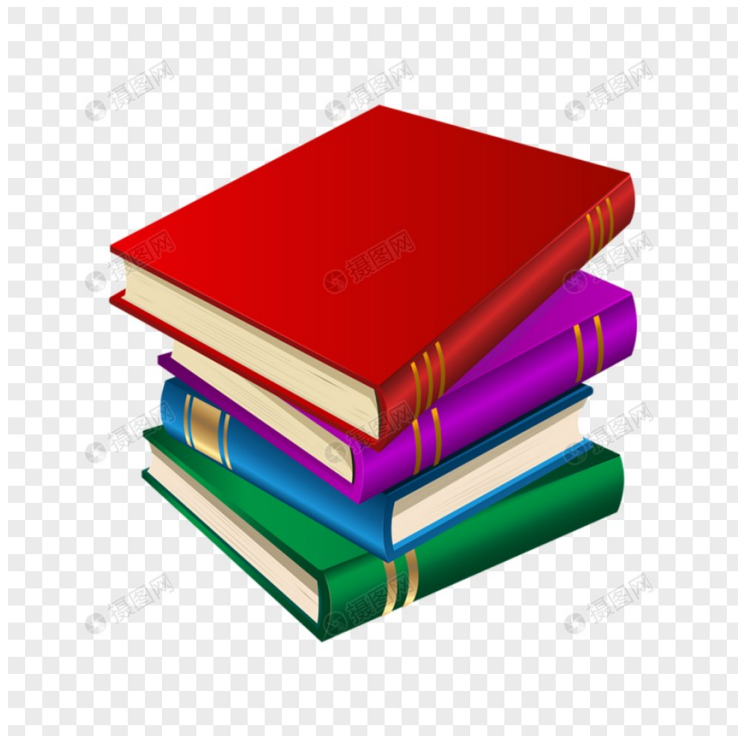


# Book shop online

Daniele Cinus 0219066 Ingegneria informatica



## index

Intro.....	2
Aim of the system.....	3
Overview of the system.....	3
Hw and Sw.....	3
Pros and Cons.....	3
PRO:.....	3
CONS:.....	3
User Stories.....	3
Functional Requirements.....	4
Use Case.....	5
Overview.....	5
Internal step.....	6
Acquista oggetto.....	6
Estensioni:.....	6
Storyboards.....	7
Design.....	8
VOPC Acquista oggetto.....	9
Design level.....	9
.....	10
Design patterns.....	10
Activity diagram.....	12
Sequence state.....	13
State diagram.....	13
Testing.....	14
Test code.....	14
Codice.....	17
exception.....	17
Dao.....	17
Database.....	17
File system.....	17
Resources.....	17
close.....	17
SonarCloud.....	17
Video.....	17

## 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:

- ) Aumento bacino utenza e vendite
- ) Catalogo più ampio
- ) Costo inferiore rispetto ad un nuovo negozio fisico
- ) in caso di indisponibilità del prodotto presso il punto vendita il negoziante può ordinarlo per conto del cliente ottenendo comunque la vendita.

### CONS:

- ) Grande concorrenza
- ) Eventuali problemi di gestione
- ) Compatibilità dei formati

# User Stories

In giallo user stories sotto esame

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

così da poter avere un resoconto totale

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

così 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

così posso avere un resoconto dei progressi

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

così 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

così da fornire un supporto alla vendita

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

così da permettere di aumentare il numero dei oggetti acquistabili

# Functional Requirements

In giallo functional requirements sotto esame

- 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 :**

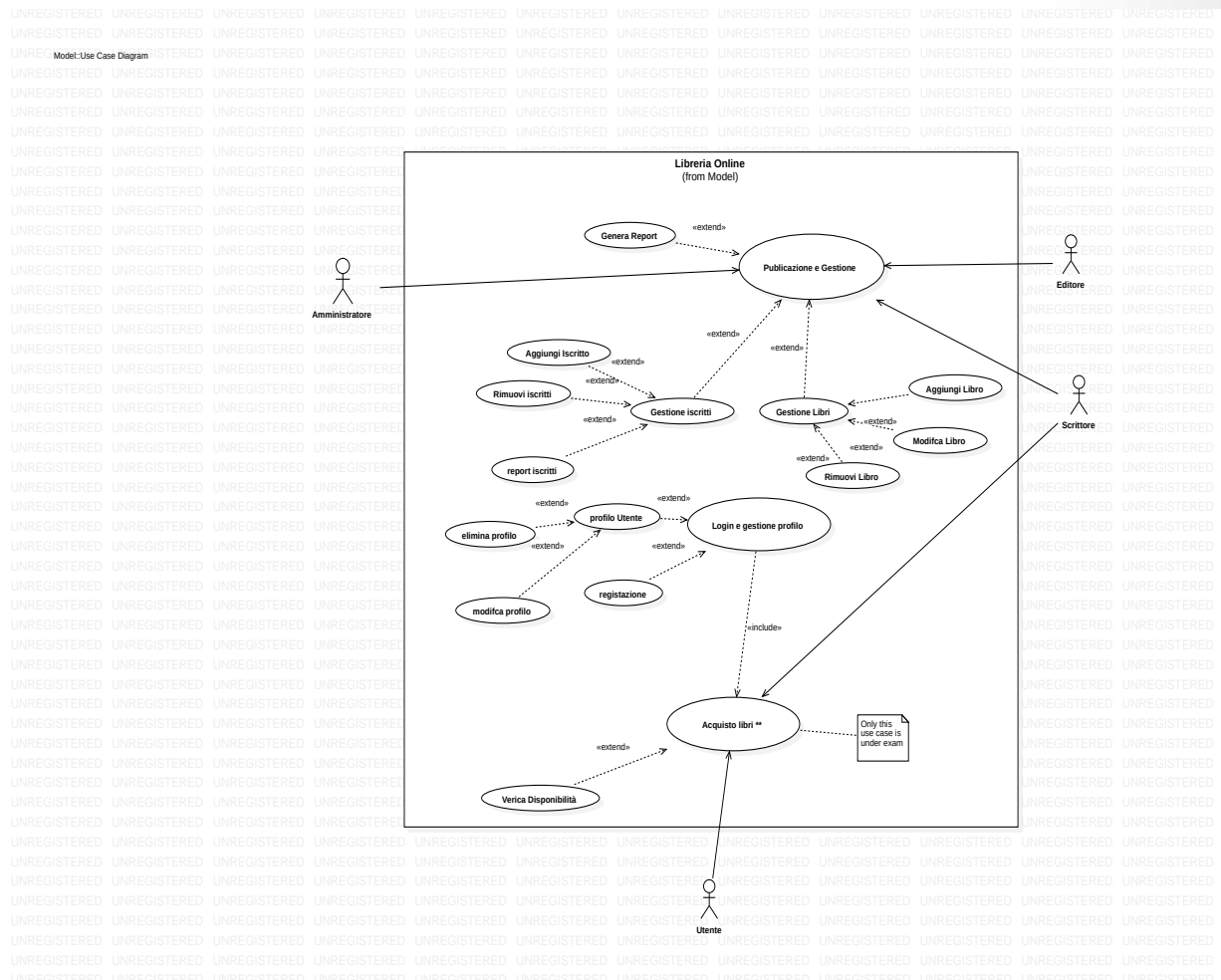
- Fatturato: Si intende l'utile guadagnato, nel periodo di tempo selezionato
- Prodotti : ci si riferisce sempre a libri, riviste e giornali
- Casa editrice ad esso associata : insieme dei libri pubblicati dall'editore e insieme degli scrittori che pubblicano con essa e nome casa editrice

#### **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. Il sistema permetter di scegliere il libro desiderato
2. Il sistema mostra l'elenco completo dell'oggetto desiderato
3. Il sistema , una volta scelto l'oggetto , ne permette acquisto pagando cash
4. Il sistema permette di scegliere come ritirare il libro in negozio

### Estensioni:

- 1.a Il sistema permette di scegliere la rivista

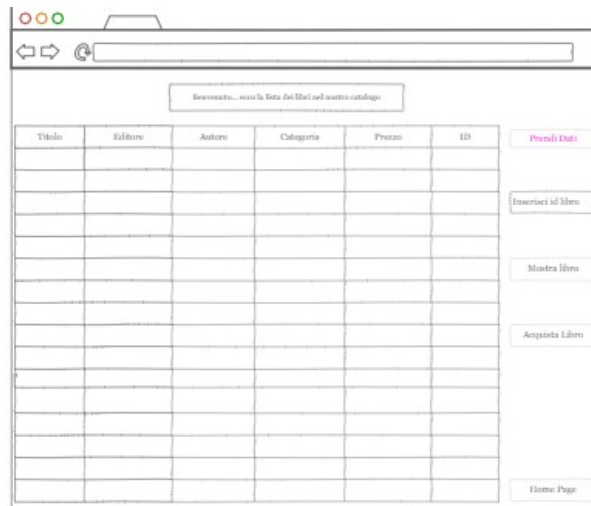
1.b Il sistema permette di scegliere il giornale

3.a Il sistema permette di poter pagare con carta credito

4.a Il sistema permette di scaricare in locale la copia .

## 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.



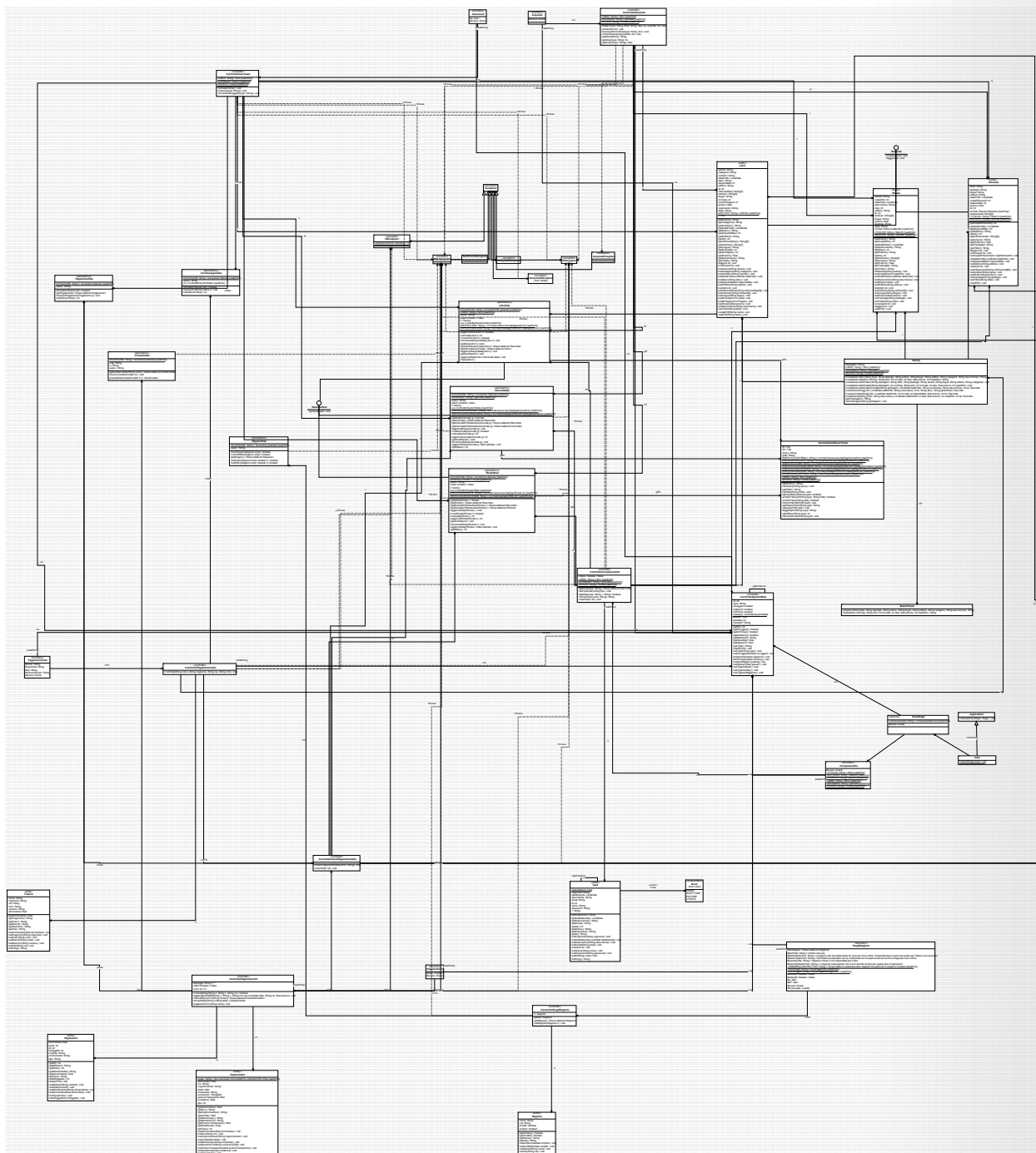
La seconda , invece mostra la storyboard n.10, ovvero la storyboard nella quale il sistema permette all'utente di scegliere il negozio in cui effettuare il ritiro.



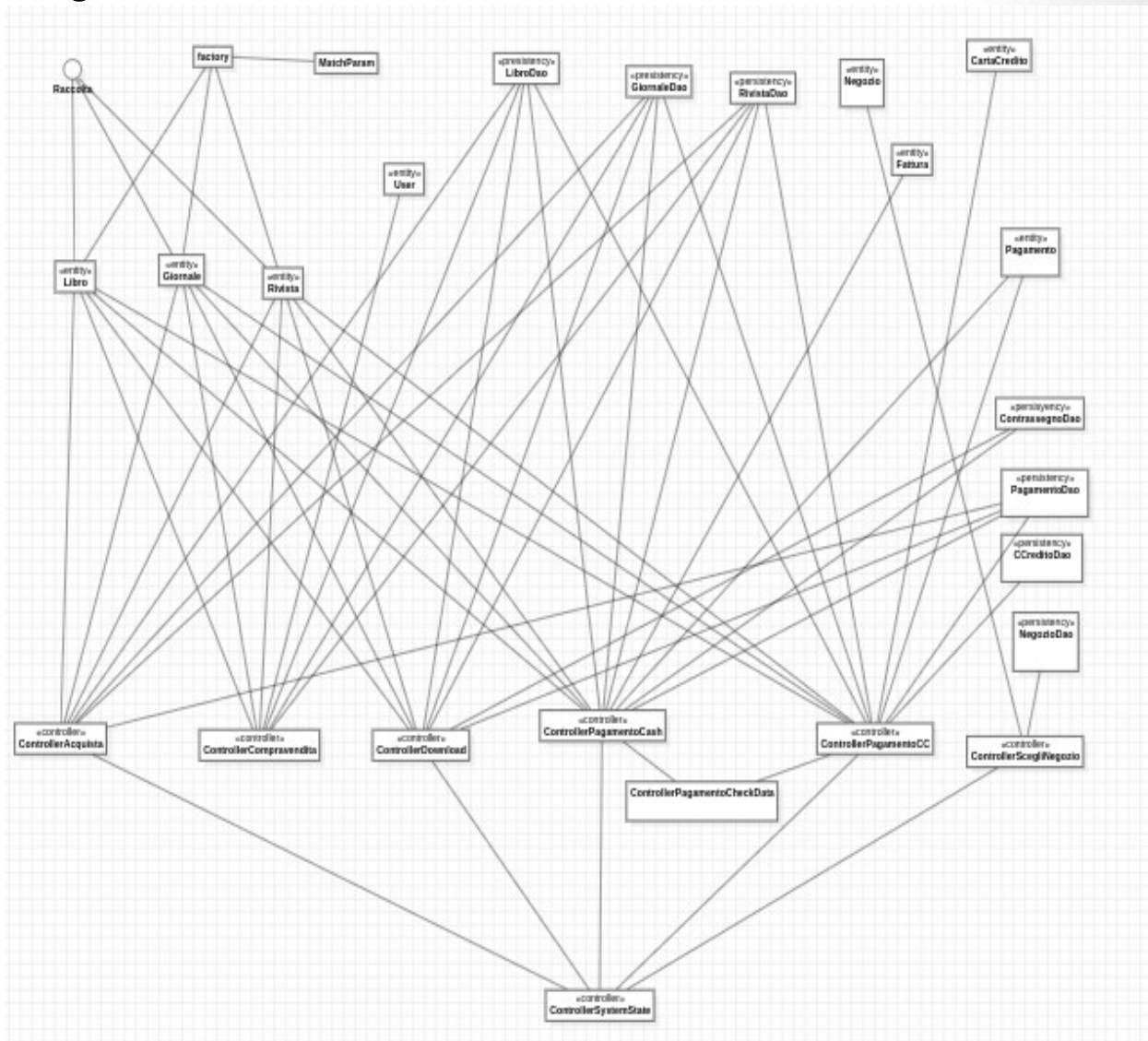


# Design

## VOPC Acquista oggetto

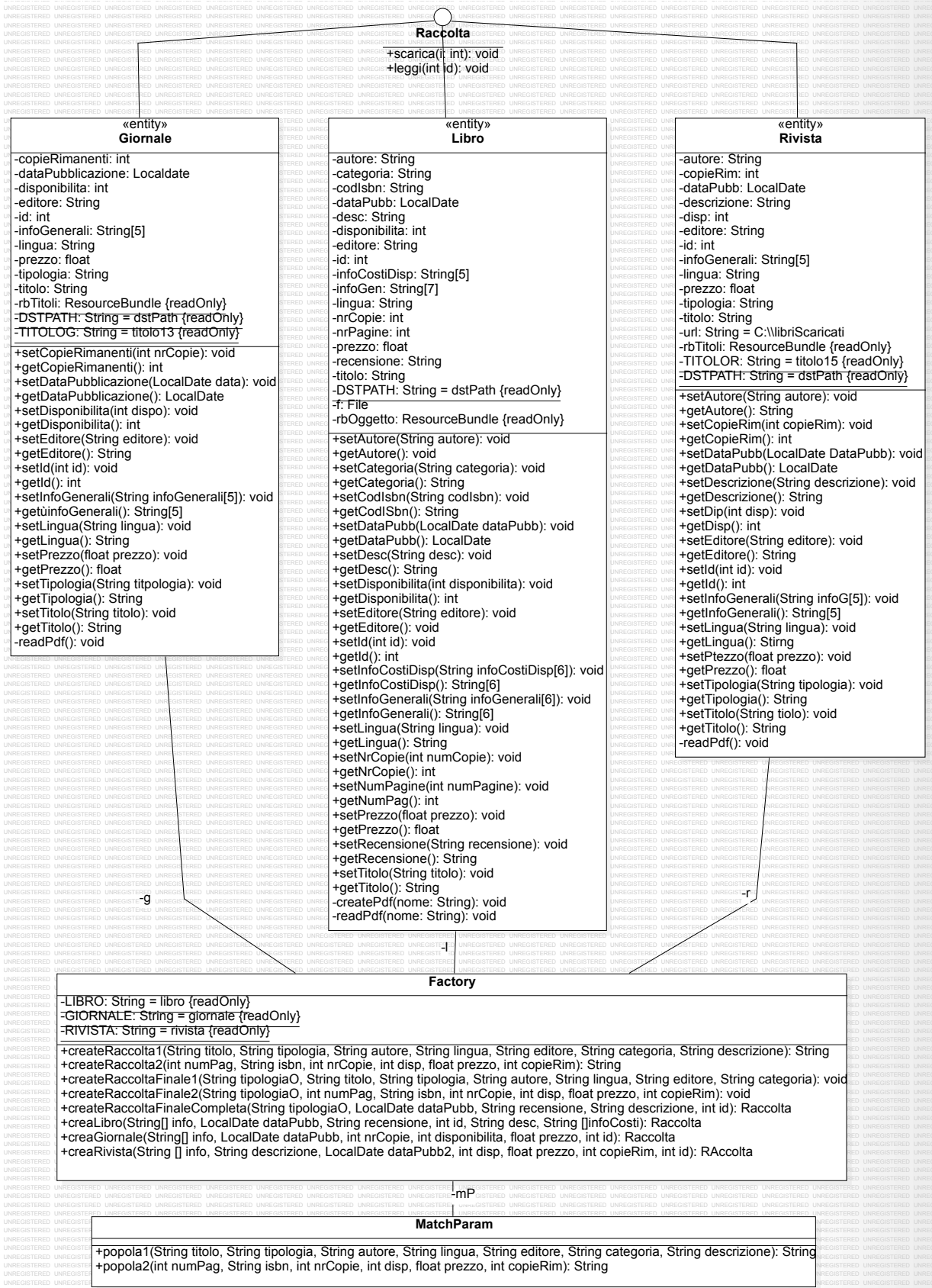


## Design level



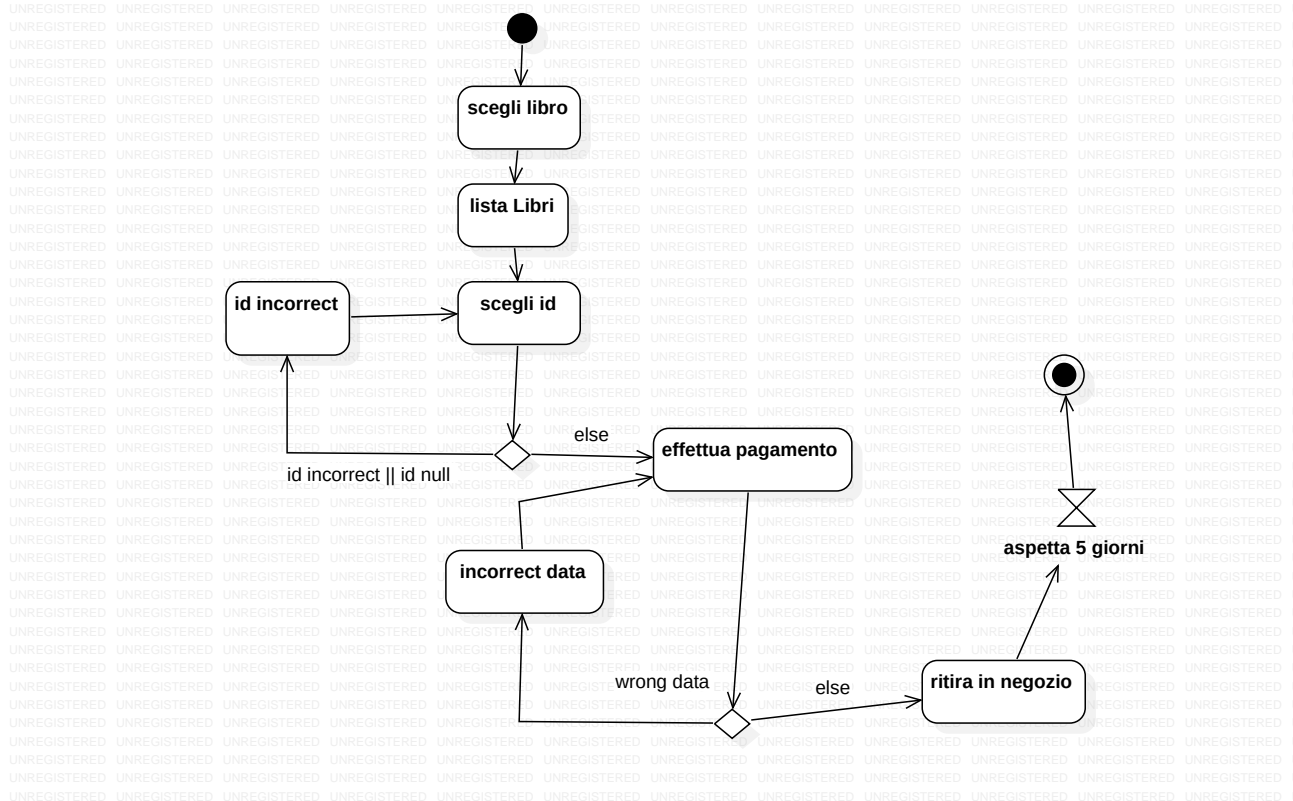
# Design patterns

Nel progetto si sono utilizzati 2 pattern : la Factory(crezionale) e il singleton (creazionale) . Da notare che solo la factory è sotto esame.



# Activity diagram

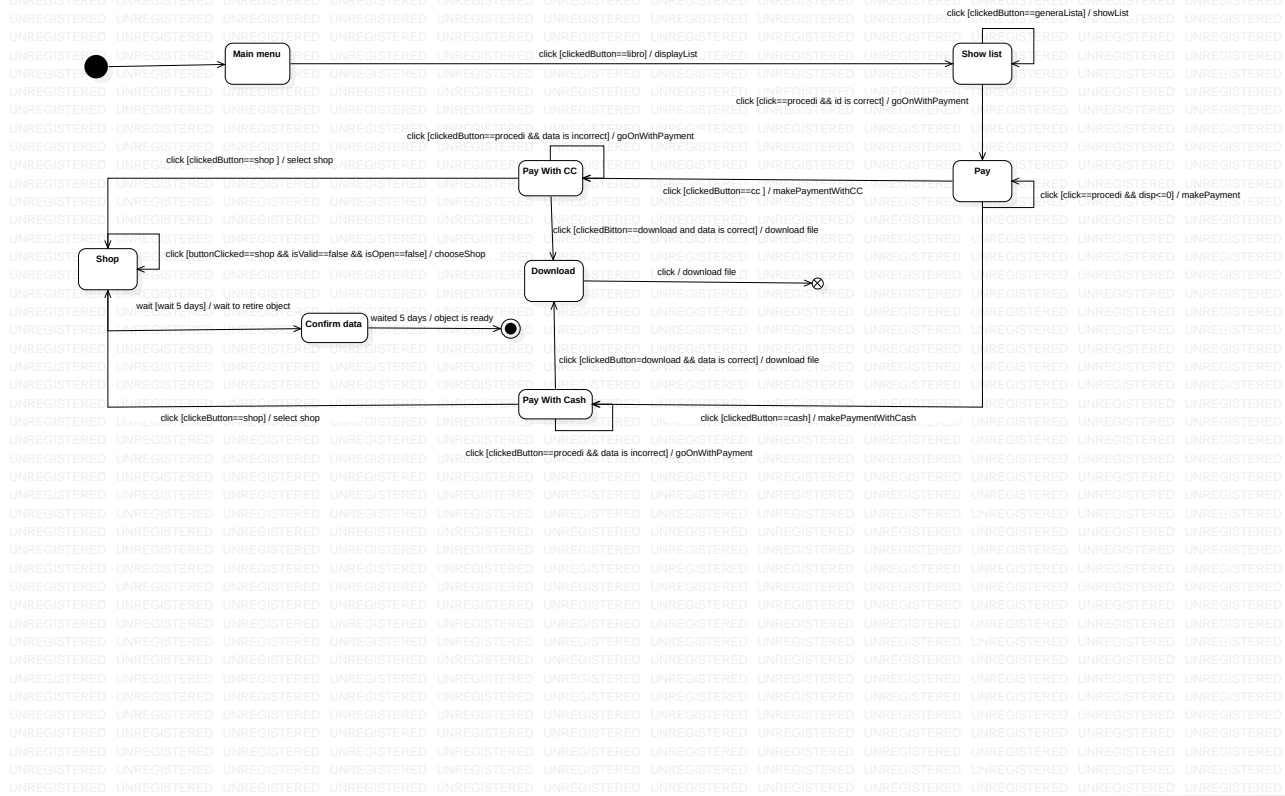
Model::Activity1::ActivityDiagram1



[illegible]

# State diagram

Model: StateMachine1: StatechartDiagram1



# 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());
    assertEquals( unexpected: 0,ints);
}
```

Questo test invece serve a testare la corretta effettuazione del pagamento

```
@Test
void controlla() 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");
    assertEquals( unexpected: 0,vis.getSpesaT());
}
```

Questoultimo 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 testLoginUserLibro() throws InvocationTargetException, IllegalAccessException,
NoSuchMethodException {
    System.setProperty("webdriver.chrome.driver", "/usr/bin/chromedriver");
    //schermata index
    driver = new ChromeDriver();
    driver.get("http://localhost:8080/original-BookShopOnline/index.jsp");
    driver.findElement(By.id("buttonLogin")).click();
    driver.findElement(By.id("emailL")).sendKeys("giuliaConforto@gmail.eu");
    driver.findElement(By.id("passL")).sendKeys("12345678Gc");
    PropertyUtils.setProperty(uB,"emailB",driver.findElement(By.id("emailL")).getAttribute("value"));
    PropertyUtils.setProperty(uB,"passB",driver.findElement(By.id("passL")).getAttribute("value"));
    driver.findElement(By.id("loginB")).click();
    //schermata utente: libro , giornale , rivista , logout,ricerca
```

```

driver.findElement(By.id("buttonL")).click();
PropertyUtils.setProperty(IB,"elencoLibriB", ID.getLibri());
driver.findElement(By.id("idOgg")).sendKeys("2");
int id=Integer.parseInt(driver.findElement(By.id("idOgg")).getAttribute("value"));
PropertyUtils.setProperty(IB,"idB",id);
I.setId((Integer) PropertyUtils.getProperty(IB,"idB"));
String titolo=ID.getData(I).getTitolo();
PropertyUtils.setProperty(sB,"idB", id);
PropertyUtils.setProperty(IB,"idB", id);
I.setId(id);
PropertyUtils.setProperty(sB, "titoloB",ID.getData(I).getTitolo());
PropertyUtils.setProperty(aB, "titoloB",sB.getTitoloB());
PropertyUtils.setProperty(aB,"prezzoB",ID.getData(I).getPrezzo());
driver.findElement(By.id("procedi")).click();
//schermata acquista
driver.findElement(By.id("quantita")).clear();
driver.findElement(By.id("quantita")).sendKeys("3");
int quantita=Integer.parseInt(driver.findElement(By.id("quantita")).getAttribute("value"));
PropertyUtils.setProperty(sB,"quantitaB",quantita);
driver.findElement(By.id("totaleB")).click();
float prezzo=Float.parseFloat(driver.findElement(By.id("totale")).getAttribute("value"));
PropertyUtils.setProperty(sB,"spesaTB",prezzo);
PropertyUtils.setProperty(aB,"prezzoB",PropertyUtils.getProperty(sB,"spesaTB"));
//metodo cash
WebElement input =driver.findElement(By.xpath("//input[@list='metodi']"));
WebElement option =driver.findElement(By.xpath("//*[@id='metodi']/option[1]"));
String value = option.getAttribute("value");
input.sendKeys(value);
PropertyUtils.setProperty(sB,"metodoPB",value);
driver.findElement(By.id("pdfB")).click();
//schermata fattura
driver.findElement(By.id("nomeT")).sendKeys("francoB");
driver.findElement(By.id("cognomeT")).sendKeys("rossiB");
driver.findElement(By.id("indirizzoT")).sendKeys("via papaveri 12");
driver.findElement(By.id("com")).sendKeys("il cap è 00005 . Chiamare prima al numero 9411526");
String nome=driver.findElement(By.name("nomeT")).getAttribute("value");
String cognome=driver.findElement(By.name("cognomeT")).getAttribute("value");
String indirizzo=driver.findElement(By.name("indirizzoT")).getAttribute("value");
String com=driver.findElement(By.name("com")).getAttribute("value");
//setto fattura
PropertyUtils.setProperty(fB,"nomeB",nome);
PropertyUtils.setProperty(fB,"cognomeB",cognome);
PropertyUtils.setProperty(fB,"indirizzoB",indirizzo);
PropertyUtils.setProperty(fB,"comunicazioniB",com);
driver.findElement(By.id("buttonC")).click();
//schermata download
driver.findElement(By.id("titoloL")).sendKeys(titolo);
driver.findElement(By.id("downloadB")).click();
driver.findElement(By.linkText("open pdf")).click();
assertNotEquals(0,PropertyUtils.getProperty(sB,"idB"));

}

```

Infine vediamo quello geerato da Selenium :

Quest’ultimo invece è il test con codice autogenerato da SeleniumIde.Va a caricare la pagina “libri.jsp” e scegliendo il valore 4 come id del libro. Infine confronta il valore scelto con il valore 4

// 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/BookShopOnline/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 :

```
193     public void checkFilePath(Path path) throws IOException {
194
195         try {
196             cleanUp(path);
197
198             if (!fd.exists())
199                 throw new IOException("file " + fd.getPath() + " not exists -> creating");
200             if (fd.exists()) {
201                 cleanUp(path);
202                 throw new IOException("file " + fd.getPath() + " -> deleted not exists -> creating");
203             }
204         } catch (IOException e) {
205             Logger.getLogger(ECCEZIONE_METODO).log(Level.INFO, msg: "creating file {0}.", fd1.getPath());
206             if (fd.createNewFile()) {
207                 Logger.getLogger(ECCEZIONE_METODO).log(Level.INFO, msg: "creating file {0}.", fd1.getPath());
208             }
209         }
210     }
211
212 }
```

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 due tipologie di persistenza : sia quella su file che quella su base di dati

## Database

### File system

Per quanto riguarda la persistenza dei file :

- "src/main/resources/riepilogoLibri.txt";
  - file che mostra il report dei libri
- "src/main/resources/riepilogoGiornali.txt";
  - file che mostra il report dei giornali
- "src/main/resources/riepilogoRiviste.txt";

- file che mostra il report delle riviste
- "src/main/resources/riepilogoUtenti.txt";
  - file che mostra il report degli utenti
- localDbFile.csv
  - file di persistenza degli utenti nel sistema

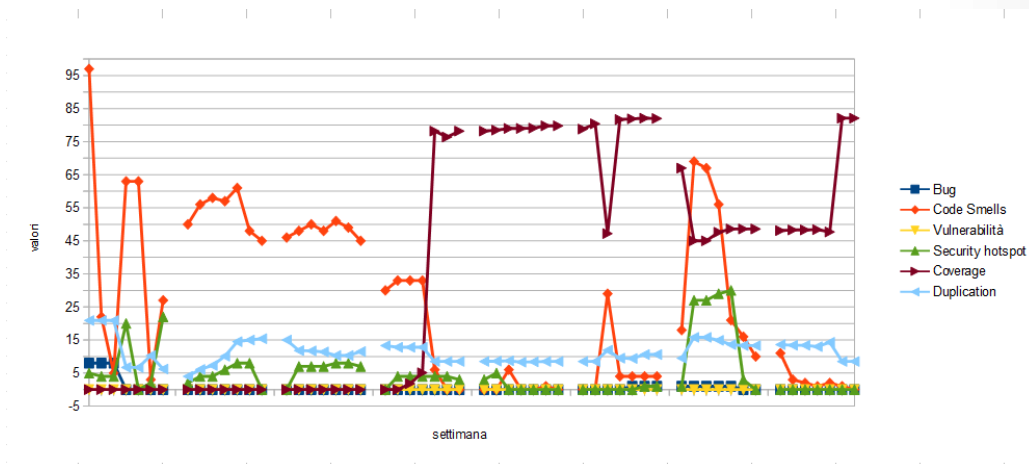
## Resources

Qui vengono descritti i file di configurazione per ambiente applicazione.

- src/main/resources/bookCategories.properties
  - elenco delle categorie di libri possibili
- src/main/resources/book.properties
  - elenco dei parametri e di eventuali libri per interazione con sistema
- src/main/resources/configDb.properties
  - configurazione per connessione a db
- src/main/resources/infoObjects.properties
  - file contenente tutte le possibili informazioni con il sistema
- src/main/resources/magazineCategories.properties
  - elenco delle categorie delle riviste possibili
- src/main/resources/titles.properties
  - file contenente i path dove poter scaricare i libri e gli opportuni titoli
- src/main/resources/users.properties
  - file contenente info degli utenti per interazione con sistema
- src/main/java/laptop/utilities/ConnToDb
  - classe usata per effettuare la connessione con la base di dati

## 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)  $\geq 80\%$ , e nel nostro caso si attesta intorno ad 80,2%:



Per quanto riguarda il link al repo è il seguente :

[https://sonarcloud.io/summary/overall?id=ilciro\\_BookShopOnline](https://sonarcloud.io/summary/overall?id=ilciro_BookShopOnline)

## Video

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

<https://github.com/ilciro/BookShopOnline>