
Book shop online

Daniele Cinus 0219066 Ingegneria informatica



index

Intro

Sistema di gestione, vendita e acquisto libri, digitali e cartacei.

Idea : Creazione di una libreria online

Attori : Amministratore , Editore, Scrittore, Cliente, Negoziante

Aim of the system

Lo scopo del nostro progetto è creare un sistema che consenta ad una libreria fisica, di prevedere l'acquisto online dei libri e di avere un resoconto delle vendite e dell'utenza che vi acquista, dividendo però gli acquisti online da quelli del negozio

Overview of the system

L'utente dopo essersi registrato e/o loggato, ha accesso ad un catalogo digitale di libri,riviste e giornali, dove può eseguire una ricerca basandosi su vari fattori,come autore, genere, titolo, etc…Dopo aver scelto il libro d'interesse può acquistarlo, l'acquisto del libro può riguardare due formati, digitale o cartaceo col ritiro in negozio. Scelto il formato di acquisto l'utente può recarsi alla libreria per ritirare il prodotto acquistato, o eseguire il download del libro, se ha scelto il formato digitale. Un Utente che vuole pubblicare un suo libro può richiedere questo permesso al sistema, una volta che l'amministratore lo ha abilitato può anche lui pubblicare un suo manoscritto senza bisogno di una casa editrice, tale manoscritto sarà disponibile solo in forma digitale. Un editore può controllare i libri della sua casa editrice e fornire al sito un'eventuale versione cartacea del libro, l'editore può gestire i libri da esso pubblicati, e avere un piccolo overview degli scrittori a lui associati. Un amministratore può avere un resoconto delle vendite così da osservare l'andamento del sito, poi in aggiunta gestisce il catalogo di libri,giornali e riviste presenti sul sito.

Hw and Sw

Il software per essere eseguito ha bisogno di programmi esterni : MYSQL per gestire la base di dati, un driver che permette l'interazione (MYSQL Driver) ed un programma per leggere i file .pdf , come per ad esempio AcrobatReader.

Pros and Cons

PRO:

- Il sistema permette di pagare anche con i contanti.
- A parità di articolo, il tempo di consegna e/o ritiro è inferiore , visto che il negozio/magazzino è vicino.
- Il formato è più compatibile , in quanto l'utente può consultare copia cartacea.

CONS:

- Elenco degli oggetti è limitato.
- Non sussiste il servizio di corriere.
- Magazzino rifornito meno frequentemente

User Stories

- Come amministratore voglio poter consultare
  - vendite totali
  - guadagno
  - resoconto

cosi da poter avere un resoconto totale

- come amministratore accedo a :
  - capienza catalogo, lista prodotti trattati, giacenza globale e giacenza singolo prodotto

cosi posso sapere quando rifornire il magazzino

- come editore voglio poter:
  - vedere i libri associati alla sua casa editrice

- vedere gli scrittori associati alla mia casa editrice

cosi posso avere un resoconto dei progressi

- come scrittore voglio poter:
  - pubblicare libri con o senza editore

cosi da aggiornare il catalogo

- come gestore del negozio voglio poter:
  - ordinare i libri online per conto del cliente
  - eventuale offerta di ritiro del prodotto in negozio
  - rifornire la scorta in magazzino

cosi' da fornire un supporto alla vendita

- come cliente voglio poter :
  - acquistare libri in versione cartacea o digitale

cosi' da permettere di aumentare il numero dei oggetti acquistabili

Functional Requirements

- Il sistema deve fornire un metodo di ritiro in negozio dei libri cartacei
- Il sistema deve fornire un file di testo o una schermata di riepilogo contenente il numero di vendite:
  - globali
  - cartacee (ritiri in negozio)
  - digitali
- Il sistema deve fornire un file di resoconto di :
  - Totale utenti registrati
  - Fatturato (venduto)
  - Pubblicazioni (prodotti presenti)
- Il sistema deve fornire le giacenze globali o dei singoli prodotti (libri, riviste o giornali) riassunte in una schermata o file di testo
- Il sistema deve prevedere un metodo per la pubblicazione, modifica e eliminazione dei libri
- Il sistema deve prevedere un metodo di categorizzazione dei prodotti pubblicati
- Il sistema deve fornire all'editore un metodo per avere un resoconto sulla casa editrice ad esso associata* (libri e autori)
- Il sistema deve fornire un metodo per la registrazione e l'accesso di nuovi e vecchi utenti

- Il sistema deve fornire agli utenti un sistema di filtraggio dei libri per gli attributi* di quest'ultimo (tasto cerca)
- Il sistema deve fornire un elenco dei negozio in cui effettuare il ritiro del prodotto

Dictionary Functional requirements :

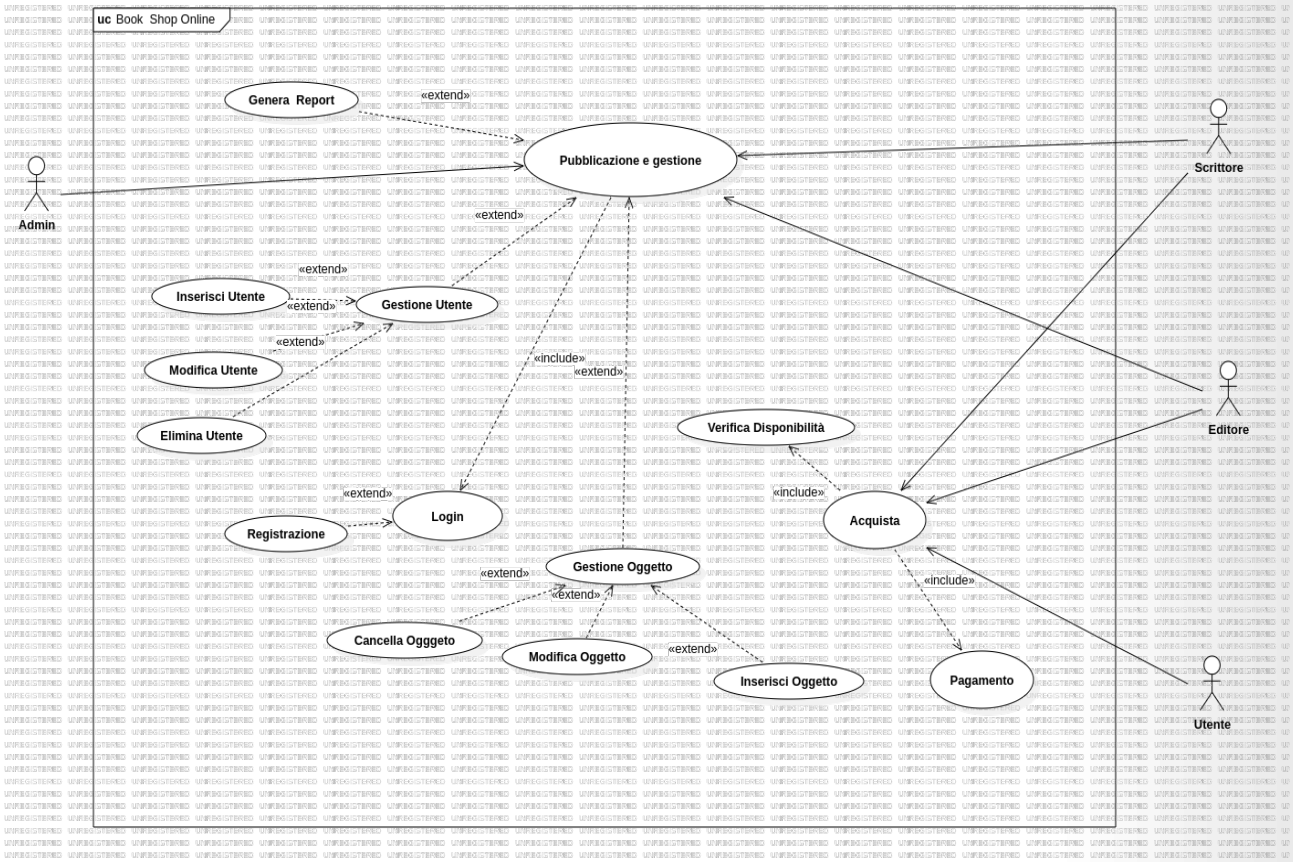
- Fatturato: Si intende l'utile guadagnato, nel periodo di tempo selezionato
- Prodotti : ci si riferisce sempre a libri, riviste e giornali
- Casa editrice ad esso associata : insieme dei libri pubblicati dall'editore e insieme degli scrittori che pubblicano con essa e nome casa editrice

Attributi : insieme delle seguenti caratteristiche:

- numero di pagine
- codice isbn
- editore
- autore
- lingua
- categoria
- prezzo
- recensioni
- numero copie
- titolo
- data pubblicazione
- breve descrizione
- disponibilita

Use Case

Overview



Internal step

Acquista oggetto

1. L’utente sceglie l’oggetto Libro dal catalogo
2. L’utente ottiene la lista dei libri nel catalogo
3. L’utente sceglie l’oggetto da comperare
4. L’utente sceglie la quantità da acquistare ed il sistema mostra l’importo
5. Il sistema , permette di pagare contanti
6. L’utente sceglie di ottenere la copia locale, scaricandola sul suo dispositivo

Estensioni:

1.a L’utente può scegliere l’oggetto Rivista

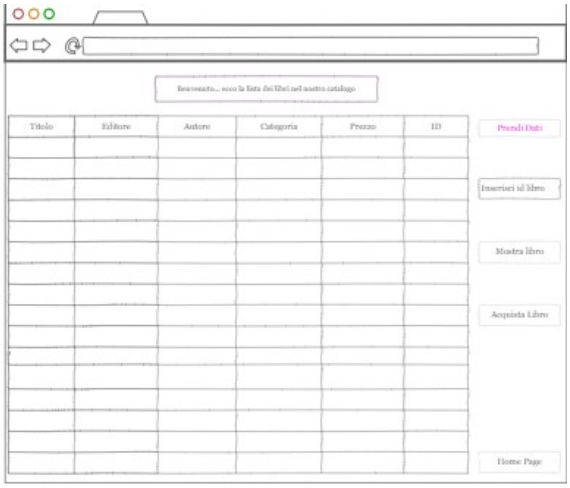
1.b L’utente può scegliere L’oggetto Giornale

5a Il sistema permette di poter pagare con carta credito

6.a Il sistema permette di ritirare la copia presso uno dei negozi abilitati

Storyboards

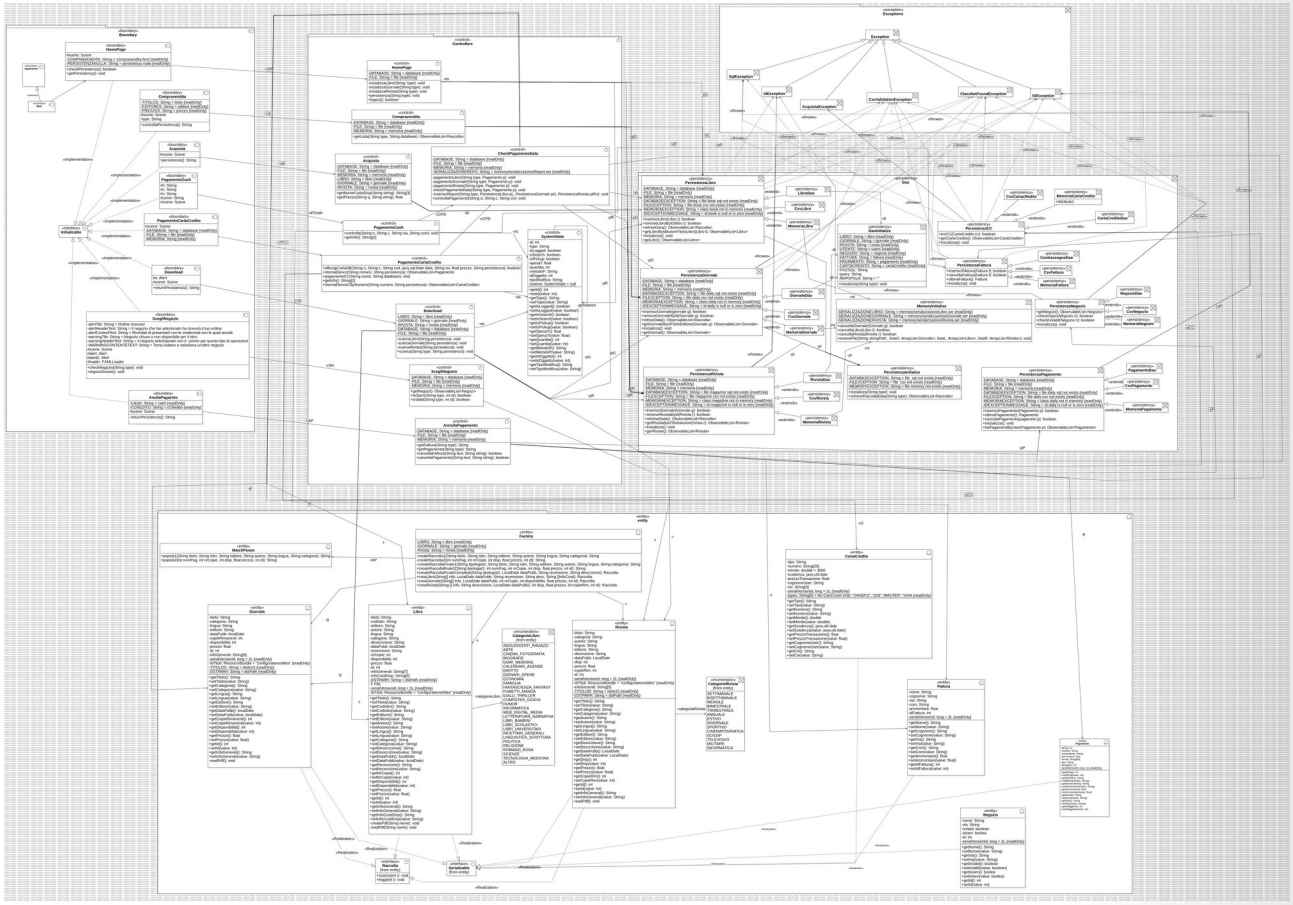
La prima storyboard mostra graficamente la functional requirements n.4, ovvero mostra le giacenze globali dei prodotti nel catalogo, sia nel caso di libri , giornali o riviste.



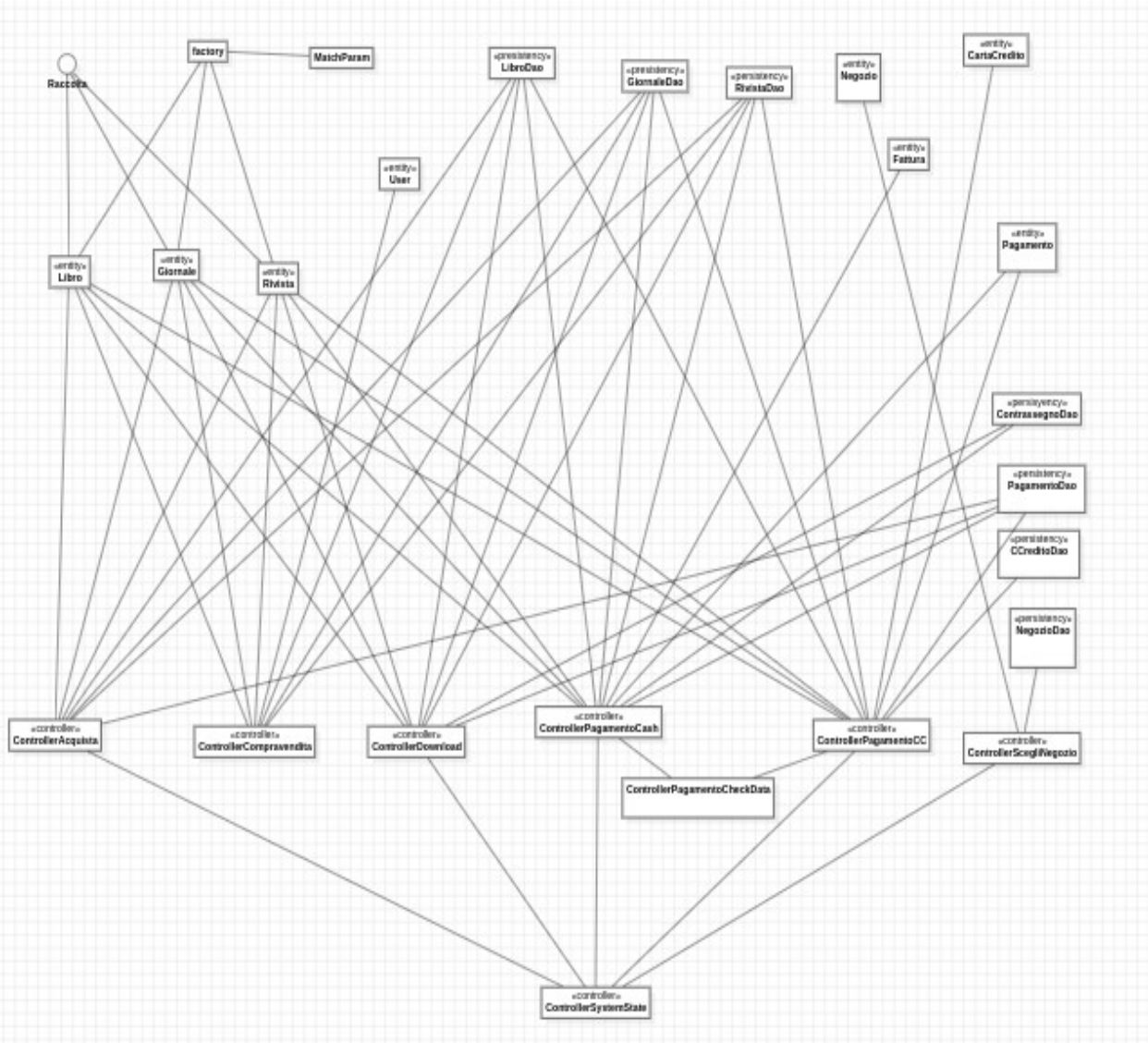
La seconda , invece mostra la storyboard n.10, ovvero la storyboard nella quale il sistema permette all'utente di scegliere il negozio in cui effettuare il ritiro.

Design

VOPC Acquista oggetto

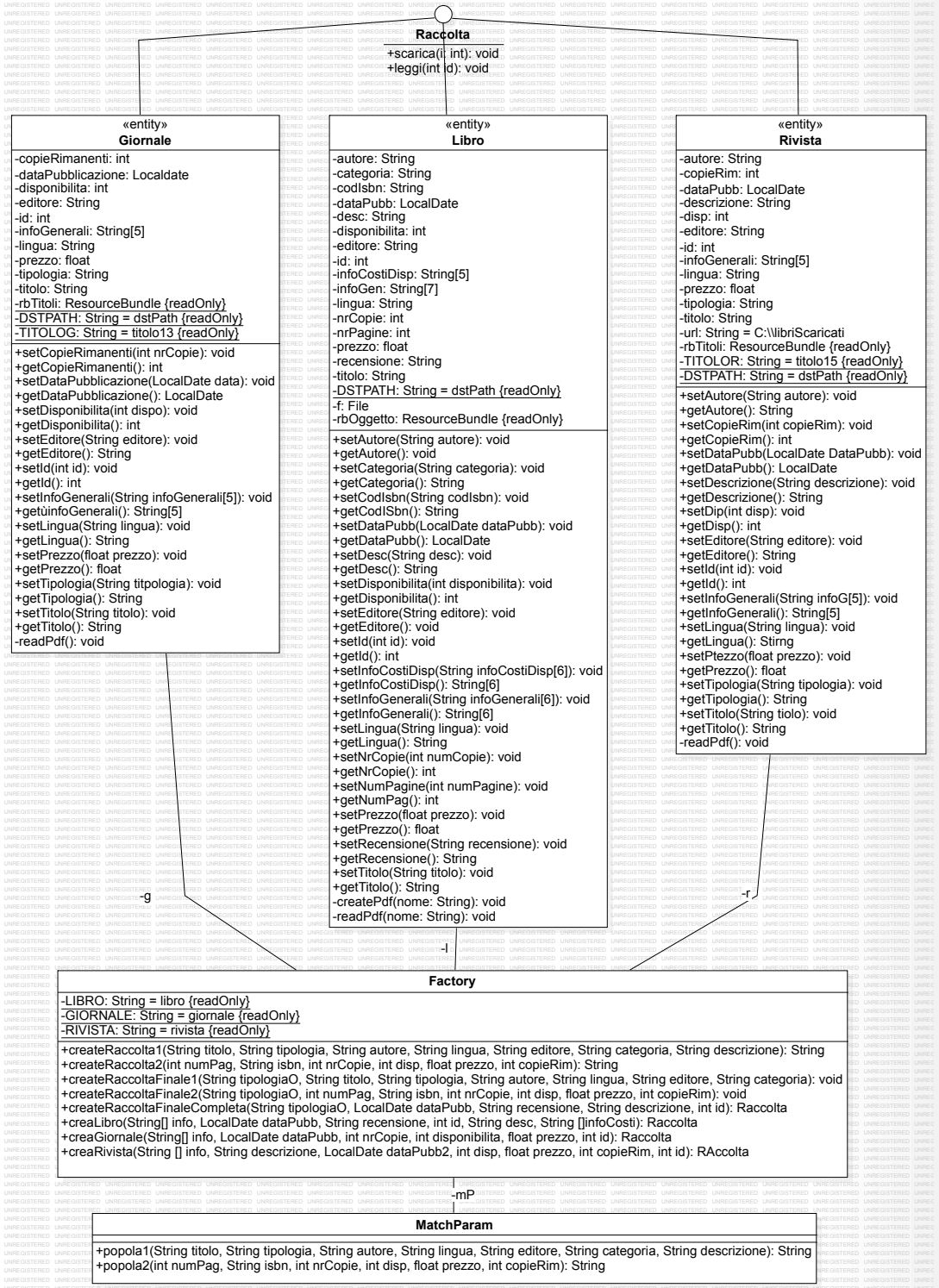


Design level

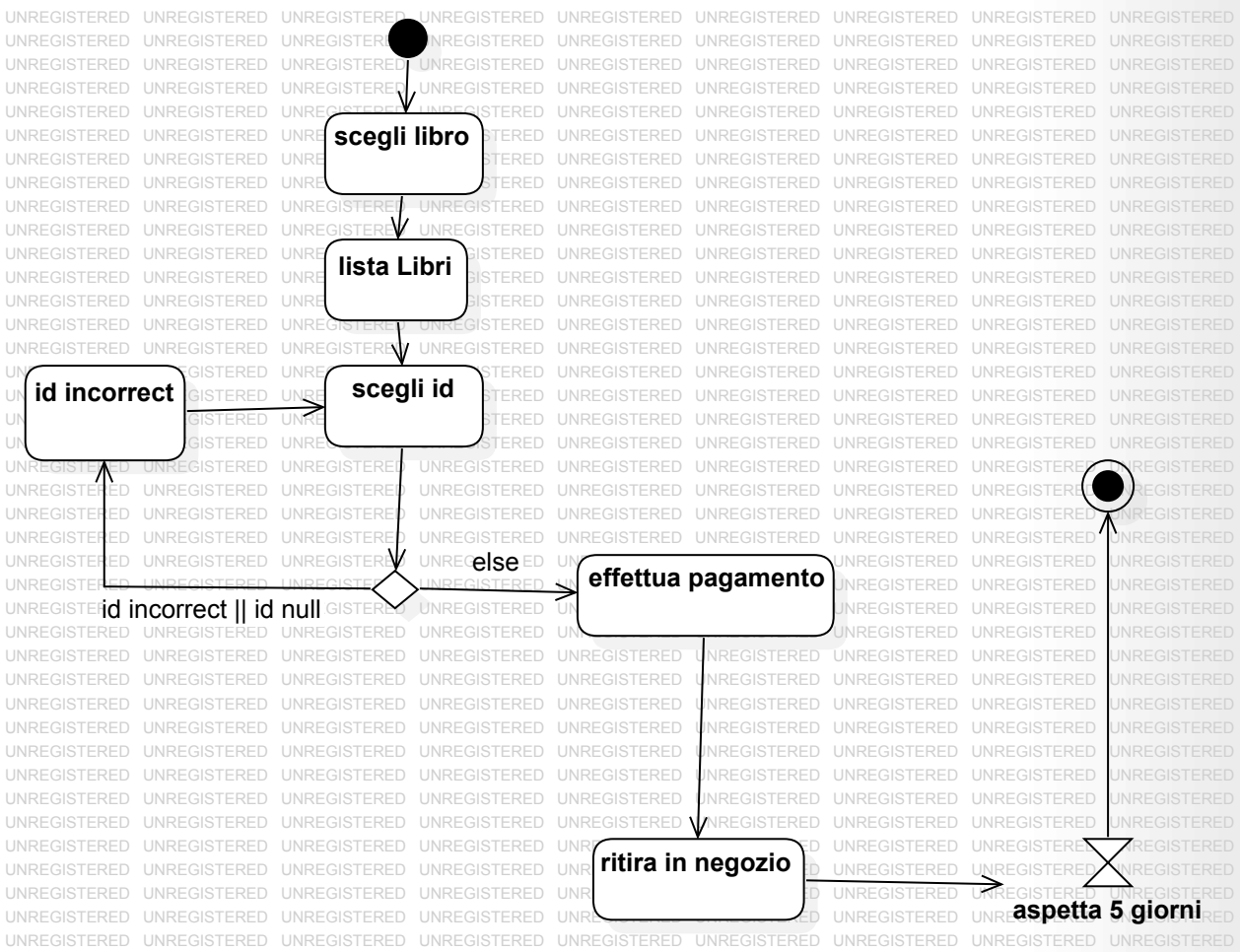


Design patterns

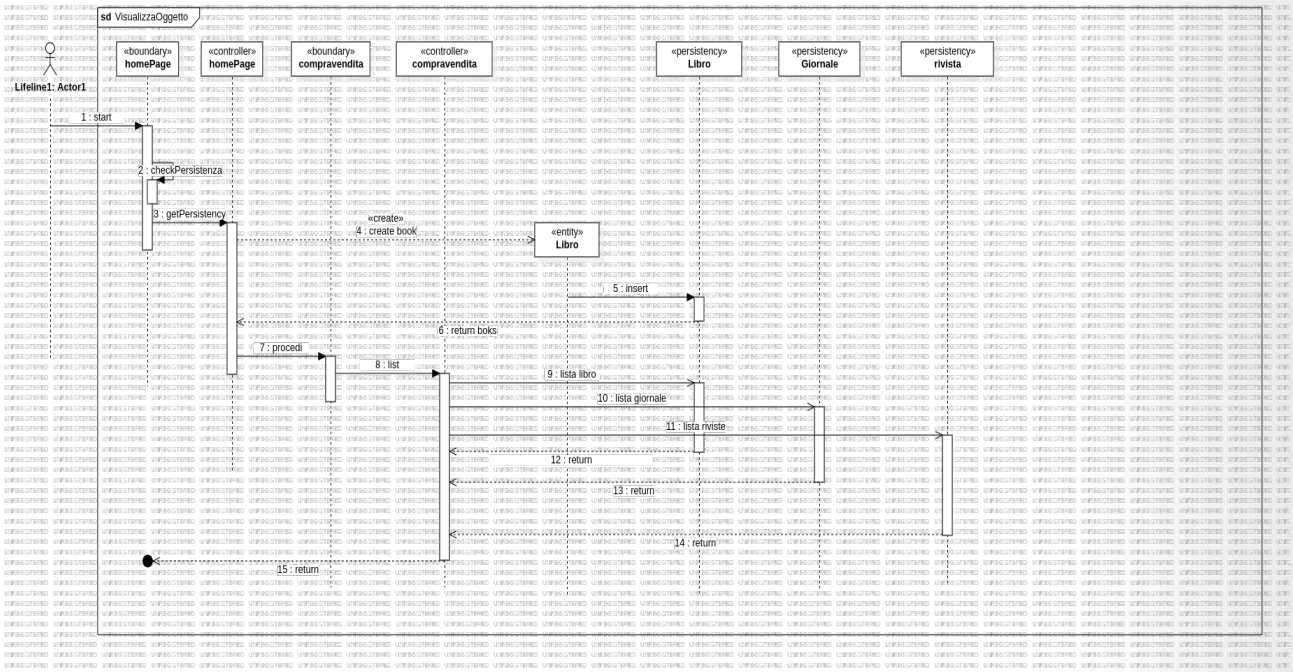
Come pattern sono stati usati due creazionali : La factory (sotto esame) ed il singleton (non sotto esame)



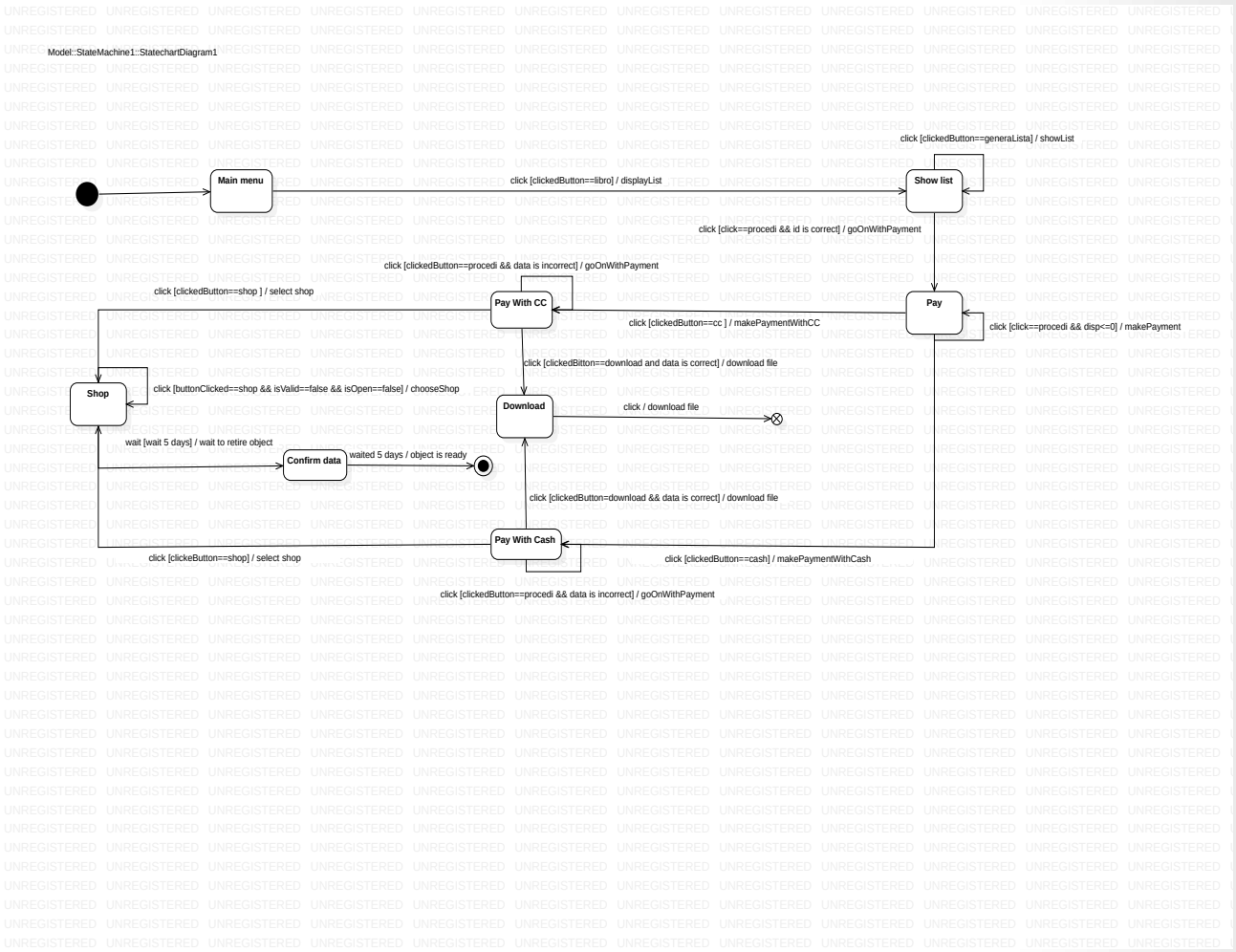
Activity diagram



Sequence Diagram



State diagram



Testing

Test code

Questo primo test parametrico serve invece a vedere , se in seguito ad acquisto di un libro e con scelta di scaricarlo, se viene visualizzato scaricato e visualizzato correttamente tramite programma apposito (ex pdf)

```
@ParameterizedTest
@ValueSource(ints = {1,2,3,4,5,6})
void scaricaL(int ints) throws DocumentException, IOException, URISyntaxException {
    vis.setTypeAsBook();
    vis.setId(ints);
    cD.scarica(vis.getType());
    assertNotEquals( unexpected: 0,ints);

}
```

Questo test invece serve a testare la corretta effettuazione del pagamento

```
@Test
void controllaL() throws SQLException, IdException {
    vis.setTypeAsBook();
    vis.setId(1);
    vis.setSpesaT(7.99f);
    cPCash.controlla( nome: "franco", cognome: "rossi", via: "via gardenie 8", com: "consegna dopo le ore 12.00");
    assertNotEquals( unexpected: 0,vis.getSpesaT());

}
```

Questo ultimo invece serve a vedere se i dati sono corretti

```
@Test
void checkPagamentoDataL() throws SQLException, IdException {
    vis.setTypeAsBook();
    vis.setSpesaT(12f);
    vis.setId(6);
    cCPD.checkPagamentoData( nome: "franco");

}
```

Andiamo ora a vedere un esempio di quello per la web-app (step-by-step) pagamento con fattura:

```
        @Test
    void testIndexBook() throws InvocationTargetException, IllegalAccessException,
NoSuchMethodException, CsvValidationException, IOException, IdException, ClassNotFoundException {
        System.setProperty("webdriver.chrome.driver", "/usr/bin/chromedriver");
        //schermata index
        WebDriver driver = new ChromeDriver();
        driver.get("http://localhost:8080/original-BookShopOnline25/index.jsp");
        driver.findElement(By.id("libri")).click();
        PropertyUtils.setProperty(sB,"typeB","libro");
        //libri.jsp
        PropertyUtils.setProperty(lB,"listaLibriB",pL.retrieveRaccoltaData());
        driver.findElement(By.id("genera lista")).click();
        driver.findElement(By.id("idOgg")).clear();
        driver.findElement(By.id("idOgg")).sendKeys("6");

PropertyUtils.setProperty(sB,"idB",Integer.parseInt(Objects.requireNonNull(driver.findElement(By.id("idOgg
")).getDomProperty("value"))));
        driver.findElement(By.id("procedi")).click();
        //acquista
        driver.findElement(By.id("quantita")).clear();
        driver.findElement(By.id("quantita")).sendKeys("2");

PropertyUtils.setProperty(sB,"quantitaB",Integer.parseInt(Objects.requireNonNull(driver.findElement(By.id(
"quantita")).getDomProperty("value"))));
```

```
        driver.findElement(By.id("totaleB")).click();
        //totale
PropertyUtils.setProperty(sB,"spesaTB",Float.parseFloat(Objects.requireNonNull(driver.findElement(By.id("t
otale")).getDomProperty("value"))));
       //download
        WebElement input =driver.findElement(By.xpath("//input[@list='metodi']"));
        WebElement option =driver.findElement(By.xpath("//*[@id='metodi']/option[1]"));
        String value = option.getDomProperty("value");
        input.sendKeys(Objects.requireNonNull(value));
        PropertyUtils.setProperty(SystemBean.getInstance(),"metodoPB",value);
        driver.findElement(By.xpath("//input[@id='pdfB']")).click();
        //fattura
        driver.findElement(By.id("nomeL")).sendKeys("francoB");
        driver.findElement(By.id("cognomeL")).sendKeys("rossiB");
        driver.findElement(By.id("indirizzoL")).sendKeys("via papaveri 12");
        driver.findElement(By.id("com")).sendKeys("il cap è 00005 . Chiamare prima al numero");
        String nome=driver.findElement(By.id("nomeL")).getDomProperty("value");
        String cognome=driver.findElement(By.id("cognomeL")).getDomProperty("value");
        String indirizzo=driver.findElement(By.id("indirizzoL")).getDomProperty("value");
        String com=driver.findElement(By.id("com")).getDomProperty("value");
        //setto fattura
        PropertyUtils.setProperty(fB,"nomeB",nome);
```

```
        PropertyUtils.setProperty(fB,"cognomeB",cognome);

        PropertyUtils.setProperty(fB,"indirizzoB",indirizzo);

        PropertyUtils.setProperty(fB,"comunicazioniB",com);

        //setto pagamento

        PropertyUtils.setProperty(sB,"tipoModifica","insert");

        PropertyUtils.setProperty(pB,"idB",0);

        PropertyUtils.setProperty(pB,"metodoB",
PropertyUtils.getProperty(SystemBean.getInstance(),"metodoPB"));


PropertyUtils.setProperty(pB,"ammontareB",PropertyUtils.getProperty(SystemBean.getInstance(),"spesaTB
"));

        PropertyUtils.setProperty(pB,"esitoB",0);

        PropertyUtils.setProperty(pB,"nomeUtenteB","");

        driver.findElement(By.id("buttonC")).click();

        //schermata download

        String titoloL=driver.findElement(By.id("titoloL")).getDomProperty("value");

        PropertyUtils.setProperty(dB,"idB",sB.getIdB());

        PropertyUtils.setProperty(dB,"titoloB",titoloL);

        driver.findElement(By.id("downloadB")).click();

        assertNotEquals(0,Integer.parseInt(PropertyUtils.getProperty(sB,"idB").toString()));

        driver.quit();



    }
```

Infine vediamo quello generato da Selenium :

```
// Generated by Selenium IDE
import org.junit.Test;
import org.junit.Before;
import org.junit.After;
import static org.junit.Assert.*;
import static org.hamcrest.CoreMatchers.is;
import static org.hamcrest.core.IsNot.not;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.remote.RemoteWebDriver;
import org.openqa.selenium.remote.DesiredCapabilities;
import org.openqa.selenium.Dimension;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.interactions.Actions;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;
import org.openqa.selenium.JavascriptExecutor;
import org.openqa.selenium.Alert;
import org.openqa.selenium.Keys;
import java.util.*;
import java.net.MalformedURLException;
import java.net.URL;
public class TestIdTest {
private WebDriver driver;
private Map<String, Object> vars;
JavascriptExecutor js;
@Before
public void setUp() {
driver = new ChromeDriver();
js = (JavascriptExecutor) driver;
vars = new HashMap<String, Object>();
}
@After
        public void tearDown() { driver.quit(); }
@Test
public void testId() {
        driver.get("http://localhost:8080/BookShopOnline25/libri.jsp");
        driver.manage().window().setSize(new Dimension(1050, 708));
```

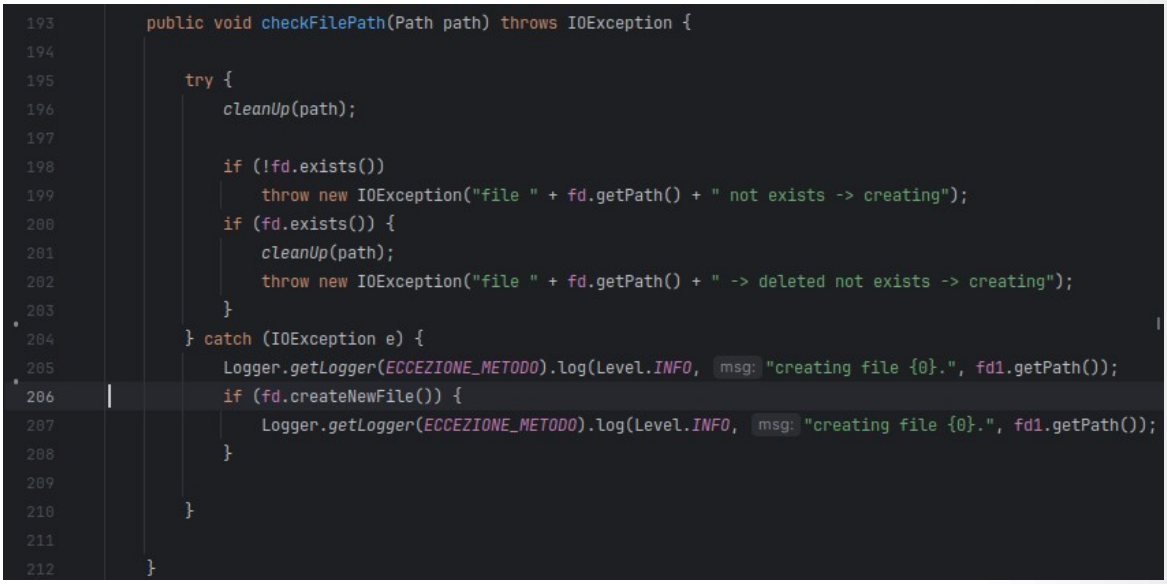
```
        driver.findElement(By.id("idOgg")).click();
        driver.findElement(By.id("idOgg")).sendKeys("4");
        {
        String value = driver.findElement(By.id("idOgg")).getAttribute("value");
        assertThat(value, is("4"));}
        }
}
```

Codice

Le righe nel nostro caso superano abbondantemente le 4k . Infatti sono 9,5k, Per quanto riguarda la versione desktop si usa javafx (interfacciamento grafico) e mysql (base di dati),

exception

Nel nostro caso oltre alla normale gestione delle eccezioni (throws) da parte della jvm, nell'esempio si va a gestire a carico del programmatore l'eccezione :



In dettaglio : alla riga 196 si elimina il file se esiste. Mentre alla riga 198 , se il file non esiste , si lancia eccezione con il messaggio , la quale gestione non è la semplice stampa a schermo del messaggio , ma il file viene creato. Analogamente succede alla riga 200 : se il file esiste , viene cancellato e poi viene lanciata eccezione . Come già detto per qualunque eccezione venga sollevata, si crea di nuovo il file.

Dao

In questa applicazione si usano tre tipologie di persistenza : quella su file (persistente), quella su database e quella su file temporaneo(memoria),

Database

File system

Per quanto riguarda la persistenza dei file :

- File per database (Cartella FileSqL)
  - cartacredito.sql : file configurazione creazione della tabella in mysql.
  - fattura,sql: file configurazione creazione tabella sql.
  - giornale.sql : file configurazione creazione tabella sql.
  - libro.sql : file configurazione creazione tabella sql.
  - rivista.sql : file configurazione creazione tabella sql.
  - negozio.sql : file configurazione creazione tabella sql.
  - utente.sql : file configurazione creazione tabella sql.
  - pagamento.sql : file configurazione creazione tabella sql.
  - files “popola***.sql” si intendono quei file che servono per la popolazione della tabella. Non tutti i file o meglio le tabelle sopra citate posseggono il file di inizializzazione. Sono escluse cartacredito.sql , fattura.sql , pagamento.sql.

File per il report persistente (Cartella report)

- Analogo a quelli sql solo che in questi file sono contenuti dati che verranno salvati su disco se si usa la modalità file .
- reportCartaCredito.csv : file che contiene tutte le carte di credito del sistema.
- reportFattura.csv : file che contiene tutte le fatture ordiate dagli utenti.

- reportFinale.csv : file che viene constatato per avere idea delle vendite totali effettuate degli articoli.

- reportGiornale.csv: file che contiene tutti i giornali presenti nel catalogo.

- reportLibro,csv: analogo a giornale, ma contiene i libri.

- reportRivista,csv: analogo a libro, ma contiene le riviste.

- reportNegozi.csv: contiene elenco dei negozi dove è possibile recarsi per ritirare e/o pagare oggetto desiderato.

- reportPagamento.csv : file che mostra tutti i pagamenti effettuati , sia in modalità contrassegno che in modalità carta di credito.

- ReportUtente: file che mostra gli utenti che possono interagire con il sistema.

File per il report non persistente finché non si spegne la macchina :Cartella report:

- questa cartella contiene i stessi dati contenuti nei file csv e database , ma in versione volatile.

- Il formato si questi file è : serializzazione***,ser , dove al posto di “*” ci vanno i soliti nomi : pagamento, fattura, libro, giornale, rivista, utente,fattura,carta credito,negozio.

Resources (src/main/resources/)

Qui vengono descritti i file di configurazione per ambiente applicazione.

Sono suddivisi in 3 cartelle : Configurations , CsvFiles e TmpFiles.

Alla prima appartengono :

- bookCategories,properties : elenco delle categorie dei libri presenti nel catalogo.
- cartaCredito,properties : contiene i dati degli utenti (magari usate per i test).
- configDB,properties : contiene i dati di configurazione per connessione al DB.
- magazineCategories,properties : contiene le categorie delle riviste presenti nel sistema.
- object,properties : contiene dei dati usati per i test .
- titles.,properties : contiene tutti i titoli dei libri presenti nel catalogo.
- users.,properties : dati degli utenti usati per i test.

Alla seconda appartengono :

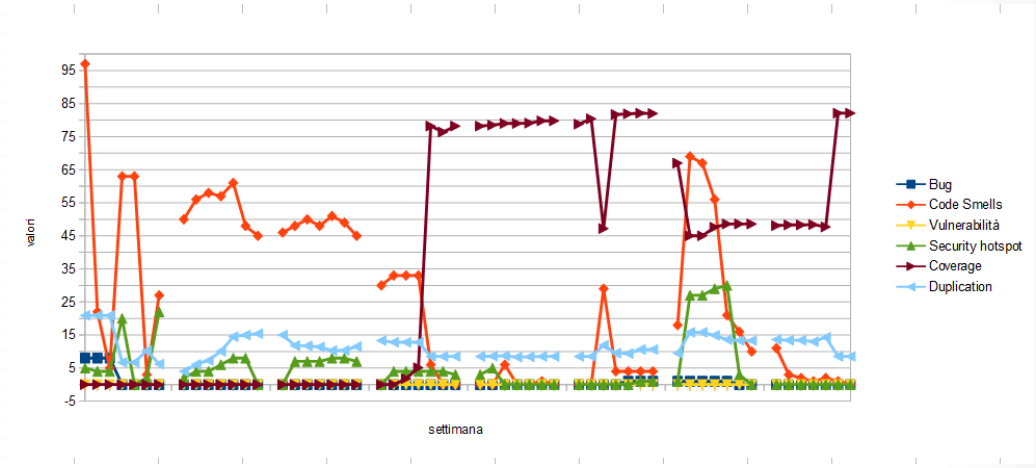
- tutti i files sopra citati nella sezione Report.

Mentre all'ultima appartengono :

- cartaCredito1/6.txt : files che vengono caricati in memoria per inizializzare.
- fattura1,txt : file che viene usato per inizializzare fattura in memoria.
- giornale1/12.txt: serve per inizializzare il catalogo con i giornali.
- libro1/19.txt ;serve per inizializzare il catalogo con i giornali.
- negozio1/4.txt: inizializza i negozi disponibili.
- rivista1/5.txt: inizializza le riviste del catalogo.
- utente1/7.txt: inizializza gli utenti del sistema,

SonarCloud

Per quanto riguarda le metriche prestazionali, sono riportate nel grafico seguente. In particolare ve ne sono 2 sotto attenzione:la prima è avere code smells pari a zero (linea arancione) , mentre la seconda è avere coverage (linea marrone) >=80%, e nel nostro caso si attesta intorno ad 80,2%:



Per quanto riguarda il link al repo è il seguente :

https://github.com/ilciro/BookShopOnline25

Video

Per quanto riguarda il video , vedere nella cartella del repo :

https://github.com/ilciro/BookShopOnline25/tree/master/documentazione