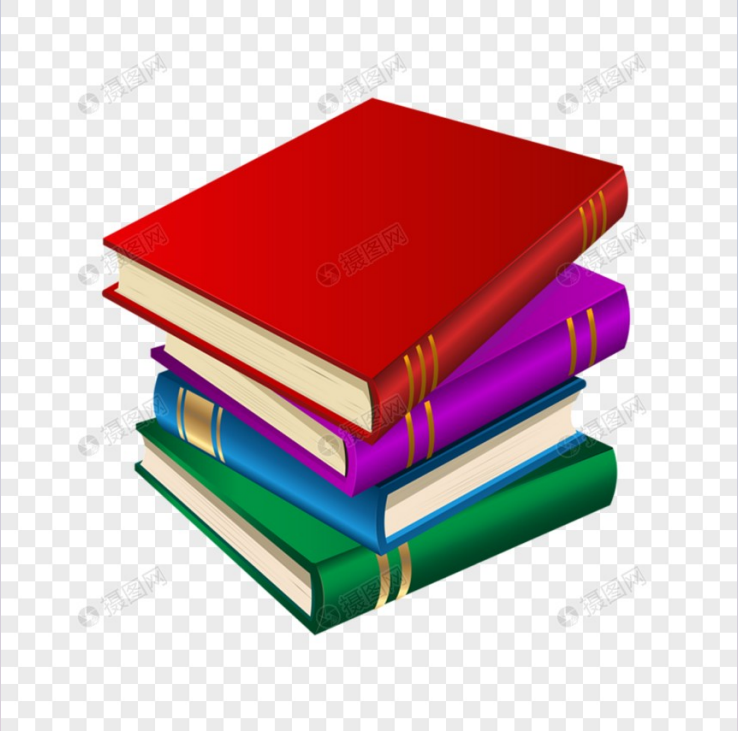
Book shop online

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 resoconto 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

Amazon

Pro: il catalogo è molto più ampio

Contro: il sistema non permette il pagamento con contrassegno

Mondadori

Pro: Catalogo più ampio

Contro: Il sistema permette solo acquisto libri

User Stories

- Come amministratore voglio poter consultare
  - vendite totali
  - guadagno
  - resoconto
- come amministratore accedo a :
  - capienza catalogo, lista prodotti trattati, giacenza globale e giacenza singolo prodotto
- come editore voglio poter:
  - vedere i libri associati alla sua casa editrice
  - vedere gli scrittori associati alla mia casa editrice
- come scrittore voglio poter:
  - pubblicare libri con o senza editore
- 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
- come cliente voglio poter :
  - acquistare libri in versione cartacea o digitale

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 report 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 fornire un metodo per la pubblicazione, modifica e eliminazione dei libri
- Il sistema deve fornire 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 :

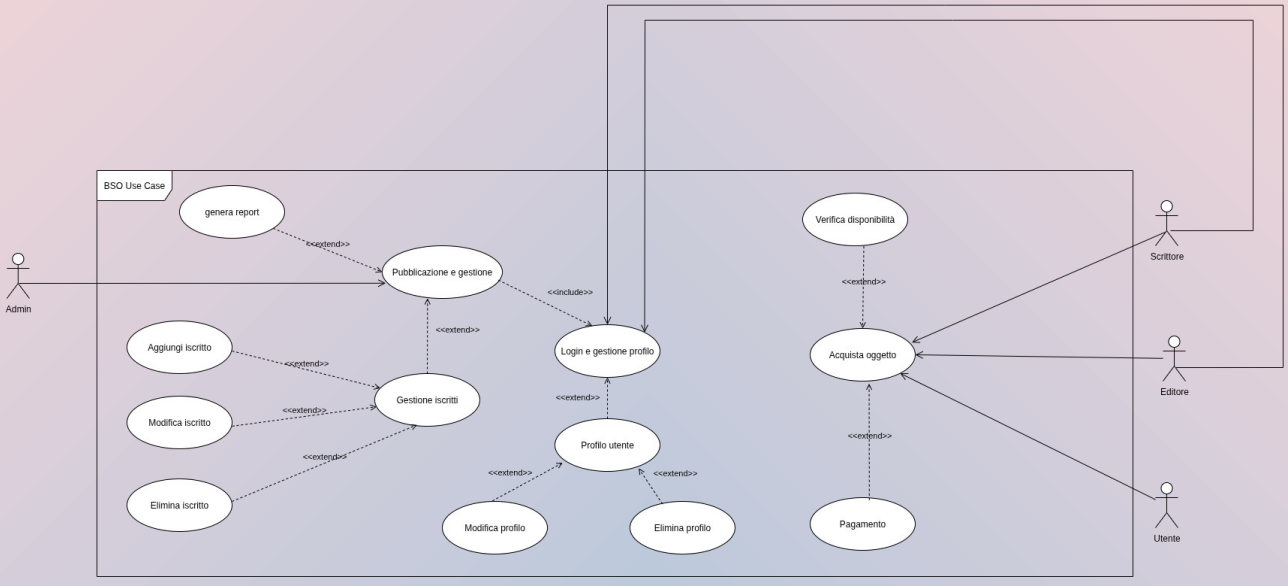
- Fatturato: Si intende l'utile guadagnato, nel periodo di tempo selezionato
- Prodotti : ci si riferisce sempre a libri, riviste e giornali
- Casa editrice ad esso associata : insieme dei libri pubblicati dall'editore e insieme degli scrittori che pubblicano con essa e nome casa editrice

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 Libro

1. Il sistema permette di scegliere il libro
2. l’utente clicca sul bottone mostra lista
3. il sistema mostra l’elenco dei libri presenti nel catalogo
4. l’utente sceglie il libro da acquistare
5. Il sistema , una volta scelto l’oggetto , ne permette acquisto pagando cash
6. l’utente effettua il pagamento inserendo i dati
7. Il sistema permette di scegliere come ritirare il libro in negozio

Estensioni:

1.a Il sistema permette di scegliere la rivista

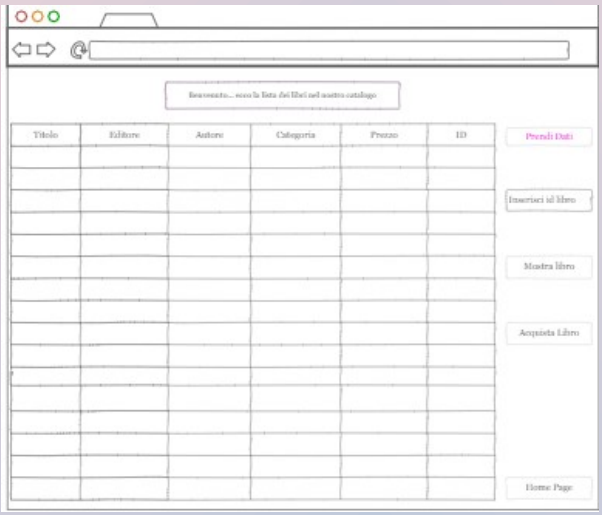
1.b Il sistema permette di scegliere il giornale

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

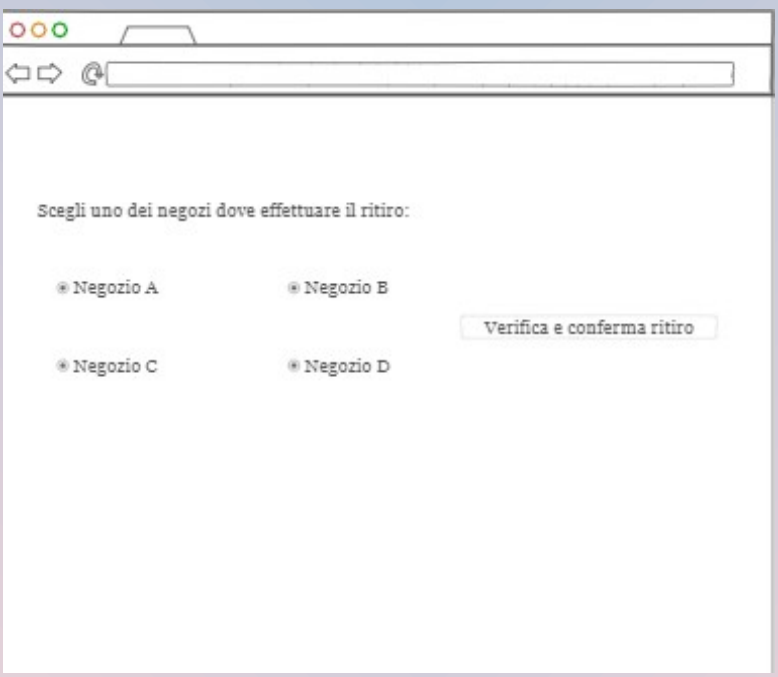
7.a Il sistema permette di scaricare in locale la copia .

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.

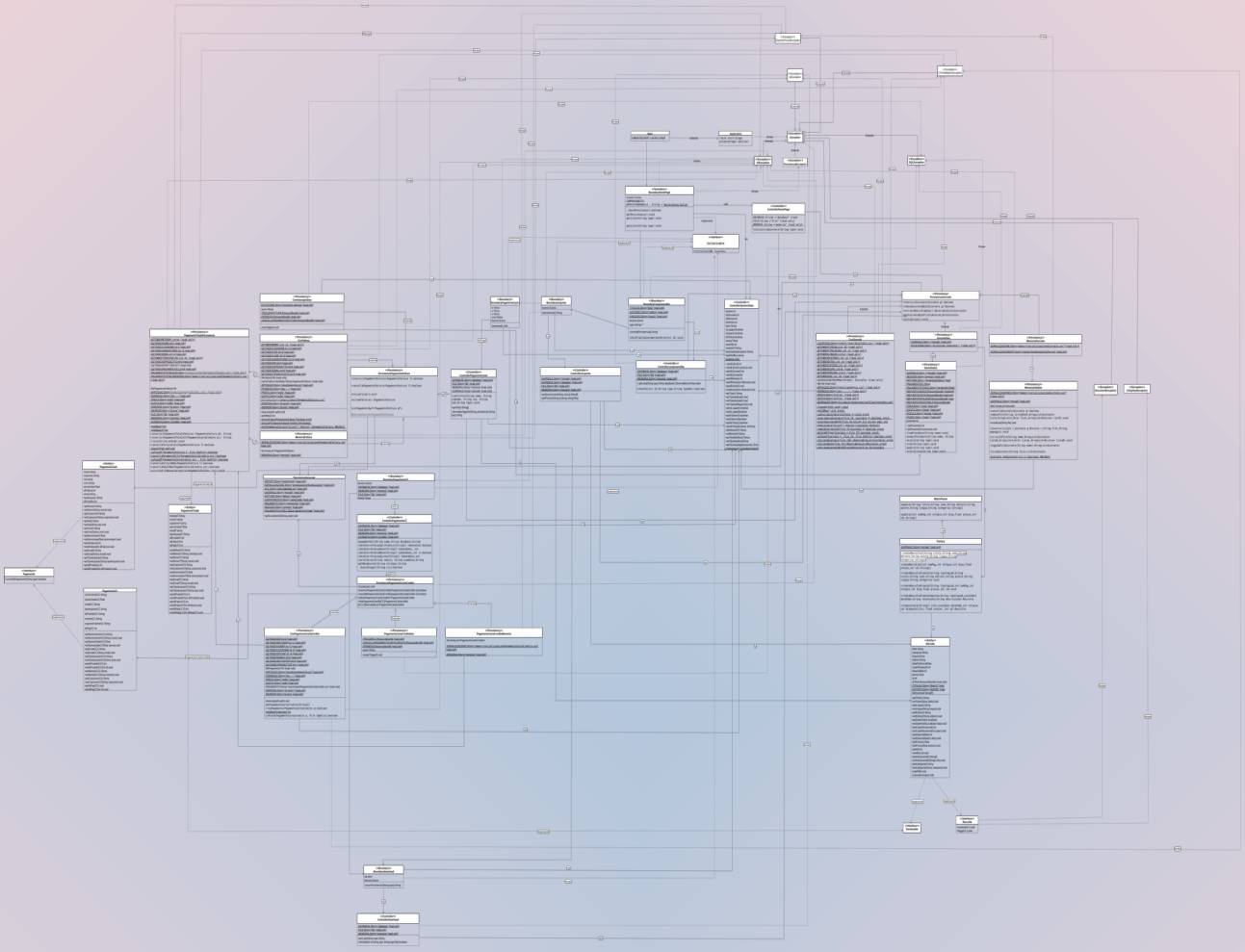


La seconda , invece mostra la storyboard n.10, ovvero la storyboard nella quale il sistema permette all'utente di scegliere il negozio in cui effettuare il ritiro.

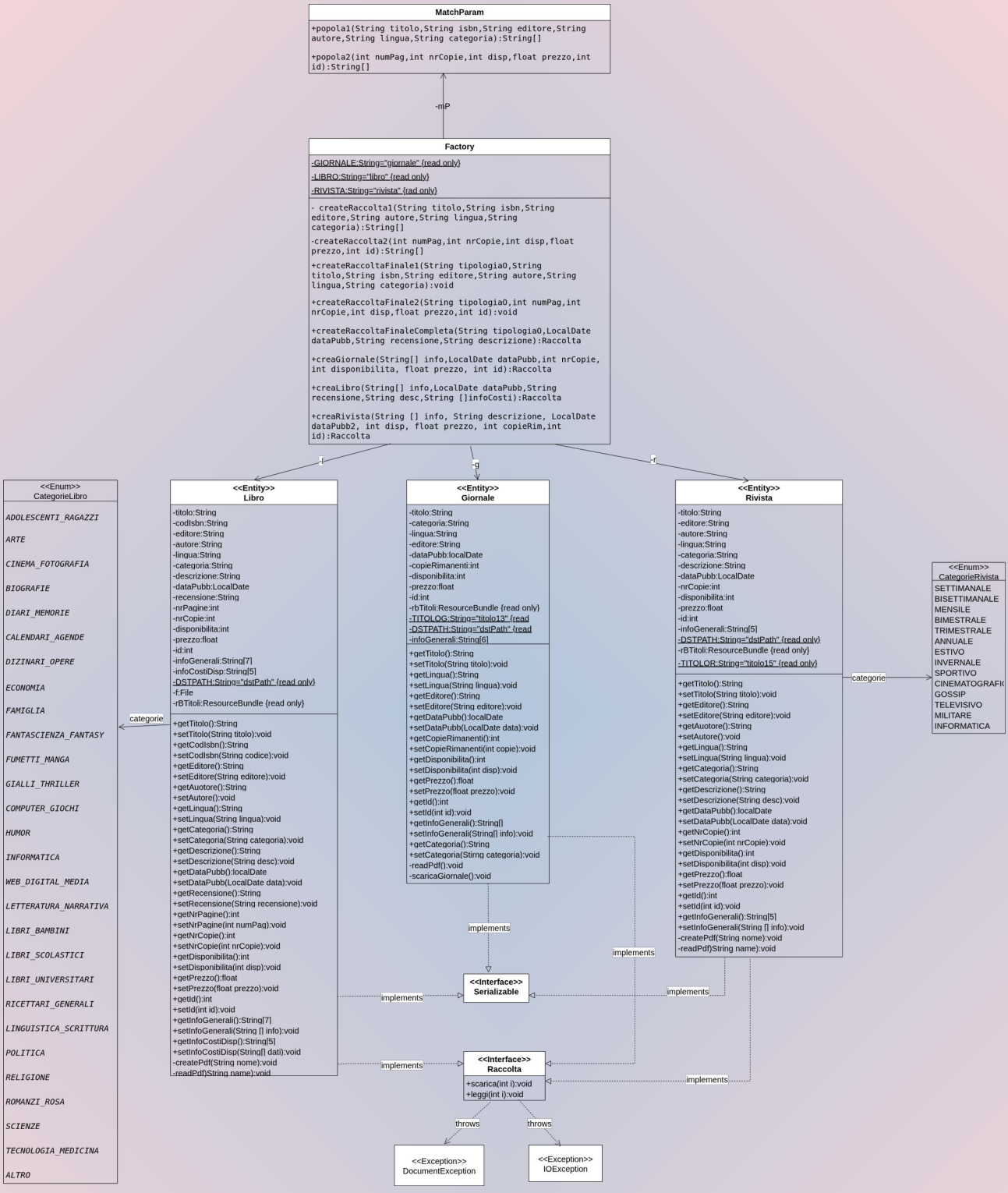


Design

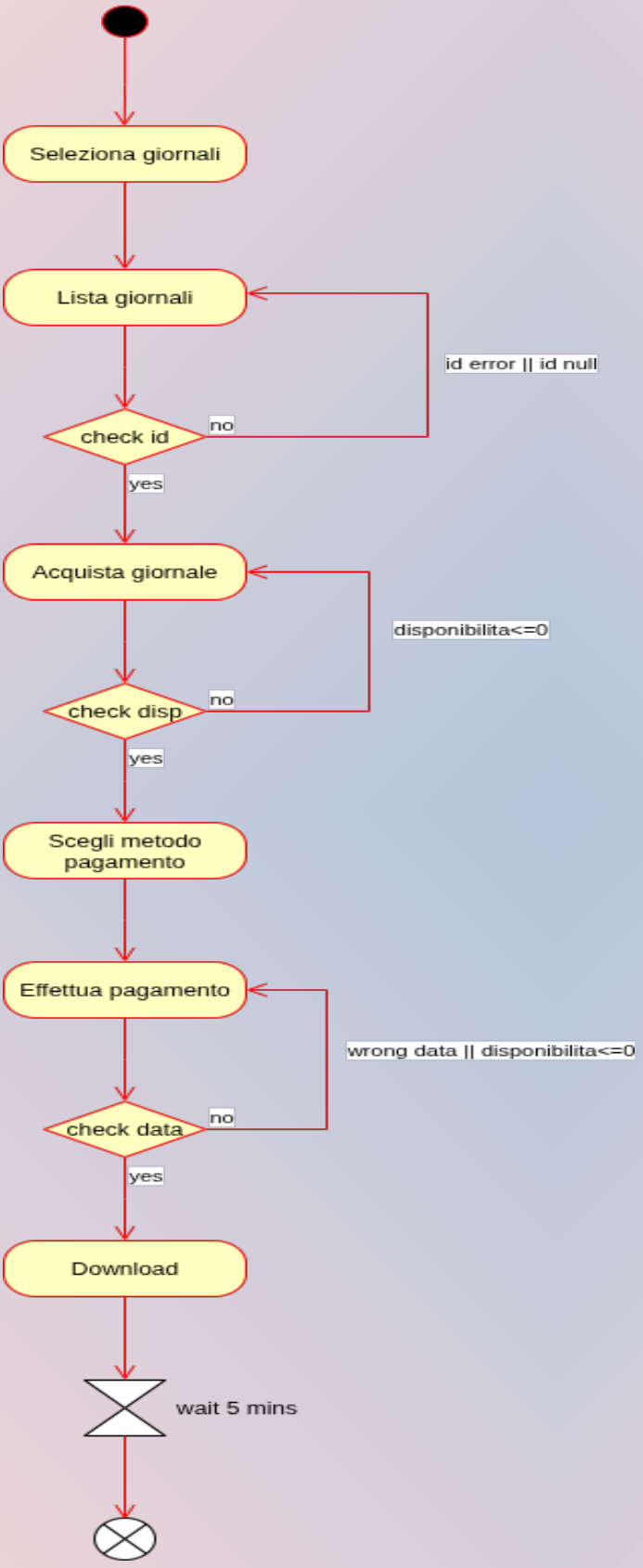
VOPC Acquista oggetto



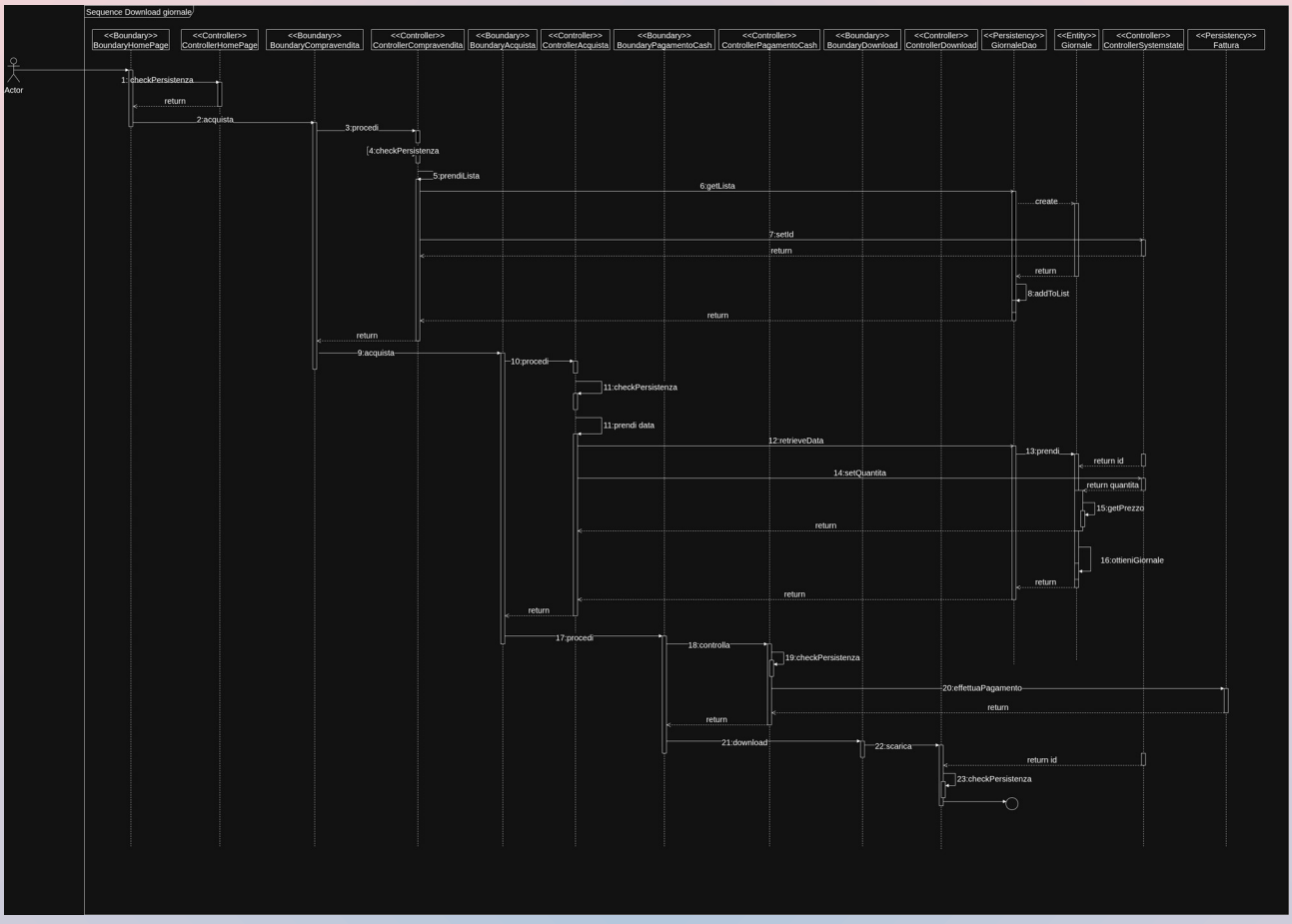
Design level



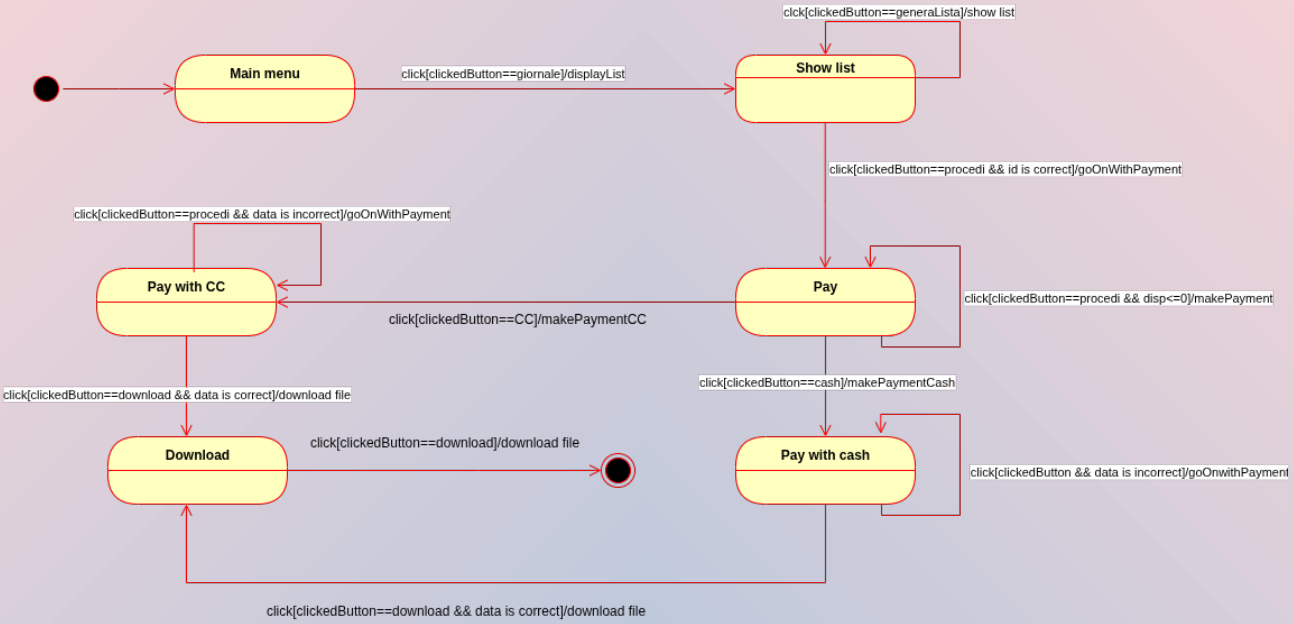
Activity diagram



Sequence Diagram



State diagram



Testing

Test code

Questo primo test va a testare il pagamento con carta di credito di un giornale in tutte le modalità previste da sistema : database, file e memoria

```
@ParameterizedTest  ilciro
@ValueSource(strings = {"database","file","memoria"})
void testAcquistaGiornaleCredito(String strings) throws CsvValidationException, SQLException, IOException, ClassNotFoundException, IdException {
    //inizializzo tabella giornale
    vis.setTypeAsDaily();
    cHP.persistenza(strings);
    //prendo lista oggetti
    cCopravendita.getLista(GIORNALE,strings);
    vis.setIdGiornale(Integer.parseInt(RBOGGETTO.getString( key: "idG")));
    //acquisto
    cA.getPrezzo( q: "2",strings);
    //pagamento cc
    vis.setMetodoP("cCredito");
    cPCC.pagamentoCC(RBUTENTE.getString( key: "nome"),strings);
    //negozio
    cSN.getNegozi(strings);
    boolean status=cSN.isOpen(strings, id: 1)&&cSN.isValid(strings, id: 1);
   assertFalse(status);
}
```

questo secondo invece va ad effettuare il test del pagamento contanti per acquisto del libro , sempre nelle tre modalità

```
@ParameterizedTest  ilciro
@ValueSource(strings = {"database","file","memoria"})
void testAcquistaLibroCash(String strings) throws CsvValidationException, SQLException, IOException, ClassNotFoundException, IdException, DocumentException, URISyntaxException {
    //inizializzo tabella giornale
    vis.setTypeAsBook();
    cHP.persistenza(strings);
    //prendo lista oggetti
    cCopravendita.getLista(LIBRO,strings);
    vis.setIdLibro(Integer.parseInt(RBOGGETTO.getString( key: "idL")));
    //acquisto
    cA.getPrezzo( q: "3",strings);
    //fattura
    vis.setMetodoP("cash");
    cPCash.controlla(RBUTENTE.getString( key: "nome"),RBUTENTE.getString( key: "cognome"),RBUTENTE.getString( key: "via"),RBUTENTE.getString( key: "com"),strings);
    //download
    cD.scarica(LIBRO,strings);
    assertEquals(Integer.parseInt(RBOGGETTO.getString( key: "idL")),vis.getIdLibro());
}
```

infine questo test va a vedere , sempre in tutte le modalità di persistenza se la rivista con id=1 è presente

```
@ParameterizedTest  ilciro
@ValueSource(strings = {"database","file","memoria"})
void testRivistaById(String strings) throws CsvValidationException, IOException, IdException, ClassNotFoundException {
    vis.setIdRivista(1);
    assertEquals( expected: 1,cG.rivistaById(strings).size());
}
```

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-LibreriaMaven/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(lB,"elencoLibriB", lD.getLibri());
    driver.findElement(By.id("idOgg")).sendKeys("2");
    int id=Integer.parseInt(driver.findElement(By.id("idOgg")).getAttribute("value"));
    PropertyUtils.setProperty(lB,"idB",id);
    l.setId((Integer) PropertyUtils.getProperty(lB,"idB"));
    String titolo=lD.getData(l).getTitolo();
    PropertyUtils.setProperty(sB,"idB", id);
    PropertyUtils.setProperty(lB,"idB", id);
    l.setId(id);
    PropertyUtils.setProperty(sB, "titoloB",lD.getData(l).getTitolo());
    PropertyUtils.setProperty(aB, "titoloB",sB.getTitoloB());
    PropertyUtils.setProperty(aB,"prezzoB",lD.getData(l).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 generato da Selenium :

Quest'ultimo invece è il test con codice auto generato 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;
```

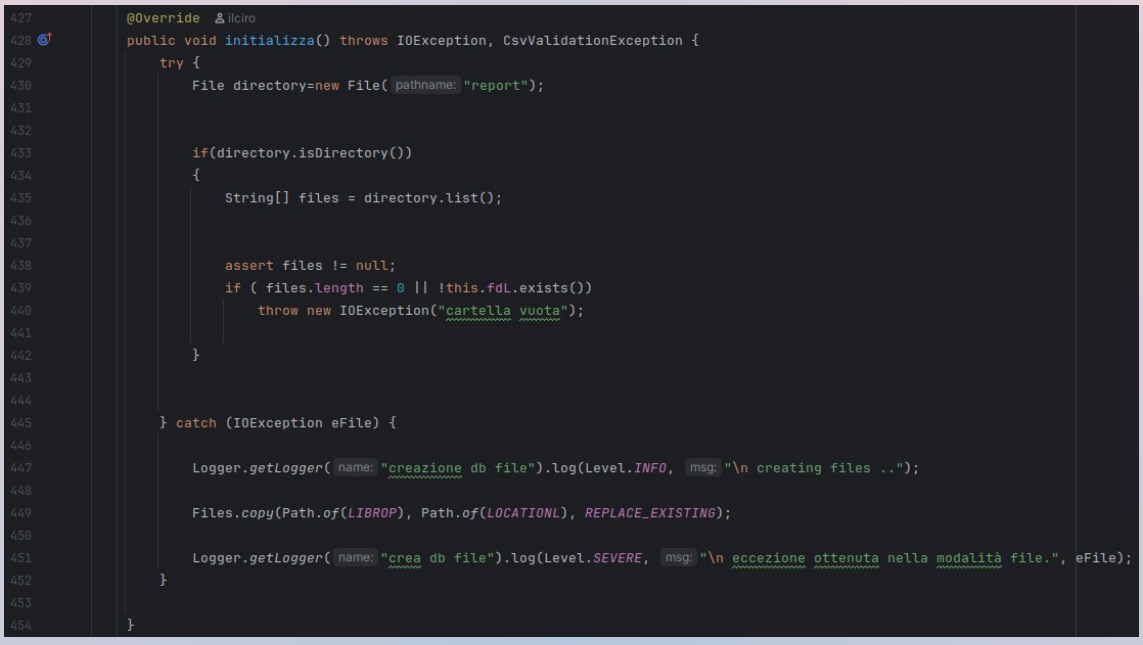
```
private Map<String, Object> vars;
JavascriptExecutor js;
@Before
public void setUp() {
driver = new ChromeDriver();
js = (JavascriptExecutor) driver;
vars = new HashMap<String, Object>();
}
@After
public void tearDown() {
driver.quit();
}
@Test
public void testId() {
driver.get("http://localhost:8080/LibreriaFinale/libri.jsp");
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 :



```
427     @Override  ilciro
428     public void initializza() throws IOException, CsvValidationException {
429         try {
430             File directory=new File( pathname: "report");
431
432
433             if(directory.isDirectory())
434             {
435                 String[] files = directory.list();
436
437
438                 assert files != null;
439                 if ( files.length == 0 || !this.fdL.exists())
440                     throw new IOException("cartella vuota");
441
442             }
443
444
445         } catch (IOException eFile) {
446
447             Logger.getLogger( name: "creazione db file").log(Level.INFO,  msg: "\n creating files ..");
448
449             Files.copy(Path.of(LIBROP), Path.of(LOCATIONL), REPLACE_EXISTING);
450
451             Logger.getLogger( name: "crea db file").log(Level.SEVERE,  msg: "\n eccezione ottenuta nella modalità file.", eFile);
452         }
453
454     }
```

In dettaglio : in questo metodo si va a vedere se la cartella report esiste . Se non esiste oppure è vuota viene lanciata una IOExcpetion con messaggio cartella vuota. Quindi invece che venire gestita dalla jvm è il programmatore che la gestisce : alla riga 446 la jvm stampa un semplice messaggio dicendo che crea il file ; alla 449 si ha la copia dei valori dai singoli file nel file globale (vedasi sotto) e stampa altro messaggio indicando che la copia è avvenuta

Dao

In questa applicazione si usano tre tipologie di persistenza : sia quella su file sia quella su base di dati sia quella in memoria

Database

File system

Per quanto riguarda la persistenza dei file :

- "report/reportLibri.csv";
  - file che mostra il report dei libri
- "report/reportGiornali.csv";
  - file che mostra il report dei giornali
- "report/reportRiviste.csv";
  - file che mostra il report delle riviste
- "report/reportUtenti.csv";
  - file che mostra il report degli utenti
- "report/reportPagamento.csv";
  - file che mostra i pagamenti
- "report/reportFattura.csv";
  - file che mostra le fatture
- "report/reportCartaCredito.csv";
  - file che mostra le carte di credito sia con le quali sono stati effettuati i pagamenti sia quelle possedute da un utente
- "report/reportNegozio.csv";
  - elenco dei negozi disponibili nel sistema
- "report/reportFinale.csv";
  - file nel quale viene generato il report dovuto agli acquisti

Memory

Per quanto la persistenza in memoria , ovvero su file binari che appena si termina applicazione vengono cancellati :

- "report/reportLibri.ser";
  - file che mostra il report dei libri
- "report/reportGiornali.ser";

    - file che mostra il report dei giornali
- "report/reportRiviste.ser";
    - file che mostra il report delle riviste
- "report/reportUtenti.ser";
    - file che mostra il report degli utenti
- "report/reportPagamento.ser";
    - file che mostra i pagamenti
- "report/reportFattura.ser";
    - file che mostra le fatture
- "report/reportCartaCredito.ser";
    - file che mostra le carte di credito sia con le quali sono stati effettuati i pagamenti sia quelle possedute da un utente
- "report/reportNegozio.ser";
    - elenco dei negozi disponibili nel sistema
- "report/reportFinale.ser";
    - file nel quale viene generato il report dovuto agli acquisti

Resources

Qui vengono descritti i file di configurazione per ambiente applicazione.

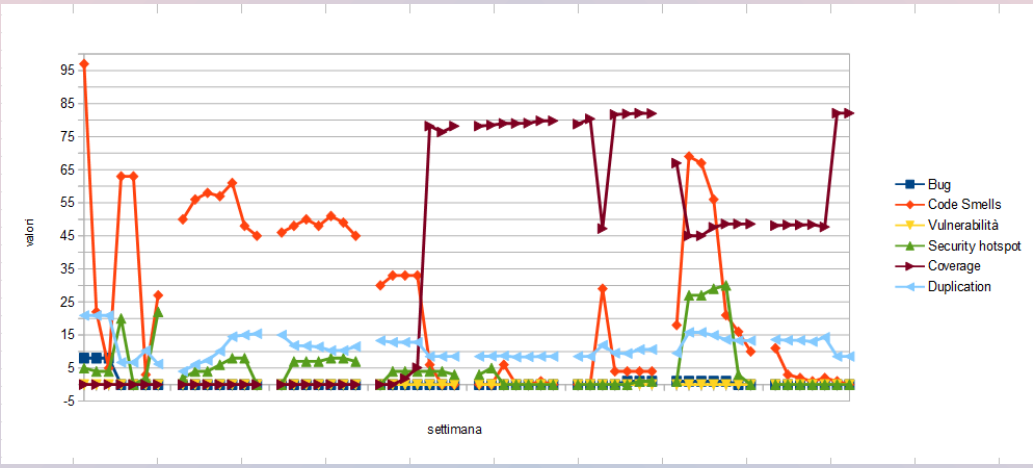
- src/main/resources/configurations/bookCategories.properties
    - elenco delle categorie di libri possibili
- src/main/resources/magazineCategories.properties
    - elenco delle categorie delle riviste possibili
- src/main/resources/titles.properties
    - file contenente i path dove poter scaricare i libri e gli opportui titoli
- src/main/resources/users.properties
    - file contenente info degli utenti per interazione con sistema
- src/main/resources/cartaCredito.properties
    - file contenente dati per carte credito
- src/main/resources/configDb.properties
    - file contenente le credenziali per accedere al database
- src/main/resources/objects.properties
    - file contenente vari dati riguardanti i vari libri, giornali e riviste

- src/main/resources/csvfile/fattura.csv
  - file contenente fatture inizialmente vuoto che verrà copiato se si usa la persistenza file
- src/main/resources/csvfile/giornale.csv
  - file contenente giornali che verrà copiato se si usa la persistenza file
- src/main/resources/csvfile/libro.csv
  - file contenente libri che verrà copiato se si usa la persistenza file
- src/main/resources/csvfile/negozio.csv
  - file contenente negozi che verrà copiato se si usa la persistenza file
- src/main/resources/csvfile/pagamento.csv
  - file contenente pagamenti inizialmente vuoto che verrà copiato se si usa la persistenza file
- src/main/resources/csvfile/rivista.csv
  - file contenente riviste che verrà copiato se si usa la persistenza file
- src/main/resources/csvfile/utente.csv
  - file contenente utenti che verrà copiato se si usa la persistenza file
- src/main/resources/sql/dbCreate.sql
  - file contenente query per il database appena creato e per settaggio
- src/main/resources/sql/tableCreate.sql
  - file contenente tabelle da creare per il database
- src/main/resources/sql/tableExists.sql
  - file contenente funzione in sql che permette di vedere se tabella esiste
- src/main/resources/sql/tablePopulate.sql
  - file contenente valori da inserire nel database
- src/main/resources/tmpFiles/cartaCredito.txt
  - file numerati da 1 a 6 che servono per caricare i dati delle carte di credito nella persistenza in memoria
- src/main/resources/tmpFiles/fattura1.txt
  - file che serve per caricare i dati della fattura nella persistenza in memoria
- src/main/resources/tmpFiles/giornale.txt
  - file numerati da 1 a 12 che servono per caricare i dati dei giornali nella persistenza in memoria
- src/main/resources/tmpFiles/libro.txt

    - file numerati da 1 a 19 che servono per caricare i dati dei libri nella persistenza in memoria
- src/main/resources/tmpFiles/negozio.txt
    - file numerati da 1 a 4 che servono per caricare i dati dei negozi nella persistenza in memoria
- src/main/resources/tmpFiles/pagamento1.txt
    - file che serve per caricare i dati del pagamento nella persistenza in memoria
- src/main/resources/tmpFiles/report.txt
    - file numerati da 1 a 3 che servono per caricare i dati del report nella persistenza in memoria
- src/main/resources/tmpFiles/rivista.txt
    - file numerati da 1 a 5 che servono per caricare i dati delle riviste nella persistenza in memoria
- src/main/resources/tmpFiles/utente.txt
    - file numerati da 1 a 7 che servono per caricare i dati degli utenti nella persistenza in memoria
- src/main/resources/tmpFiles/view.txt
    - cartella contenente tutte le interfacce grafiche del sistema con eventuali immagini

Sonar Cloud

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 81%:



Per quanto riguarda il link al repo è il seguente :

https://sonarcloud.io/summary/overall?id=ilciro_BookShopOnline25&branch=master

Video

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

https://github.com/ilciro/BookShopOnline25