ASTE ONLINE



**INDICE**

1. Installazione progetto
2. Installazione del database Mysql
3. DATABASE
4. Analisi/Sviluppo
5. Diagramma delle classi
6. Diagramma di dominio
7. Diagramma dei casi d’uso
8. Diagramma di sequenza
9. Diagramma dell’architettura
10. Diagramma stati
11. Diagramma attività
12. Analisi SonarQube
13. Analisi Understand

1.INSTALLAZIONE PROGETTO

Nel Repository su GitHub sono presenti 2 versioni del progetto finale: uno con solo il codice senza alcun commento personale (AsteOnlineSenzaComm), e l’altro con vari commenti personali che spiegano le azioni che vengono effettuate nella maggior parte del codice (AsteOnlineConComm).

I commenti personali sono tutti quelli scritti in italiano.  
I commenti in inglese sono stati generati automaticamente da NetBeans (ambiente di sviluppo utilizzato).

Noi abbiamo testato con SonarQube la versione senza commenti personali (AsteOnlineSenzaComm).

Si possono utilizzare entrambe le versioni per verificare il funzionamento del programma.  
Per importarlo bisogna scaricare tutto il repository in una cartella sul desktop tramite la gitBash e scegliere una delle due versioni da Utilizzare.  
Per il corretto funzionamento del programma bisogna andare su Eclipse ed importare il progetto (come progetto maven esistente) selezionando la cartella Asteonline che si troverà dentro la cartella AsteOnlineSenzaComm o AsteOnlineConComm(a seconda della propria scelta).  
Dopo l’importazione del codice bisogna aggiungere la libreria per la connessione al database Mysql selezionando il file Jar (Connector J8.0-> mysql-connector-java-8.0.19 ) se non è già presente.  
  
In quasi tutte le classi JAVA , in cima alle classi, ci sono le variabili in cui sono inseriti: il percorso, la password e lo username del Database.

String idU = "root";

String pwd = "ciao";

String url = "jdbc:mysql://localhost:3306/astedb?useTimezone=true&serverTimezone=UTC";

Se i dati del proprio database sono diversi da quelli riportati qua sopra bisogna andare a modificarli in tutte le classi con il proprio username(idU) e password(pwd) della propria MYSQL CONNECTION.  
L’url dovrebbe essere uguale, ma nel caso in cui non lo sia bisogna aggiornare anche quello tenendo sempre la parte ?useTimezone=true&serverTimezone=UTC.

Per avviarlo selezionare la classe AsteOnline.java e avviare il programma (bottone verde di avvio).

Per simulare il funzionamento di un’asta abbiamo inserito un numero massimo di offerte (scelto dall’utente che crea l’asta o impostato dall’amministratore -Vedi sezione 3.DATABASE-) associate all’asta che sostituisce il timer.

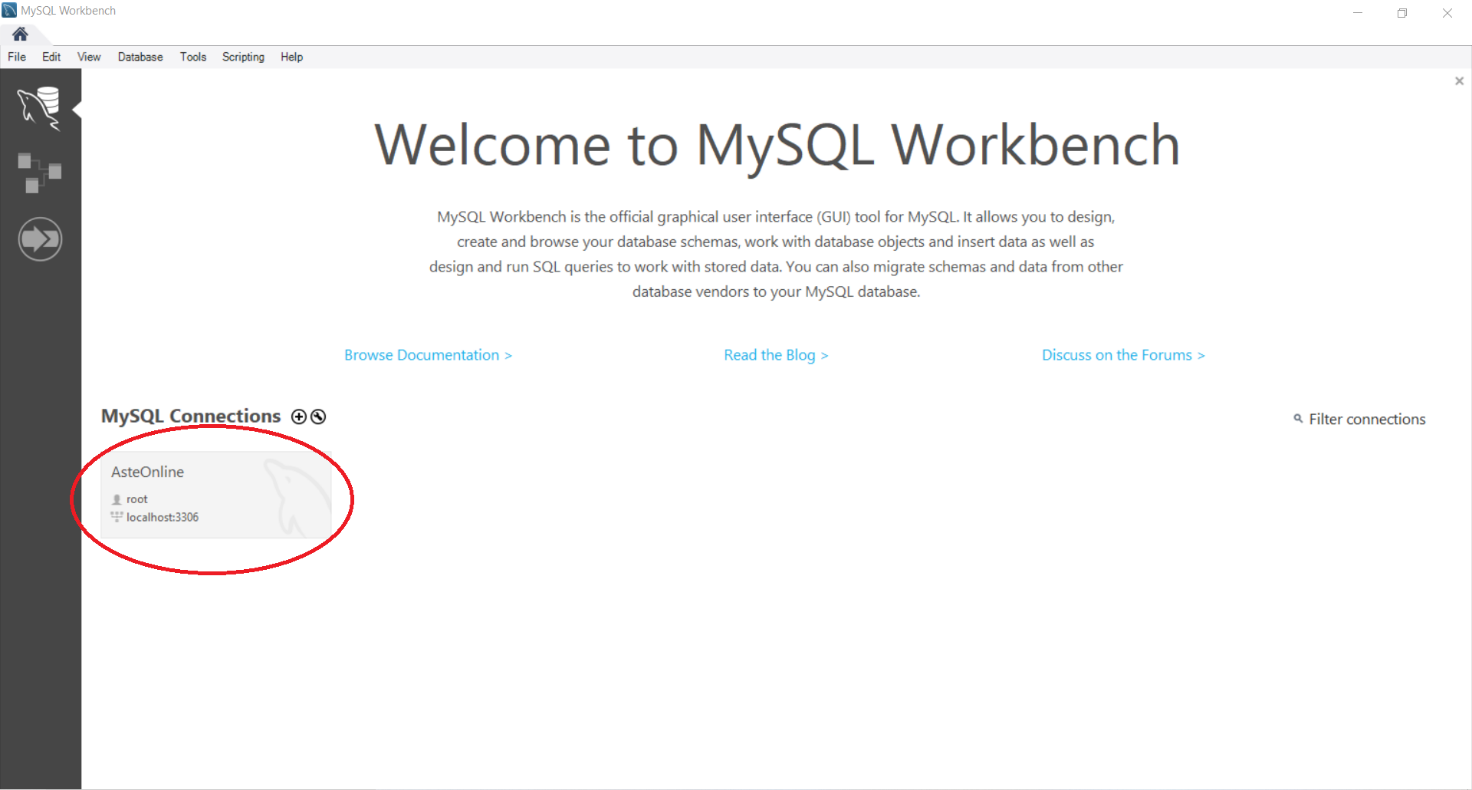
Per simulare la contesa di un oggetto in un’asta e quindi effettuare la partecipazione di più utenti ad un’asta bisogna effettuare una (o più) offerta/e utilizzando un account e dopo bisogna effettuare il LOGOUT ed accedere con un altro account il quale selezionando la stessa asta di prima potrà partecipare anch’esso a quell’asta e quindi contendersi l’oggetto con gli altri utenti, altrimenti è libero di partecipare ad un’asta diversa.

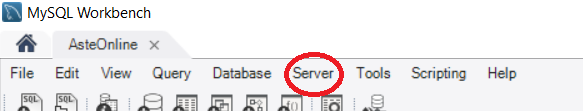
Abbiamo messo il vincolo che un utente può partecipare ad una sola asta alla volta e può effettuare anche più offerte consecutivamente a prescindere dal tipo di asta a cui sta partecipando.

Se l’utente che sta partecipando ad un’asta effettua il LOGOUT potrà verificare lo stato della propria asta al prossimo LOGIN premendo sul bottone “Visualizza Asta a cui stai partecipando” e se è ancora in corso ti riporterà alla pagina dell’asta, altrimenti se è finita ti porterà alla pagina dove ti mostrerà il risultato dell’asta a cui stava partecipando.  
Nel caso di vittoria ti salverà il saldo disponibile decrementato dell’offerta massima effettuata, invece in caso di perdita dell’asta il saldo torna ad essere quello prima di aver effettuato le offerte.

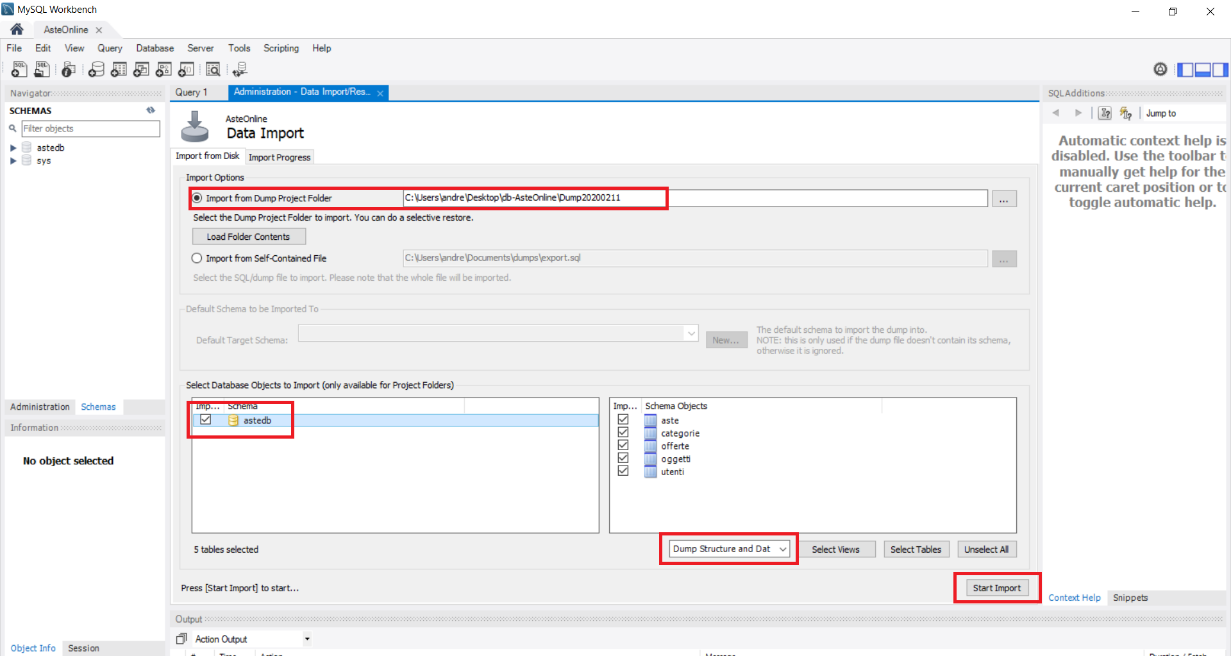
2.INSTALLAZIONE DATABASE MYSQL

Nel repository sarà presente una cartella chiamata db-AsteOnline nella quale ci sarà la cartella nominata “Dump20200211”.  
Per poterla utilizzare bisogna andare su MYSQL Workbench, accedere al proprio Server locale (root, localhost:3306)

  
e premere su Data Import presente nella sezione Server.



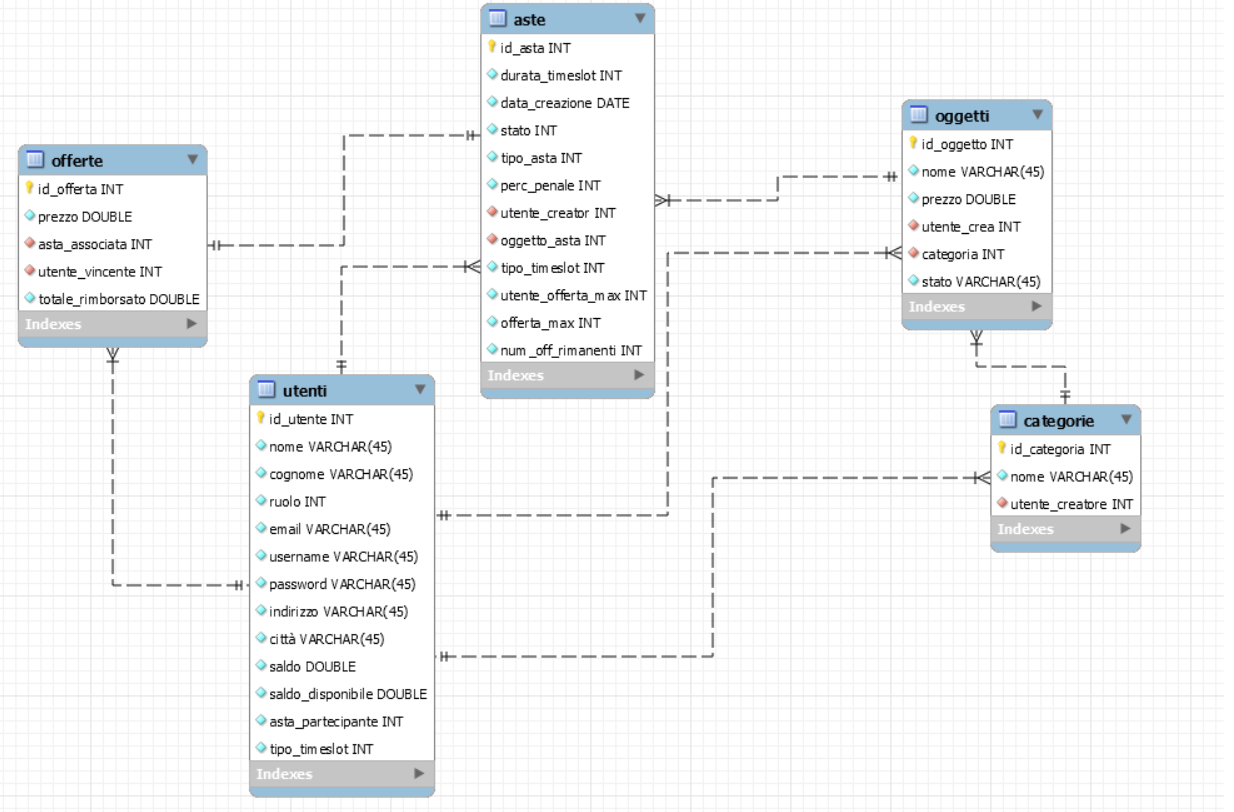
Dopo di che bisogna selezionare l’impostazione “Import From Dump Project Folder”, selezionare il percorso fino alla cartella “Dump20200211”, mettere la spunta su “astedb” nella sezione “Select DataBase Objects to import” e selezionare l’opzione “Dump Structure and Data” nel menu a tendina accanto al bottone “Select Views”.

  
Infine premere su Start Import.

3.DATABASE

Nel database fornito sono già presenti dei dati che non devono essere eliminati.

Nel Database saranno presenti 5 tabelle:  
 1- utenti  
 2- aste  
 3- oggetti  
 4- categorie  
 5- offerte



**ASSOCIAZIONI**

-L’associazione **oggetti-aste** è UNO a MOLTI in quanto una volta che un utente ha eseguito il reso, tale oggetto può essere rimesso in un’altra asta dal suo proprietario originale.  
Invece ad un’asta può essere associato un solo oggetto.

-L’associazione **utenti-oggetti** è UNO a MOLTI in quanto un utente può creare più oggetti e più oggetti possono appartenere ad un solo utente.

-L’associazione **utenti-categorie** è UNO a MOLTI in quanto un utente può creare più categorie e più categorie possono essere associate a un solo utente (creatore).

-L’associazione **categorie-oggetti** è UNO a MOLTI in quanto una categoria può contenere più oggetti e più oggetti posso essere associati ad un'unica categoria.

-L’associazione **utenti-offerte** è UNO a MOLTI in quanto un utente può fare più offerte e più offerte possono appartenere ad un solo utente.

-L’associazione **utenti-aste** è UNO a MOLTI in quanto un utente può creare più aste mentre più aste possono appartenere ad un solo utente.

-L’associazione **offerte-aste** è UNO a UNO in quanto abbiamo deciso di salvare solo le offerte vincenti associate alla propria asta, quindi un offerta può essere associata a solo un’asta.

**SPIEGAZIONE DI ALCUNI CAMPI IMPORTANTI NELLE TABELLE**

-Nella tabella **Utenti**:

-la colonna **ruolo** indica se un utente è amministratore o un utente registrato.  
 0 = utente registrato, 1 = amministratore.

-la colonna **tipo\_timeslot** serve ad indicare se un amministratore di sistema ha deciso di impostare un vincolo   
 sul numero di puntate massime che si possono effettuare in un’asta.  
 Quel campo verrà modificato solo nella riga dell’amministratore “Pecos” e se sarà impostato a 0, gli utenti   
 avranno la possibilità di scegliere il numero massimo di puntate per la propria asta da un elenco di scelte   
 multiple(Combo box), altrimenti il numero massimo di puntate per asta assumerà il valore scelto   
 dall’amministratore che sarà uguale al numero presente nella colonna tipo\_timeslot.

-la colonna **saldo\_disponibile** inizialmente sarà uguale alla colonna saldo (che contiene il saldo massimo) e   
 successivamente verrà modificato e aggiornato a seconda delle offerte che farà l’utente nelle aste.  
 Se alla fine un utente vince un’asta la colonna saldo verrà aggiornata col valore di saldo\_disponibile (che sarà   
 stato decrementato per via delle offerte effettuata durante l’asta), altrimenti la colonna saldo\_disponibile   
 sarà aggiornata col valore della colonna saldo(che sarà rimasta invariata durante le offerte effettuate in   
 un’asta).

-la colonna **email** e **username** non possono contenere duplicati.

-Nella tabella **Oggetti**:

-la colonna **stato** indica se l’oggetto è nuovo/pronto per essere messo in un’asta (settato a 0), o se è già   
 stato venduto/già presente in un’asta (settato a 1).

-la colonna **nome** non può contenere duplicati.

-Nella tabella **Aste**:

-la colonna **durata\_timeslot** indica il numero massimo di puntate selezionate per quell’asta.

-la colonna **stato** indica se l’asta:  
 - è stata creata e nessuno ha ancora fatto un’offerta (settato a 0).  
 - è in esecuzione, quindi qualcuno ha fatto un’offerta (settato a 1).  
 - è terminata (settato a 2).  
 - è in stato rimborso effettuato (settato a 3).

-la colonna **tipo\_aste** indica se la puntata è aperta (settato a 0), quindi l’utente può vedere l’offerta massima   
 attuale, o a busta chiusa (settato a 1), quindi l’utente potrà vedere solo la propria offerta corrente.

-la colonna **tipo\_timeslot** indica se il numero di puntate massimo è stato scelto dall’utente (settato a 0) o se è   
 stato impostato dall’amministratore (settato a 1).

-Nella tabella Offerte:

-la colonna **totale \_rimborsato** indica il valore del rimborso che è stato effettuato all’utente.

4.ANALISI/SVOLGIMENTO

Inizialmente avevamo deciso di sviluppare un’applicazione WEB con l’utilizzo di un database chiamato Postgre SQL, Javascript, le servlet e i file HTML/CSS per gestire la parte grafica.  
Nella nostra prima fase di progettazione avevamo creato i diagrammi richiesti basandoci sulla nostra idea iniziale andando ad implementare tre diversi Desing Pattern:   
 -il **MVC** (Model-View-Controller) con pattern architetturale Template View;  
 -lo **Strategy**   
 -l’**Intercepting Filter**

I giorni successivi alla creazione di tali diagrammi abbiamo iniziato ad informarci sul come sviluppare il progetto tramite l’utilizzo del nuovo database e dei nuovi linguaggi per lo sviluppo: JavaScript, HTML/CSS (conosciuti poco da tutto il gruppo).  
Poco dopo abbiamo deciso di cambiare il database e di utilizzare MYSQL Workbench che era più comodo.  
Purtroppo per mancanza di tempo non siamo riusciti ad informarci abbastanza per permetterci di sviluppare l’applicazione come avevamo pensato inizialmente.  
Questo ci ha portato a modificare drasticamente tutto il progetto dovendo trovare un altro metodo per lo sviluppo, ripartendo dalla fase di progettazione.   
In questa seconda fase abbiamo deciso di sviluppare solo in Java e di utilizzare NetBeans per lo sviluppo in quanto semplifica la creazione della parte grafica.  
Come database abbiamo utilizzato MYSQL Workbench.  
I nuovi diagrammi sono completamente diversi da quelli creati nella prima fase di progettazione e non implementano alcun Desing Pattern.  
Con questo nuovo metodo siamo riusciti ad ultimare il progetto con le seguenti caratteristiche:

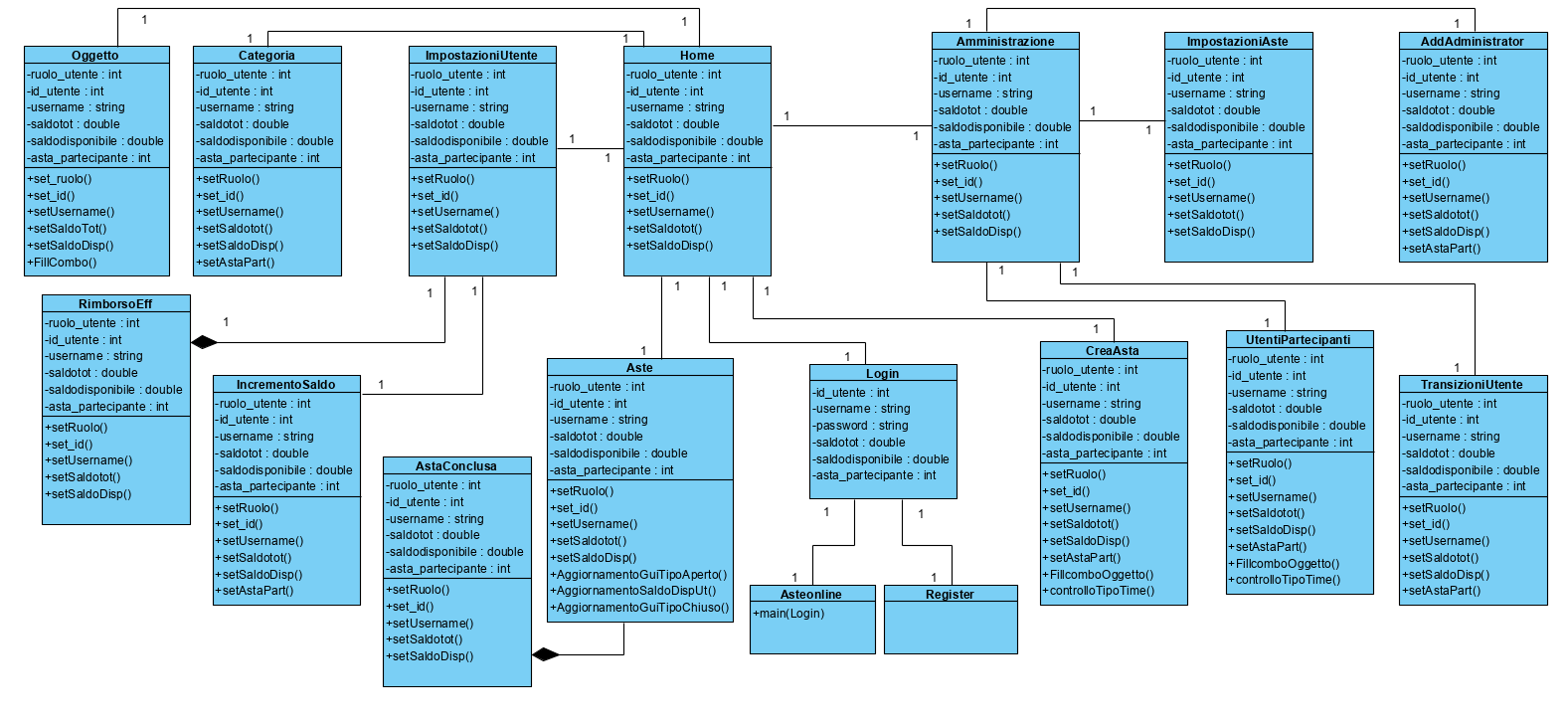
-Un utente, appena avvierà l’applicazione, potrà eseguire solamente il Login oppure Registrarsi nel caso in cui non abbia un Account.  
Una volta effettuato il Login un utente potrà svolgere diverse attività:  
 1- visualizzare/creare Categorie  
 2- visualizzare/creare Oggetti  
 3- Creare nuove Aste inerenti ai propri oggetti creati  
 4- Visualizzare le Aste in corso  
 5- Partecipare ad un’asta  
 6- Fare più offerte   
 7- Visualizzare le proprie Aste Vinte  
 8- Richiedere il Rimborso  
 9- Incrementare il proprio saldo

Se l’utente è un amministratore, oltre a poter svolgere tutte le attività degli utenti normali potrà:  
 1- Modificare le impostazioni dell’asta (Numero offerte massime in un’asta)  
 2- Visualizzare l’elenco degli Utenti che stanno partecipando ad un’asta  
 3- Rendere Amministratori altri utenti  
 4- Visualizzare tutte le transazioni degli utenti (offerte vincenti con eventuale rimborso)

Un utente può partecipare solo ad un’asta alla volta.

5.DIAGRAMMA DELLE CLASSI

Il diagramma delle Classi finale è il seguente



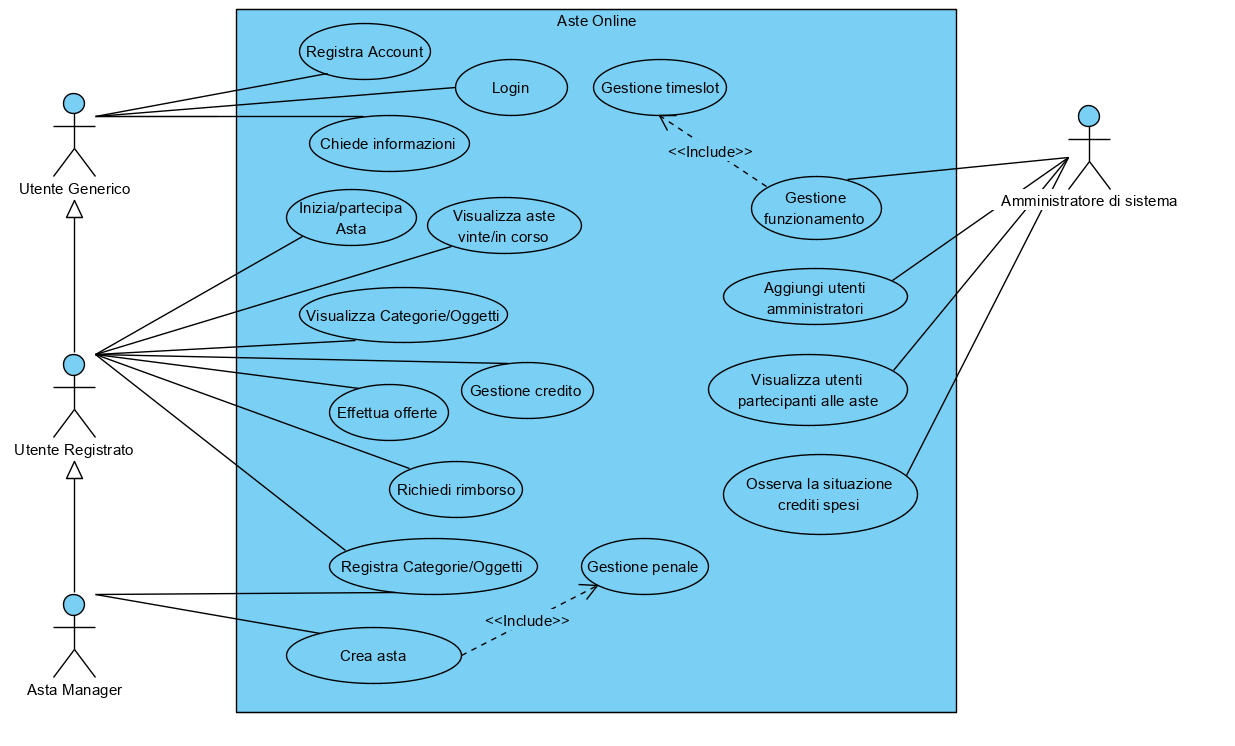
La classe Asteonline è il main dove non è presente alcun metodo in quanto serve solo per aprire la pagina Principale (Login).  
In tutte le classi sono presenti solo i metodi set per i loro attributi, in quanto le azioni che può svolgere un utente sono gestite degli eventi dei bottoni.  
Gli unici metodi diversi dai vari set sono i metodi FillCombo… che servono per inserire i valori nelle combobox presenti in alcune pagine.

6.DIAGRAMMA DI DOMINIO

Immagine che contiene parete, interni, segnale

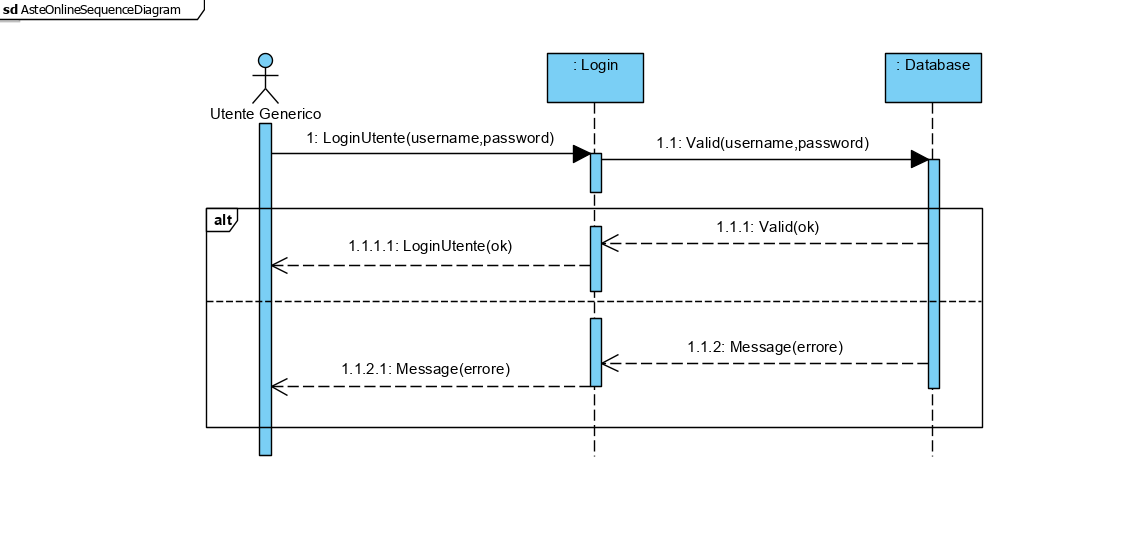
Descrizione generata automaticamente

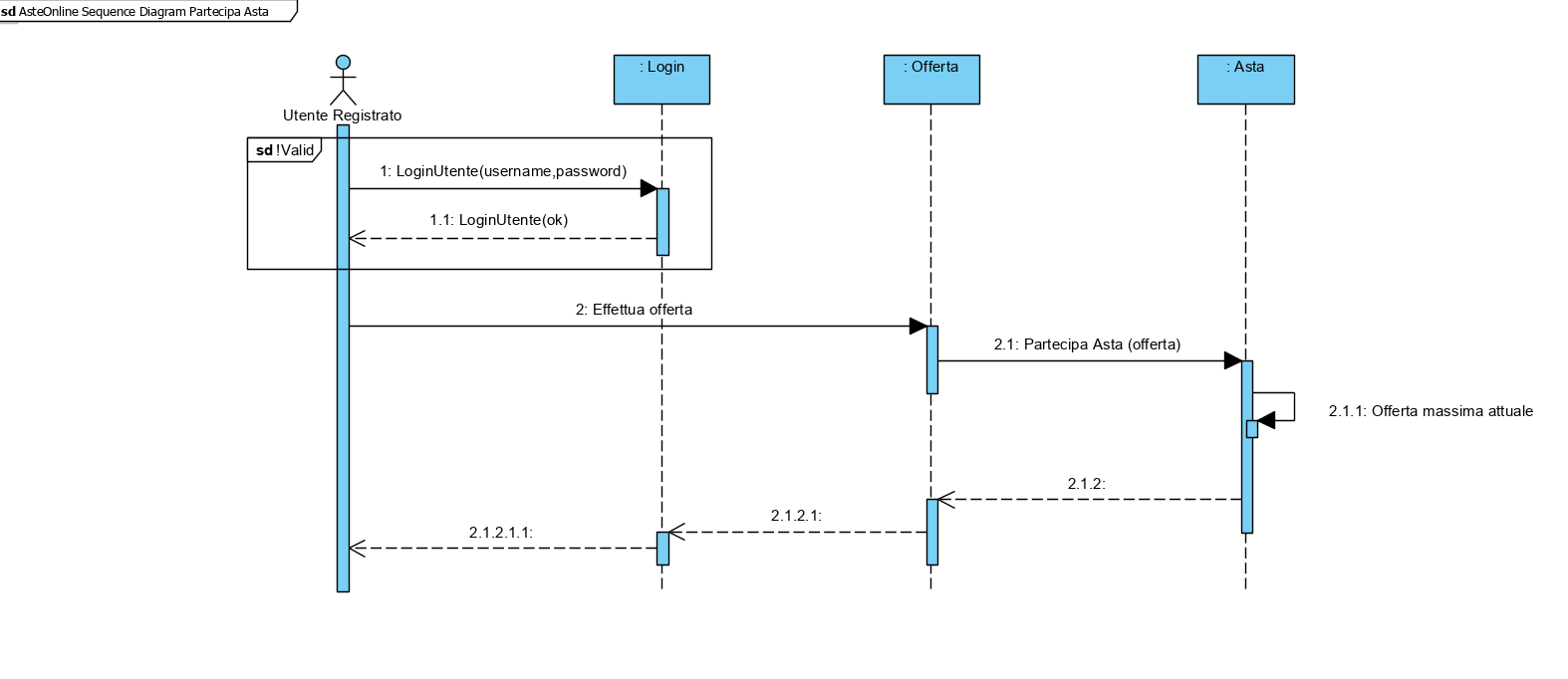
7.DIAGRAMMA DEI CASI D’USO



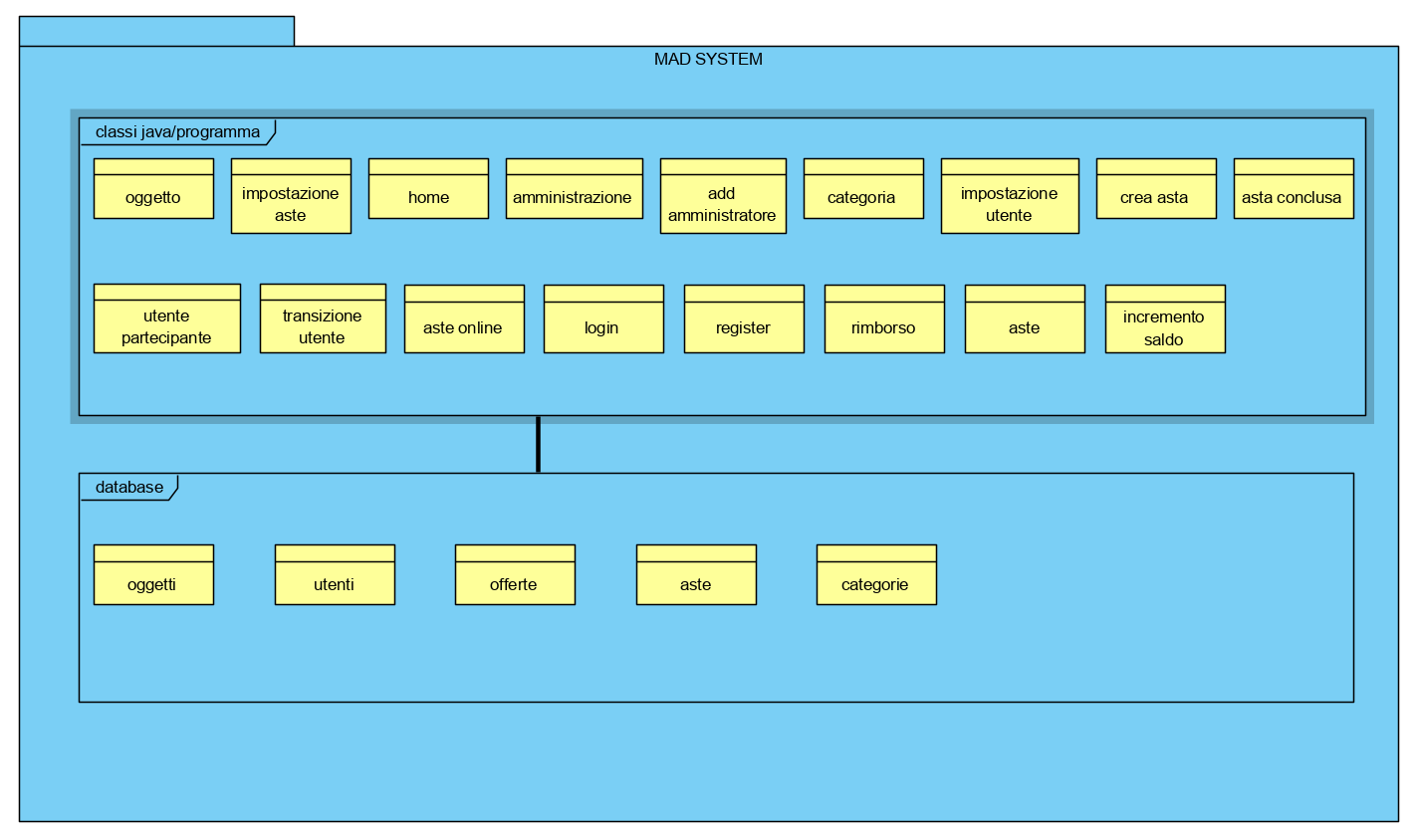
Abbiamo presupposto che l’amministratore di sistema possa svolgere anche tutte le attività degli altri utenti, oltre a quelle dedicate solo a lui.

8.DIAGRAMMA DI SEQUENZA

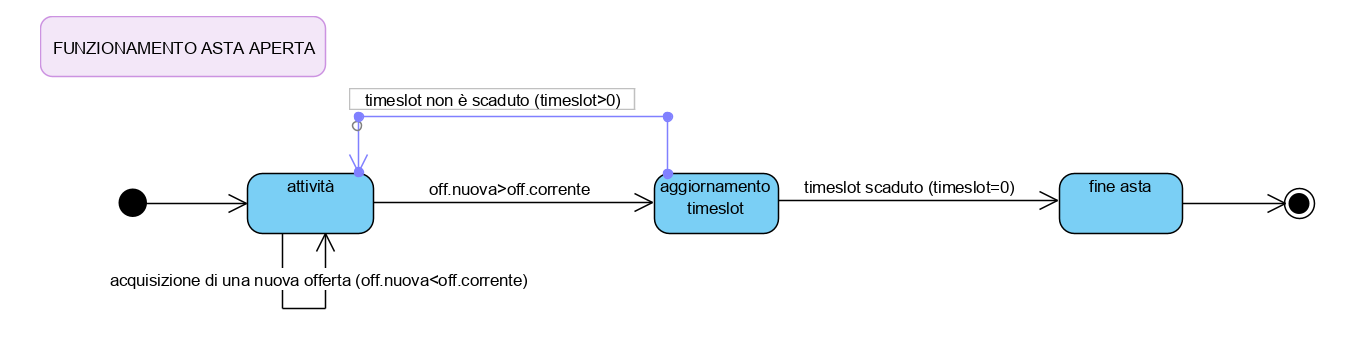


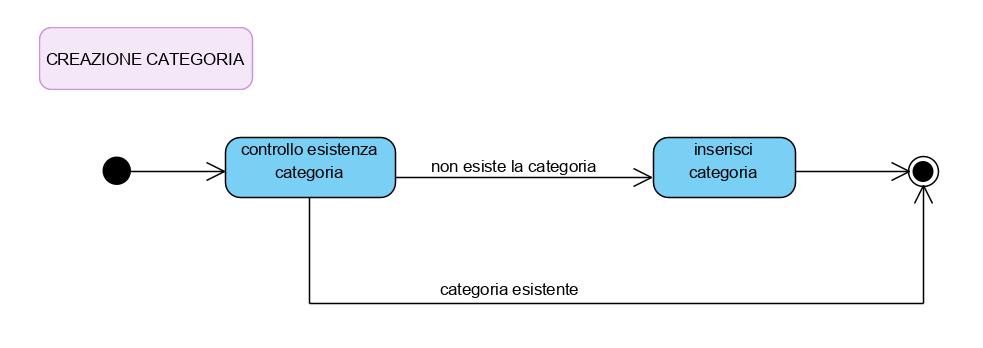


9.DIAGRAMMA DELL’ARCHITETTURA

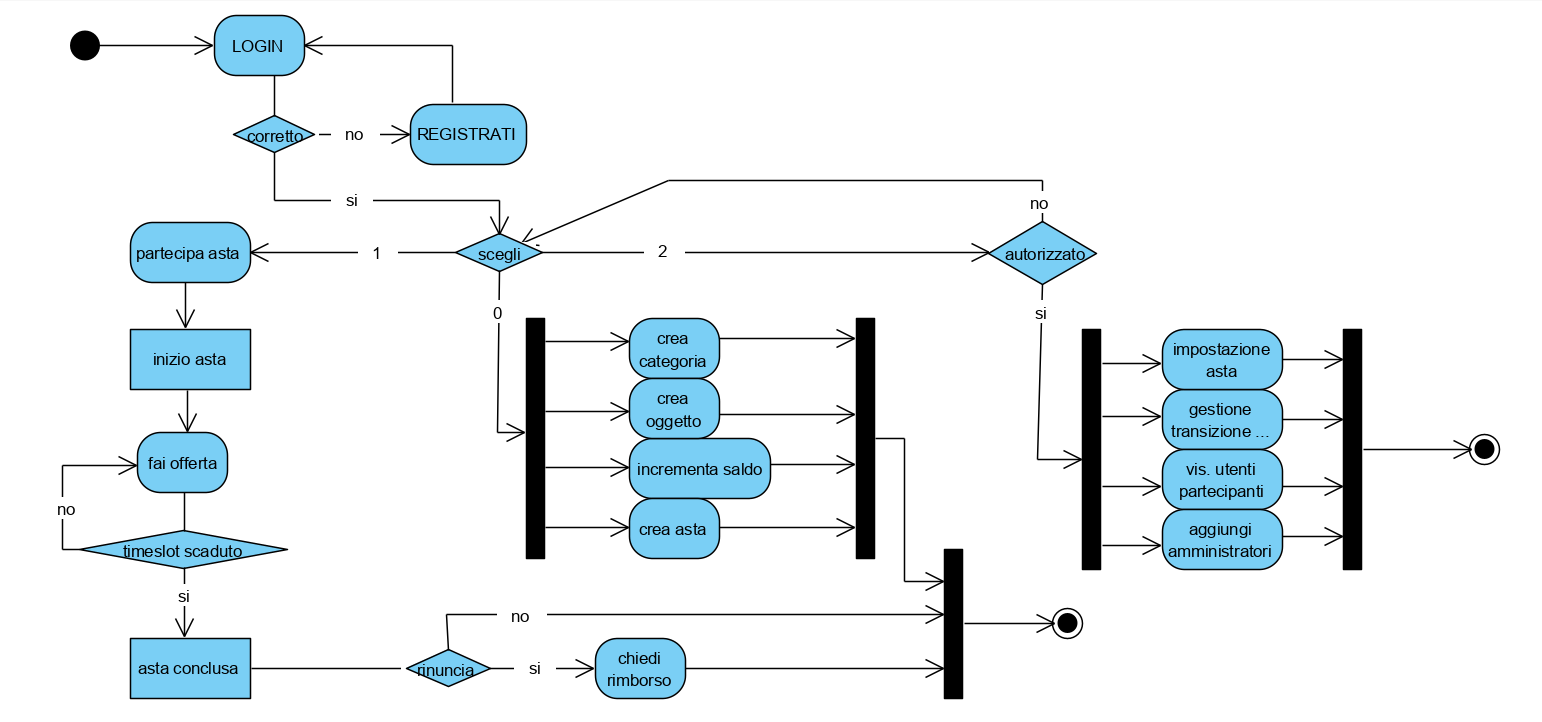


10.DIAGRAMMA DEGLI STATI





11.DIAGRAMMA DELLE ATTIVITA’

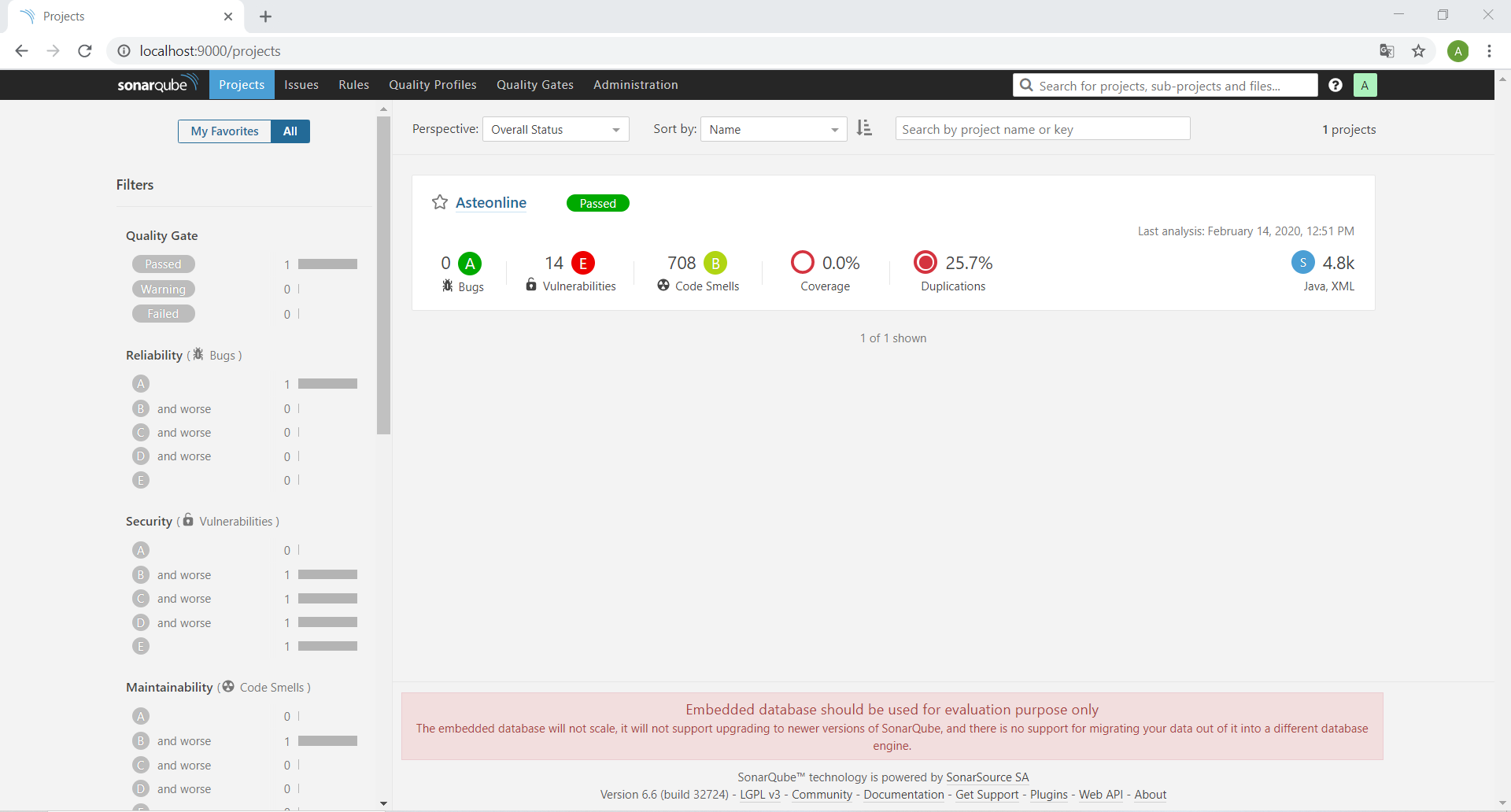


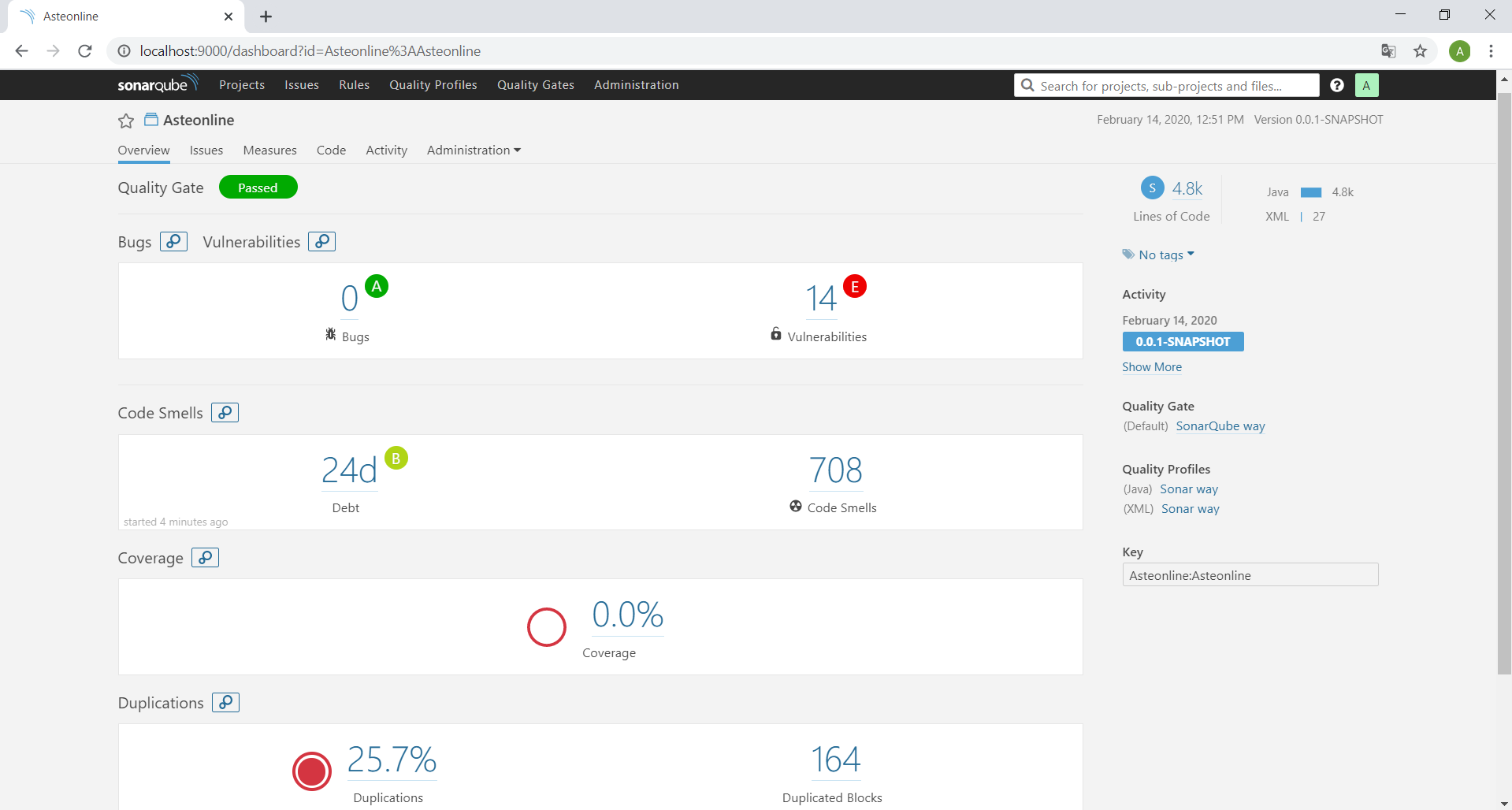
Il controllo “autorizzato” verifica se l’utente è un amministratore.

12.ANALISI SONARQUBE

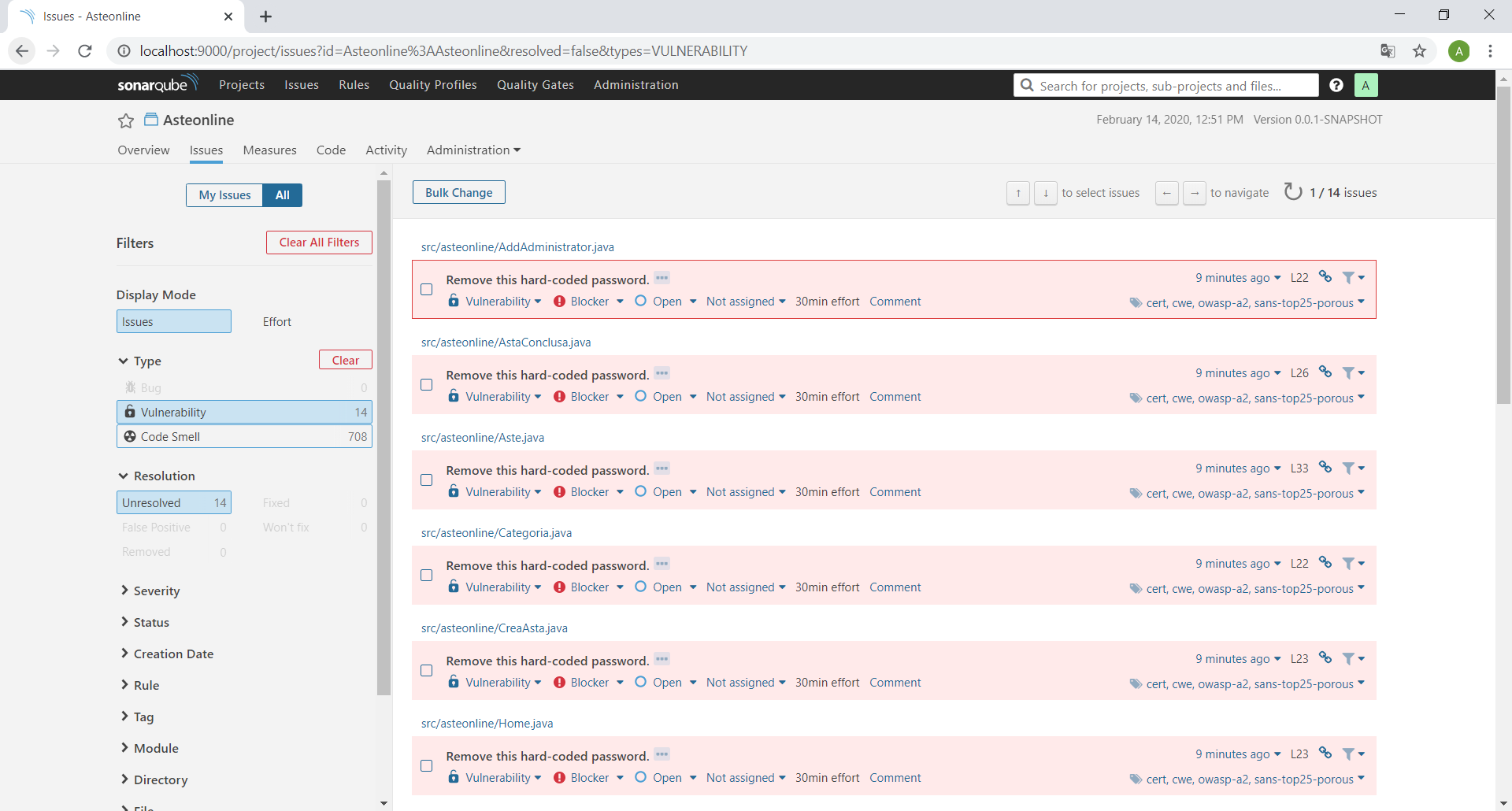
Abbiamo analizzato il codice senza commenti.

L’analisi del progetto ci ha dato i seguenti risultati:





Nel campo VULNERABILITIES otteniamo una E con 14 vulnerabilità segnalate.  
Tutte queste segnalazioni indicano tutte le classi in cui setto nel codice la password per accedere al database, ed essendo essenziale per il corretto funzionamento del programma (senza quel campo non posso accedere al db mysql) le ho considerate come falsi positivi ottenendo una A.



Nel campo CODE SMELLS otteniamo una B e la maggior parte delle violazioni che risultano sono di tipo major e minor ma anche queste le ritengo abbastanza inutili come segnalazioni, in quanto:

1- la maggior parte delle segnalazioni major indicano che è presente un parametro nei metodi usati per far funzionare i bottoni che non viene mai richiamato(ma se tolgo quel parametro il bottone non funziona)  
2-quasi tutte le segnalazioni minor sono del tipo cambia la variabile id\_utente in idutente oppure indica di modificare il controllo if(condizioni &&  isNum == true  && altre condizioni) in  if(condizioni &&  isNum  && altre condizioni).  
In più risultano alcune violazioni CRITICAL che indicano solo due tipi di casi, che anche questi a mio avviso non sono così importanti:

1-mi segna come critical ogni volta che in una classe trova più query dove all'interno trova scritto, ad esempio "ruolo", solo che "ruolo" è il nome di una colonna nel database e quindi è normale che possa essere presente in più query nella stessa classe.

2-mi segna critical la dichiarazione di alcune variabili che mi servono per poter operare col database come:

       Connection conn = **null**;

     Statement stm = **null**;

     ResultSet rs = **null**;

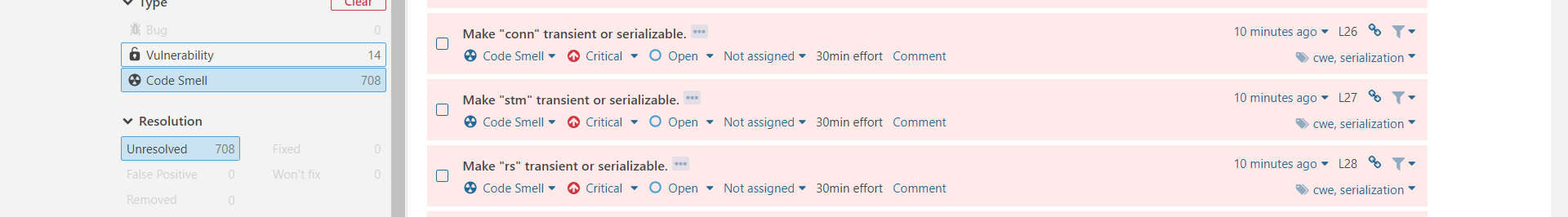
dandomi:

**Make "conn" transient or serializable.**

**Make "stm" transient or serializable.**

**Make "rs" transient or serializable.**

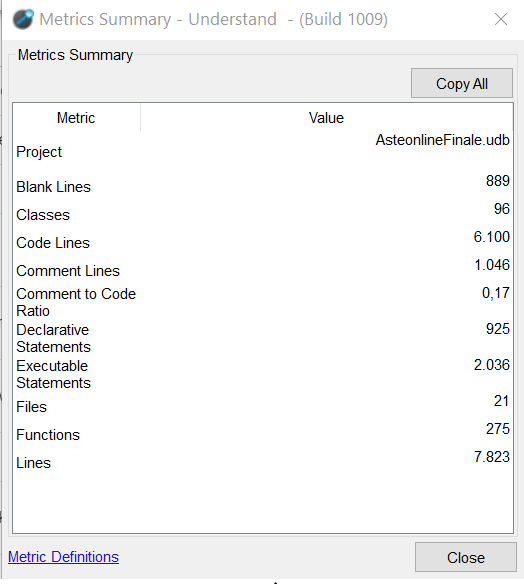
Solo che queste dichiarazioni che ho messo nel codice non potranno mai dare problemi in quanto la loro vera inizializzazione avviene nel costruttore della classe dove sono dentro dei try-catch e quindi non potranno dare problemi.  
(quelle dichiarazioni scritte sopra sono messe prima del costruttore in modo tale che sono salvate in modo globale per la classe).

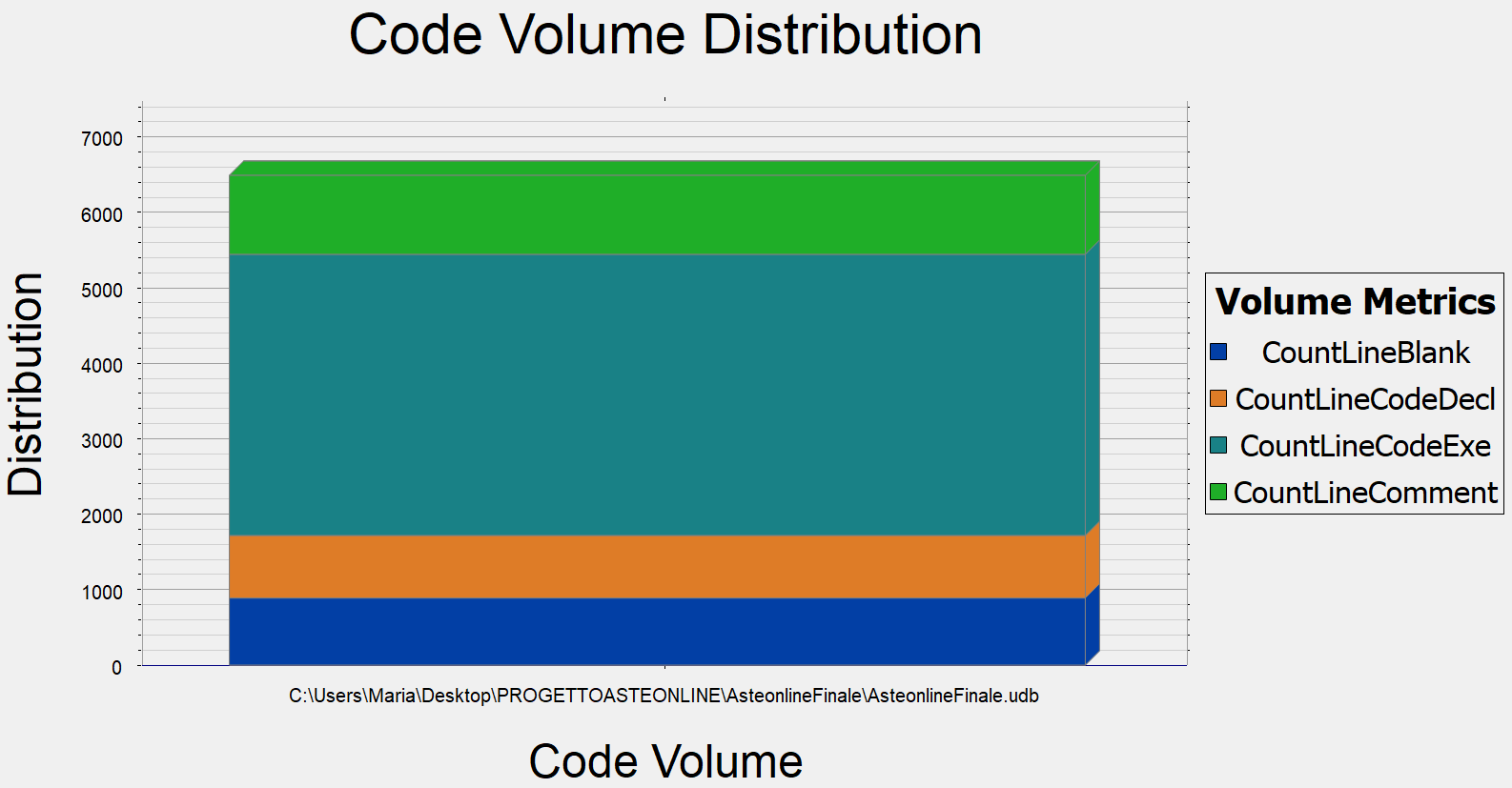


A seguito dei ragionamenti esposti precedentemente ho deciso di considerarli come "resolve as won't fix".

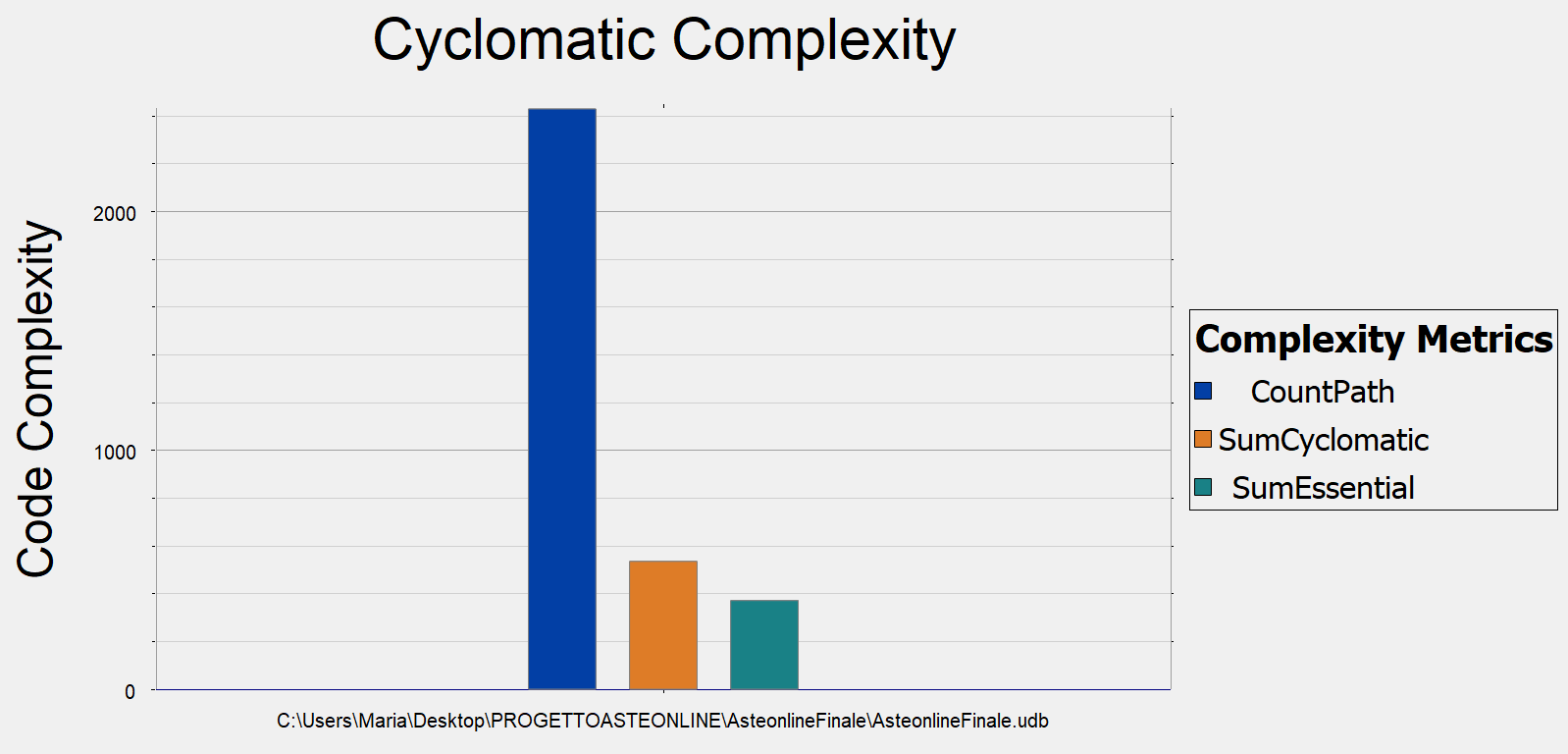
13.ANALISI UNDERSTAND

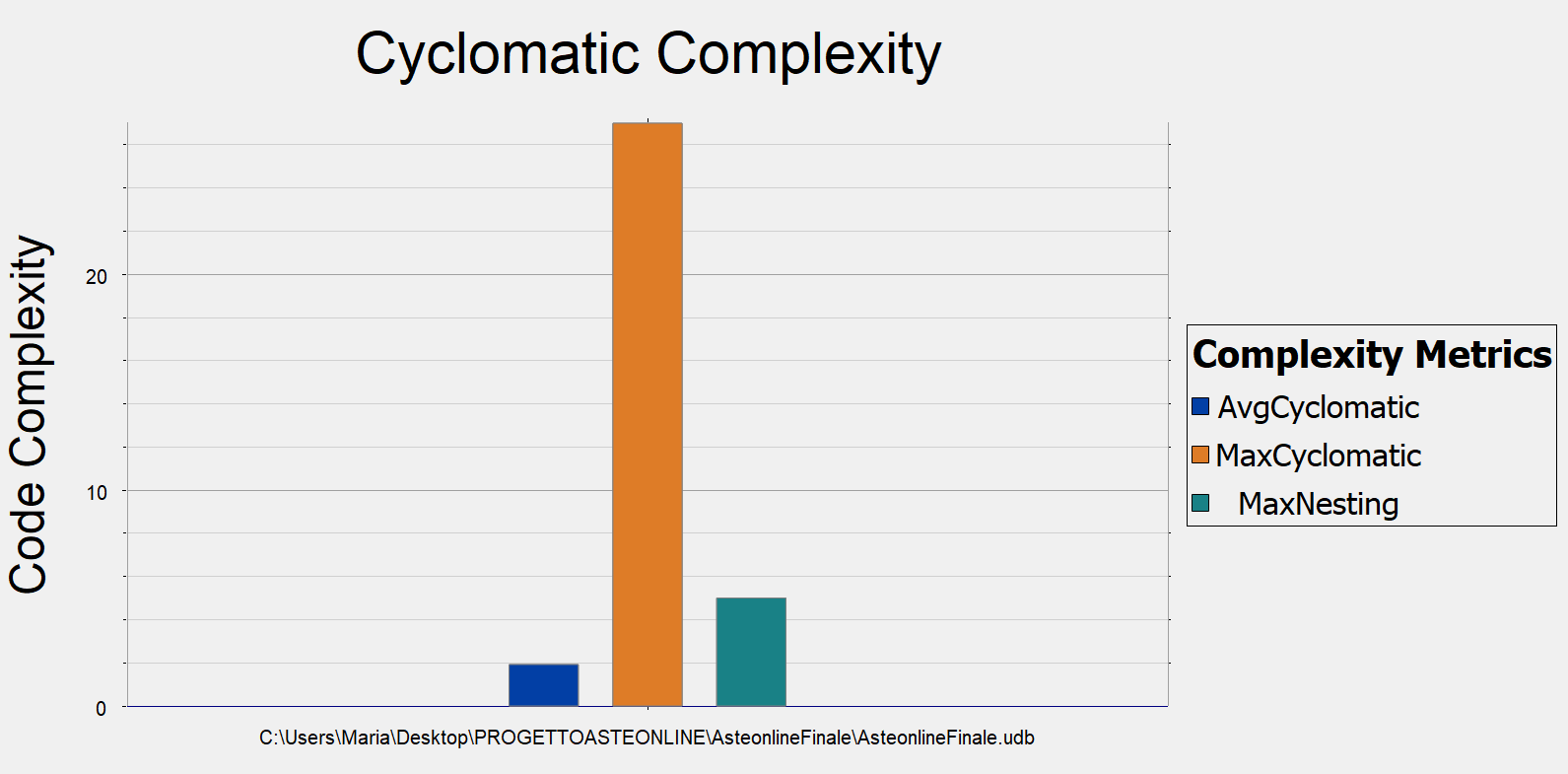
Per analizzare il nostro progetto è stato utilizzato Understand, con il quale abbiamo effettuato analisi di diverse Metriche:

**Metrics Summary:**

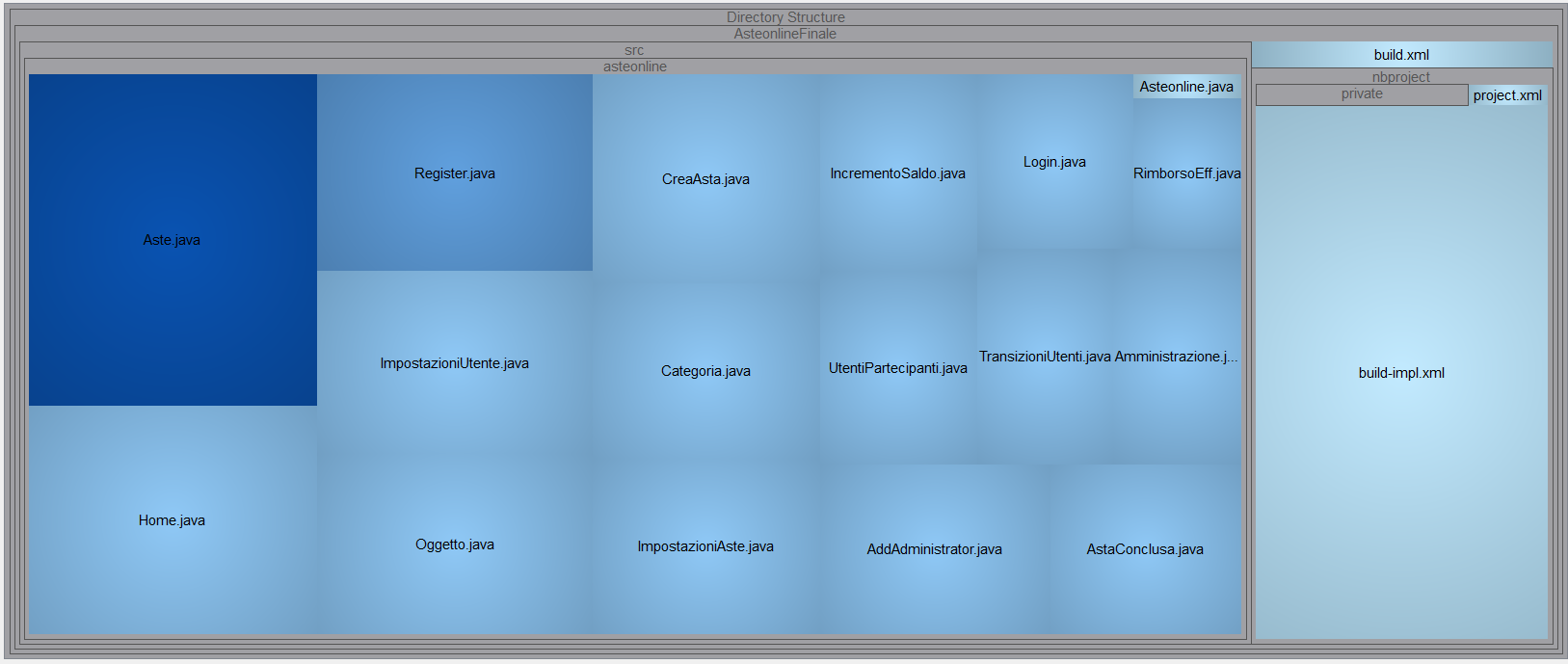


Dal grafico prodotto dalla Metrica **Code Volume Distribution**, si nota un alto numero di linee contenente codice sorgente eseguibile(CountLineCodeExe), rispetto alle altre.

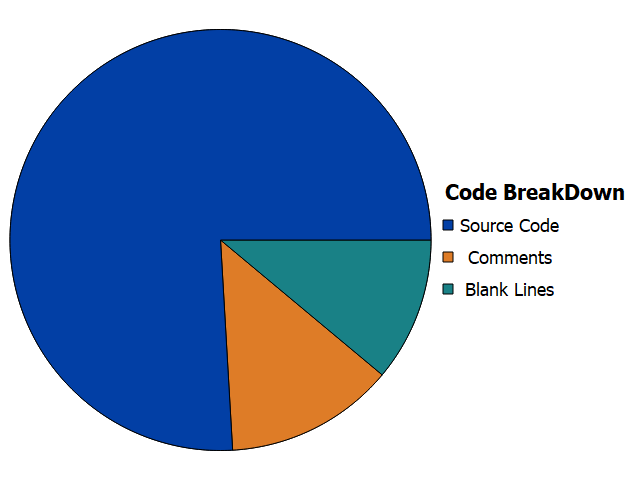


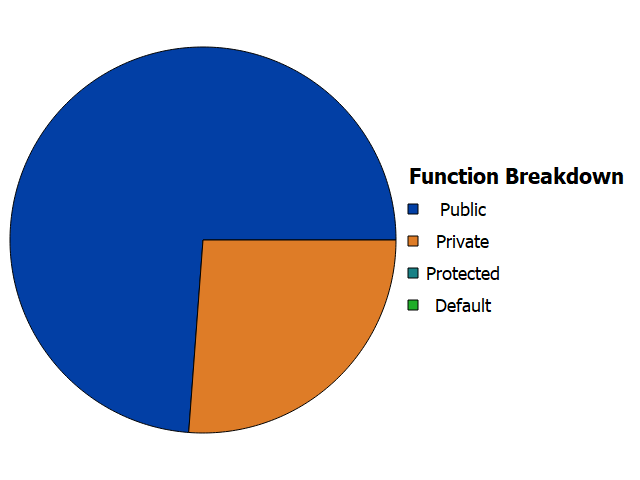
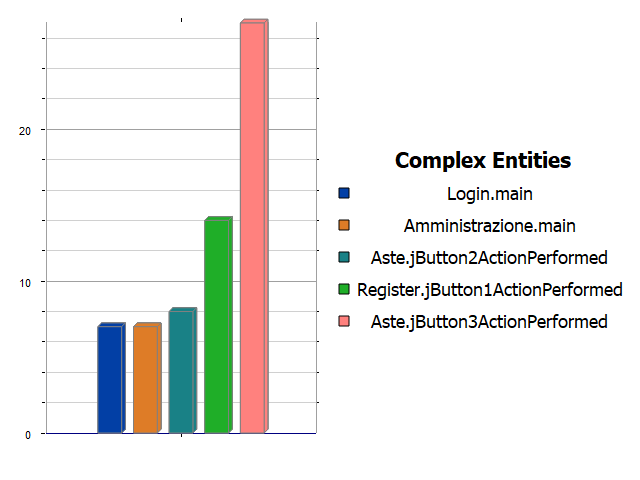


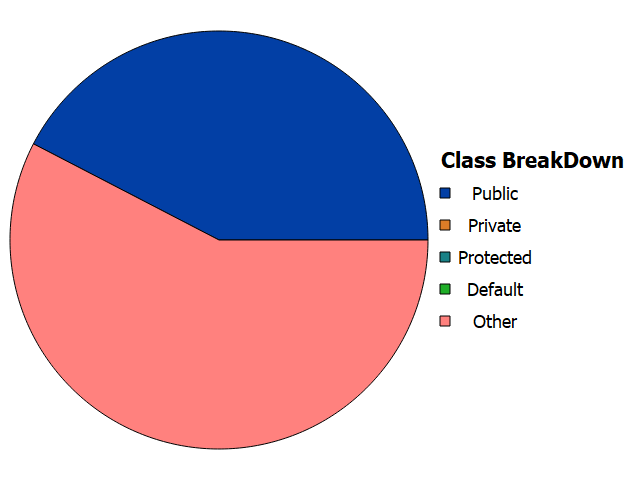
Da questo grafico notiamo che ci sono classi che sono molto più complicate della media.

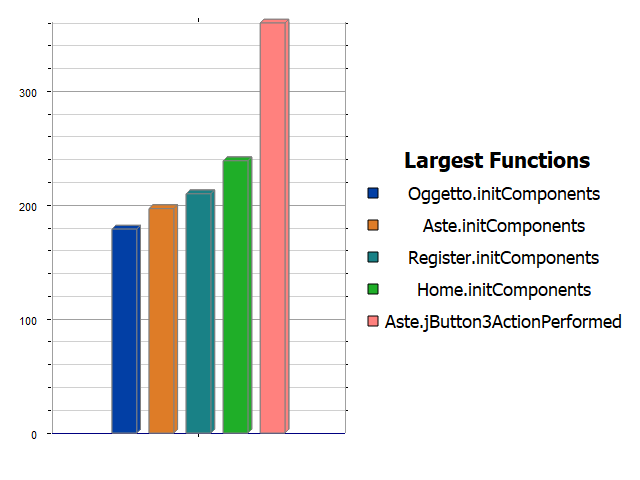
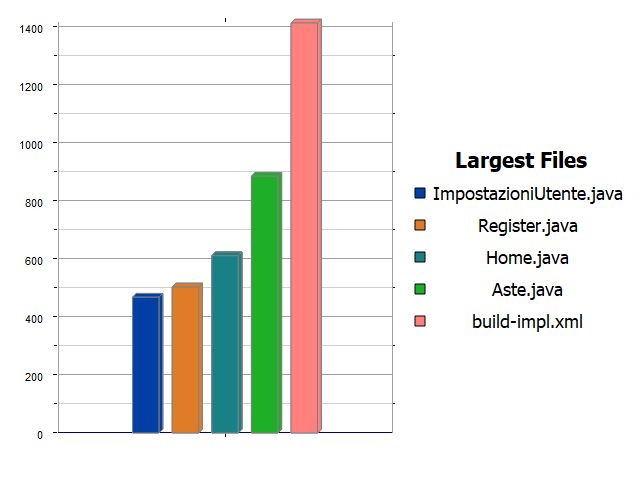
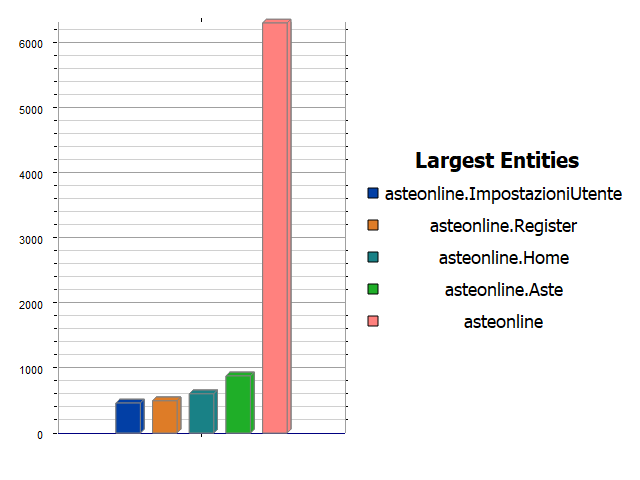
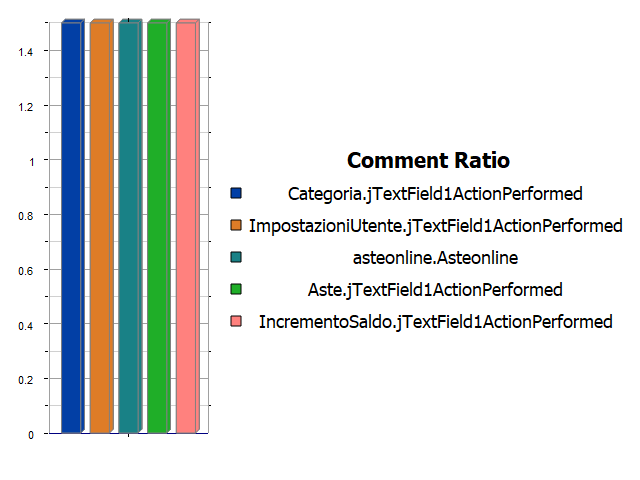
**Metric TreeMap**

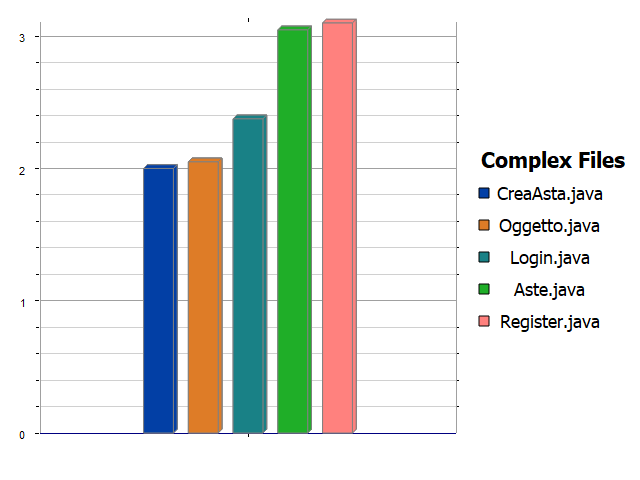
Dal risultato della Metric TreeMap, con parametri il numero di linee per classe e la massima complessità ciclomatica, si nota che la classe più complessa è proprio Aste.java, ovvero la classe più grande.

**Project Overview Charts**







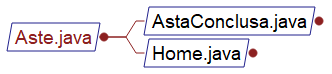


**Alcuni esempi di grafi relativi a classi:**

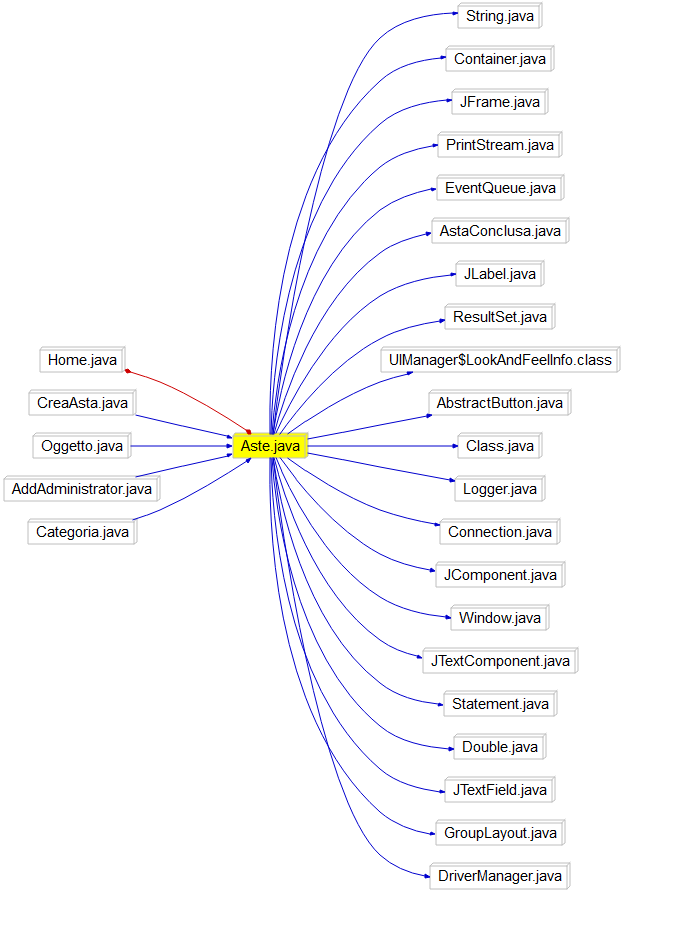
Depended On By: Aste.java



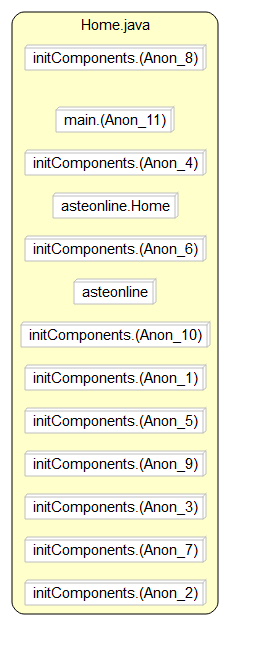
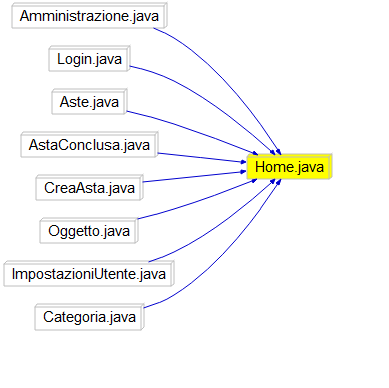
Depends On: Aste.java



Cluster Call Butterfly: Aste.java



Cluster Call Internal: Home.java Cluster Callby: Home.java



**Report Generato AsteOnline :** [**AsteonlineFinale\AsteonlineFinale.txt**](AsteonlineFinale/AsteonlineFinale.txt)

**CodeCheck AsteOnline:** [**CodecheckResult.csv**](CodecheckResult.csv)

Progetto realizzato da:  
Cilento Maria Matricola: 833235  
Pecoriello Andrea Matricola: 833008  
Danny Dominguez Matricola: 822446