MONEY ART

The art to make money

GRUPPO C02

TEAM

Stefano ZARRO
0512107273

Daniele GALLOPPO
0512106955

Aurelio SEPE
0512108638

Michael DE SANTIS
0512109736

Alfonso CANNAVALE
0512108068

Mario PELUSO
05212107078

Dario MAZZA
0512106742

Nicolò DELOGU
0512106214

Schedule

01
Architettura Del Sistema

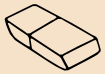
02
Object Design

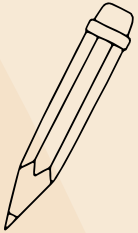
03
Testing

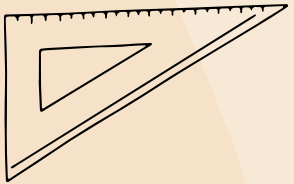
04
Lessons Learned

01

Architettura Del Sistema

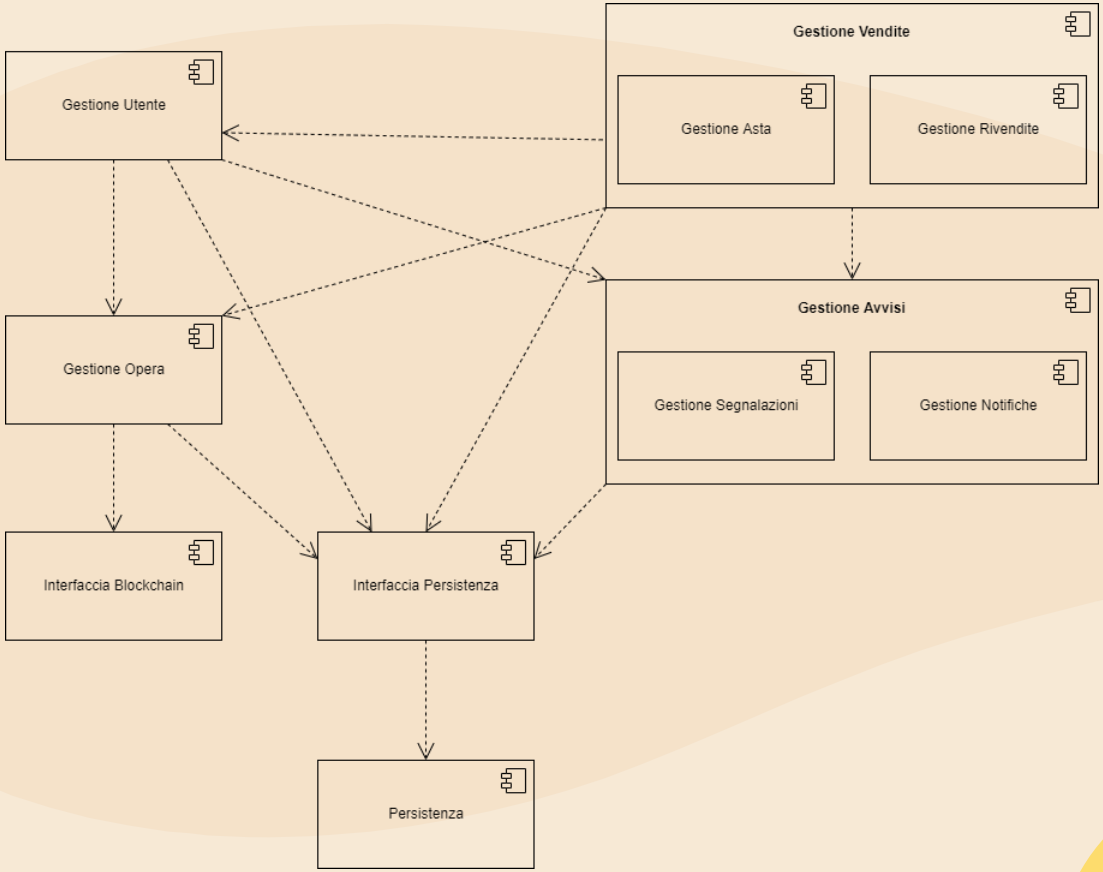




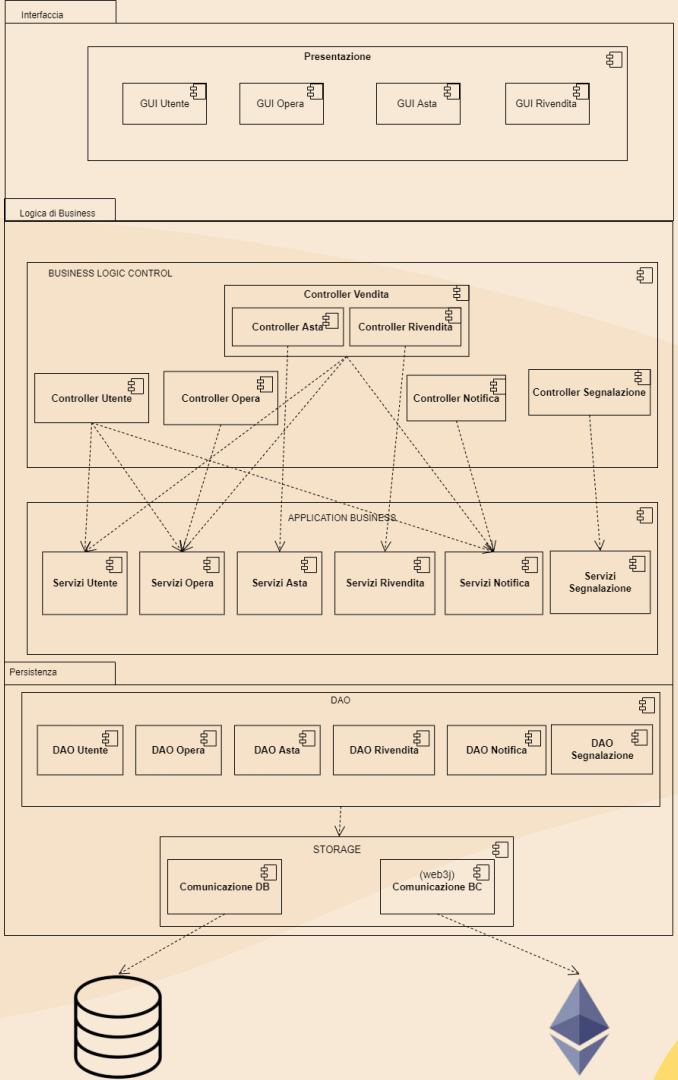


DECOMPOSIZIONE SOTTOSISTEMI

Una visione dall'alto



THREE-TIER

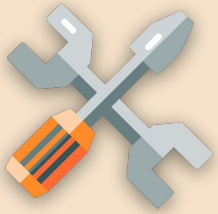

Interfaccia
Presentazione
GUI Utente
GUI Opera
GUI Asta
GUI Rivendita
Logica di Business
BUSINESS LOGIC CONTROL
Controller Vendita
Controller Asta
Controller Rivendita
Controller Utente
Controller Opera
Controller Notifica
Controller Segnalazione
APPLICATION BUSINESS
Servizi Utente
Servizi Opera
Servizi Asta
Servizi Rivendita
Servizi Notifica
Servizi Segnalazione
Persistenza
DAO
DAO Utente
DAO Opera
DAO Asta
DAO Rivendita
DAO Notifica
DAO Segnalazione
STORAGE
Comunicazione DB
(web3j)
Comunicazione BC

02

Object Design

OBJECT DESIGN GOALS

SECURITY



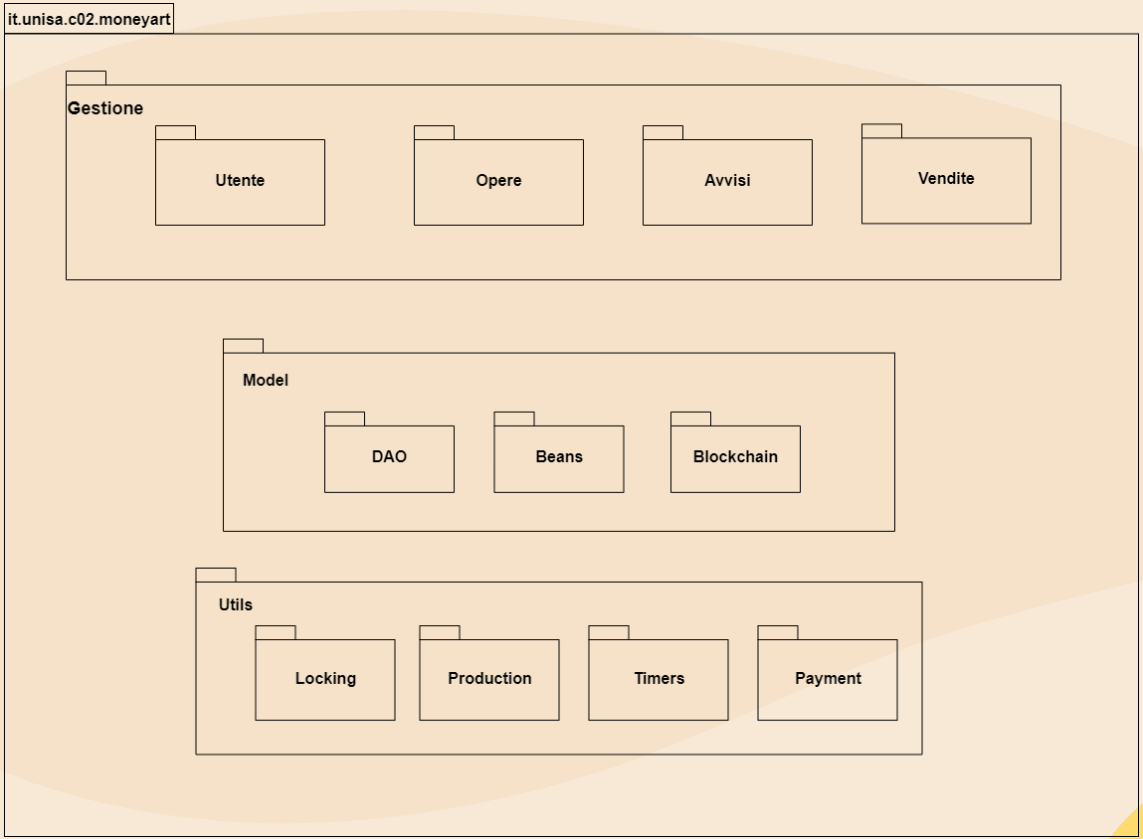
MAINTAINABILITY

ROBUSTNESS

USABILITY

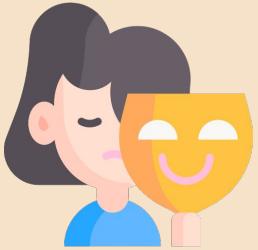
PERFORMANCE

PACKAGES


it.unisa.c02.moneyart
Gestione
Utente
Opere
Avvisi
Vendite
Model
DAO
Beans
Blockchain
Utils
Locking
Production
Timers
Payment

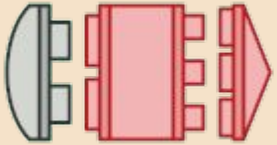
DESIGN PATTERNS

FACADE

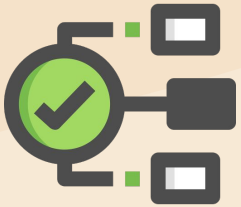


DAO

ADAPTER



SINGLETON



DESIGN PATTERNS (1)

FAÇADE

Pattern Strutturale

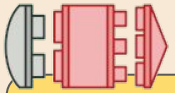
Utilizzato per astrarre dalla complessità dei sottosistemi fornendo un'interfaccia unica di accesso.

DATA ACCESS OBJECT

Pattern Strutturale

MoneyArt ha un'architettura three tier. Il pattern DAO è ideale per separare il livello di logica di business dal livello di persistenza.

DESIGN PATTERNS (2)



ADAPTER

Pattern Strutturale

Utilizzato per l'incapsulamento delle librerie esterne di PayPal e Stripe. Consente la fruizione del loro gateway di pagamento.

SINGLETON

Pattern Creazionale

Consente di mantenere uno stato globale per gestire l'eventuale concorrenza di molteplici offerte

03

Testing

TECNOLOGIE UTILIZZATE

JUnit 5

MOCKITO

Se

SELENIUM IDE

TESTING

UNIT

Tramite l'utilizzo di **mockito** e **junit**

JUnit 5

INTEGRATION

Strategia ***bottom-up***

SYSTEM

Black-box category partition **selenium** ide

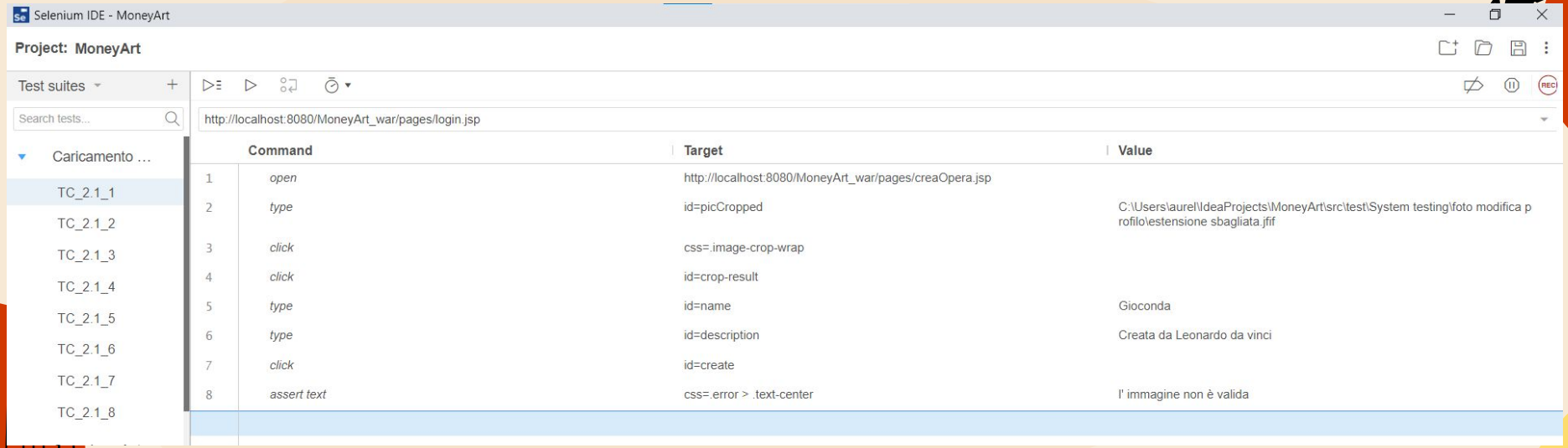
ESEMPIO – UNIT

```java
@DisplayName("Participate in an ongoing auction with no bids and a single user and a valid offer")
        @Test
        void participateOngoingAuctionWithNoBidsAndASingleUserAndAValidOfferTest() {
            Asta asta = AstaCreations.ongoingAstaLastingSevenDaysWithNoArtistFollowersAndNoBids();

            Utente u9 = new Utente();
            u9.setId(9);
            u9.setSaldo(500d);

            double offerta = 150d;



            // recupero versione aggiornata dei dati
            when(utenteDao.doRetrieveById(anyInt())).thenReturn(u9);
            when(astaDao.doRetrieveById(anyInt())).thenReturn(asta);

            // recupero lista partecipazioni per ricavare la migliore offerta

when(partecipazioneDao.doRetrieveAllByAuctionId(anyInt())).thenReturn(asta.getPartecipazioni());

            Partecipazione nuovaOfferta = new Partecipazione(asta, u9, offerta);

            // aggiornamento dati persistenti
            when(partecipazioneDao.doCreate(nuovaOfferta)).thenReturn(true);
            doNothing().when(utenteDao).doUpdate(u9);

            boolean result = astaService.partecipateAuction(u9, asta, offerta);

            Assertions.assertEquals(350d, u9.getSaldo());
            Assertions.assertEquals(true, result);
        }
```

ESEMPIO - INTEGRATION

```java
@Test
    public void offertaRiuscitaNuovoMiglioreOfferente() {
      Asta asta = astaDao.doRetrieveById(2);
      Utente utente = utenteDao.doRetrieveById(2);
      Utente utente2 = utenteDao.doRetrieveById(3);

      double saldoPrecedente = utente.getSaldo();
      int notifichePrima = notificaDao.doRetrieveAllByUserId(utente2.getId()).size();

      Partecipazione partecipazione = new Partecipazione(asta, utente2, 100);
      partecipazioneDao.doCreate(partecipazione);

      Assertions.assertTrue(astaService.partecipateAuction(utente, asta, 200));

      utente = utenteDao.doRetrieveById(2);

      Assertions.assertEquals(saldoPrecedente - 200, utente.getSaldo());
      Assertions.assertEquals(notifichePrima + 1,
          notificaDao.doRetrieveAllByUserId(utente2.getId()).size());

    }
```

ESEMPIO – SYSTEM

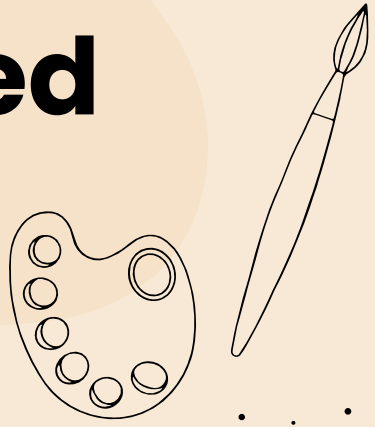


METRICHE

TIPO	#TEST_CASE
UNIT	461
INTEGRATION	333
SYSTEM	59

04

Lessons Learned



RETROSPECTIVE (LESSONS LEARNED)

Non aver paura di usare tecnologie nuove

Primo approccio al mondo del lavoro

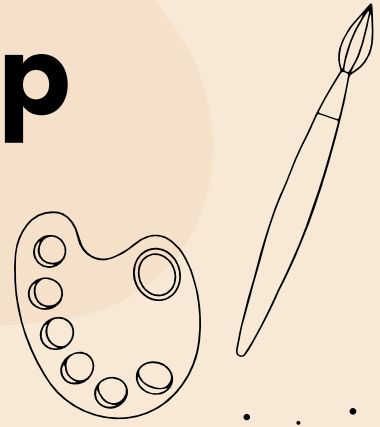
Importanza della comunicazione

Effort richiesto

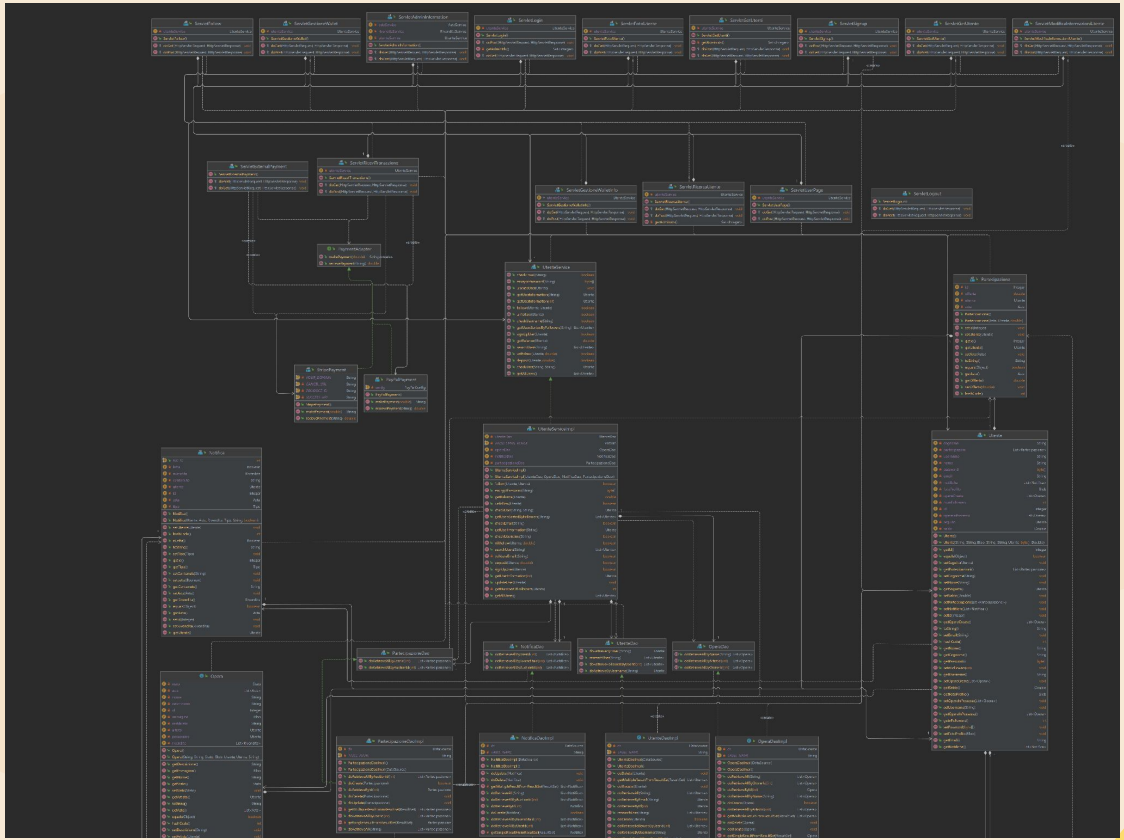
FINE

05

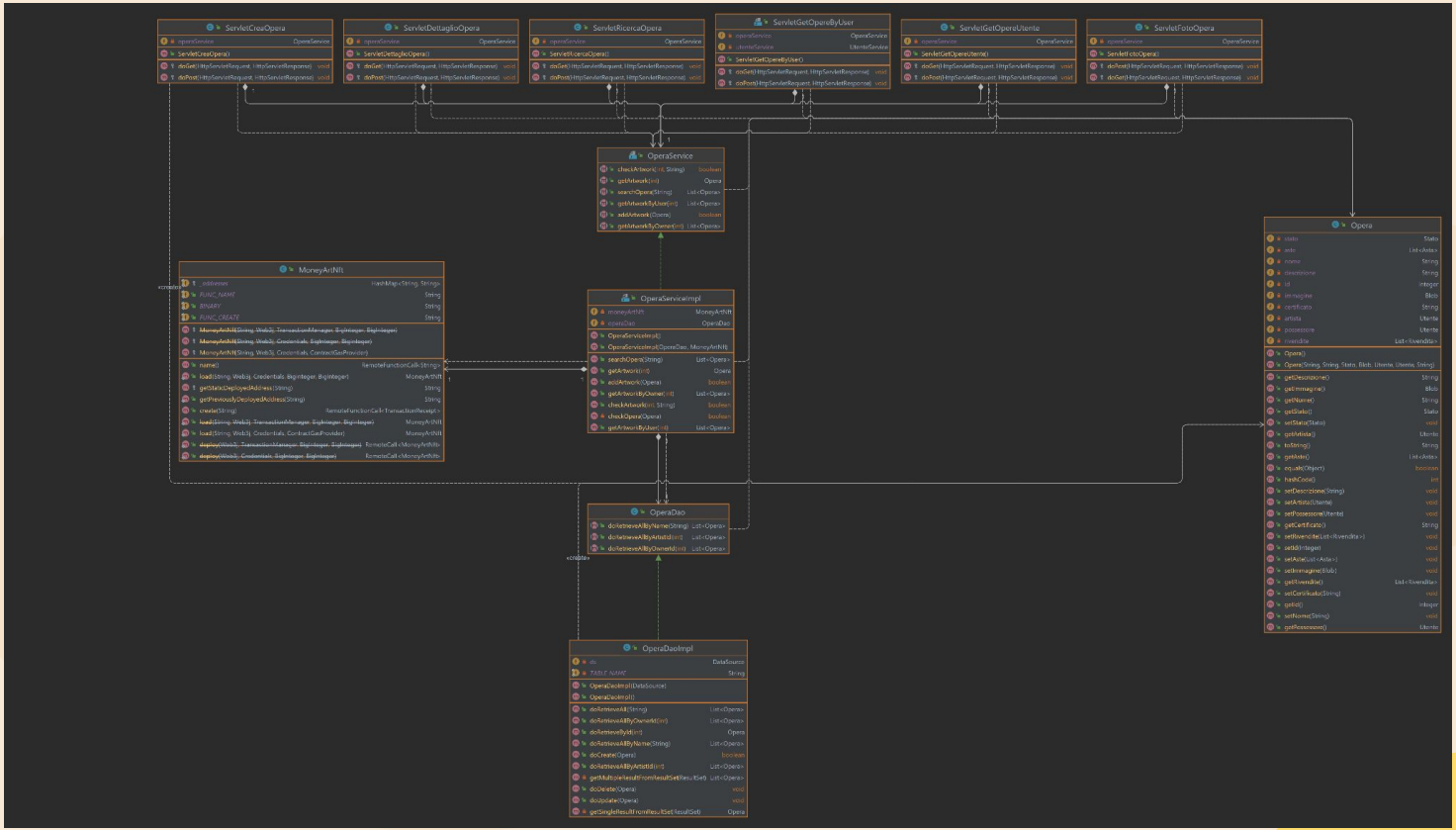
Slide di Backup



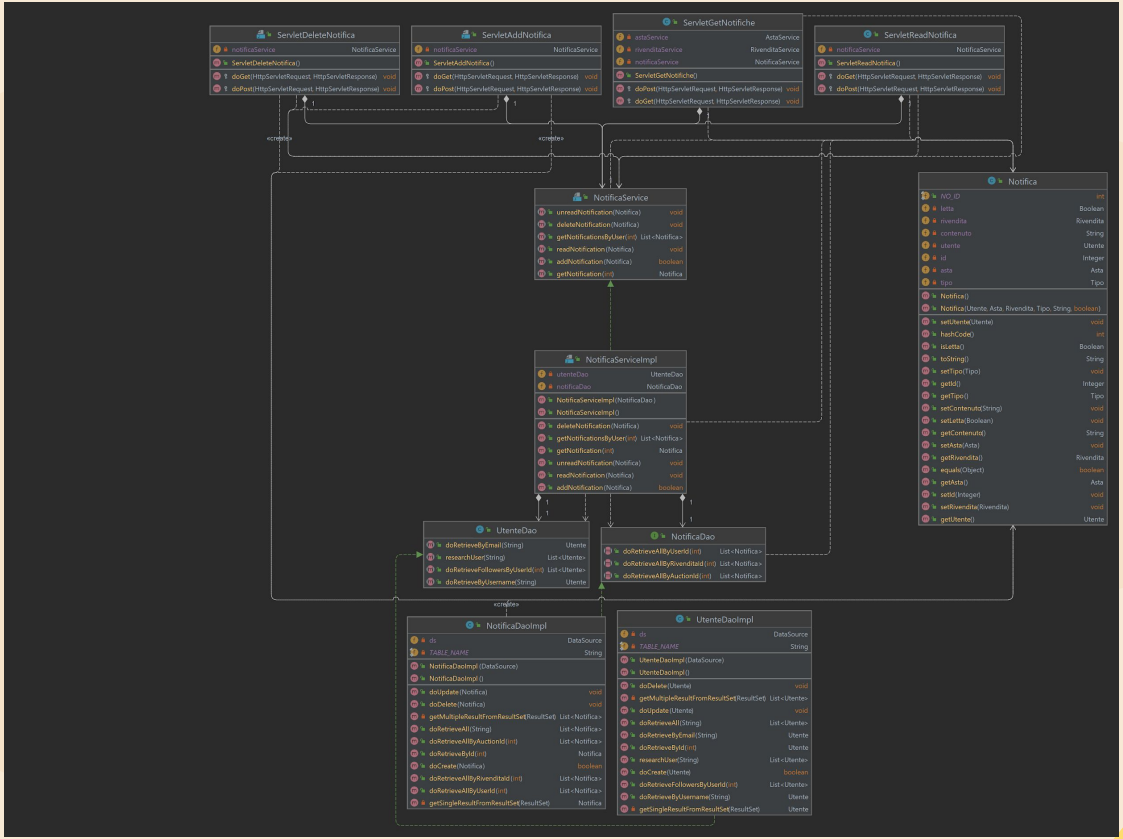
CLASS DIAGRAM-UTENTE



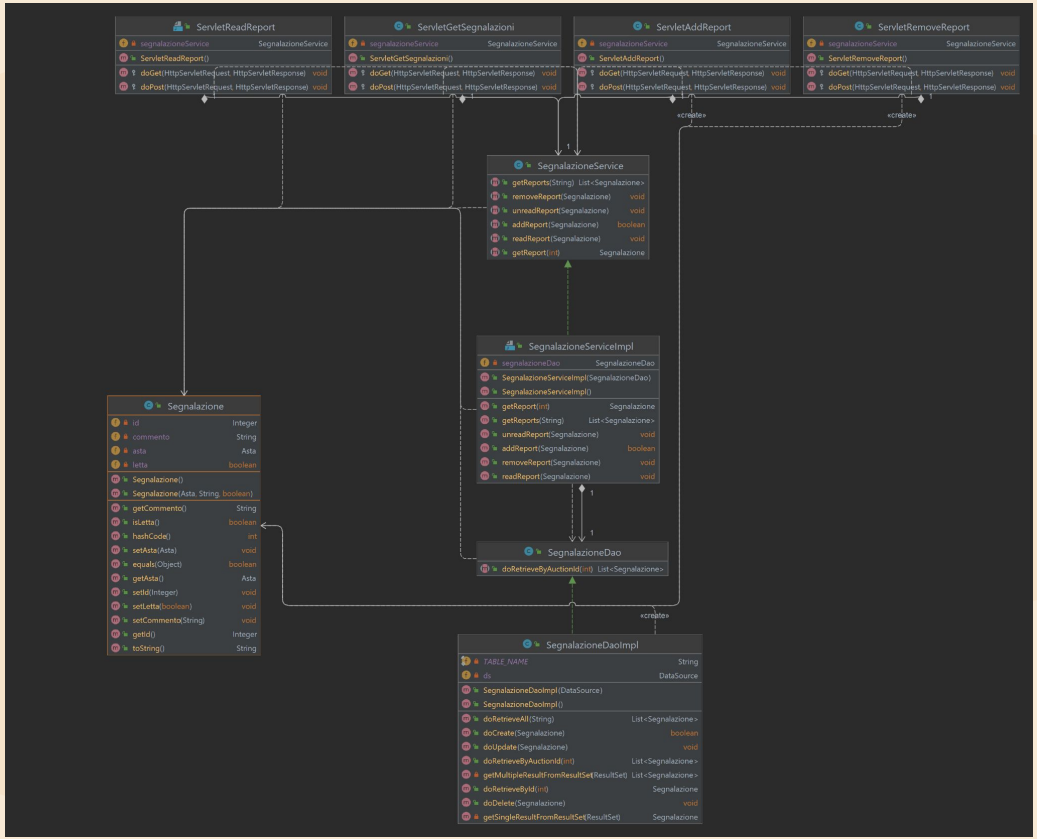
CLASS DIAGRAM-OPERA



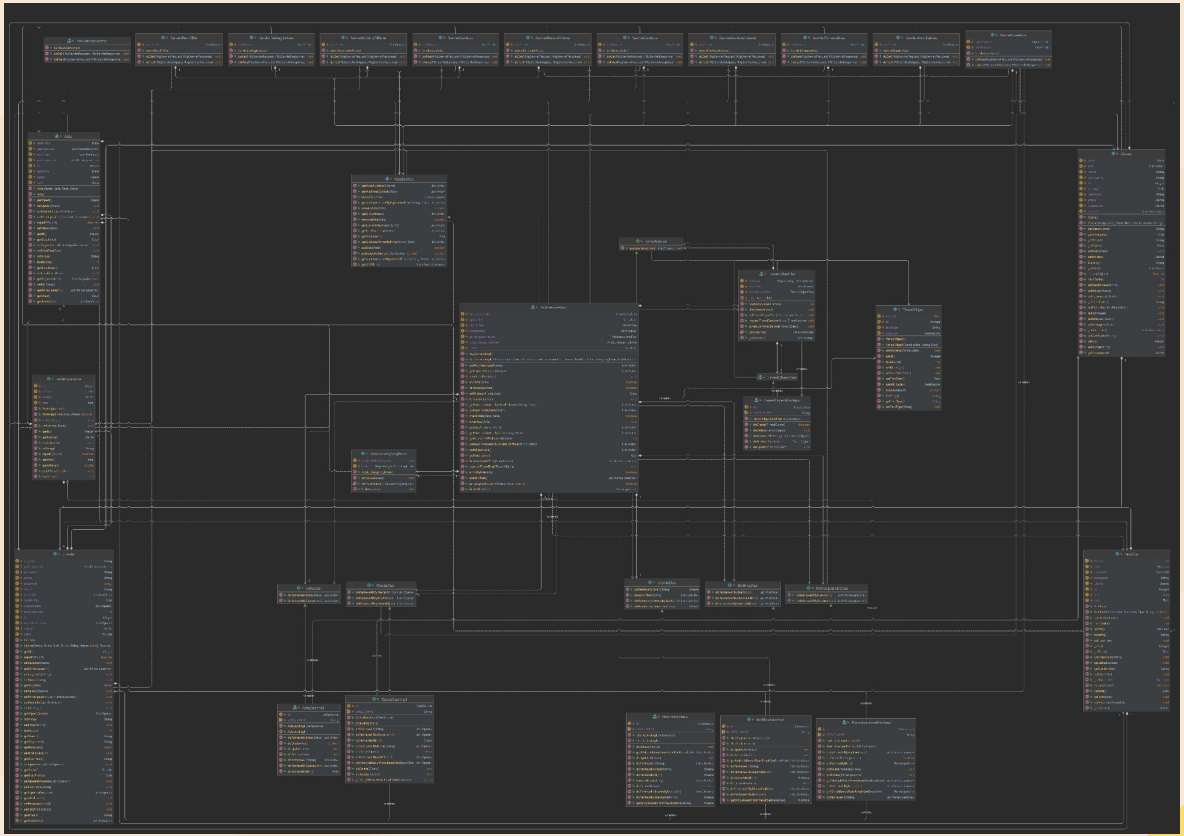
CLASS DIAGRAM-NOTIFICA



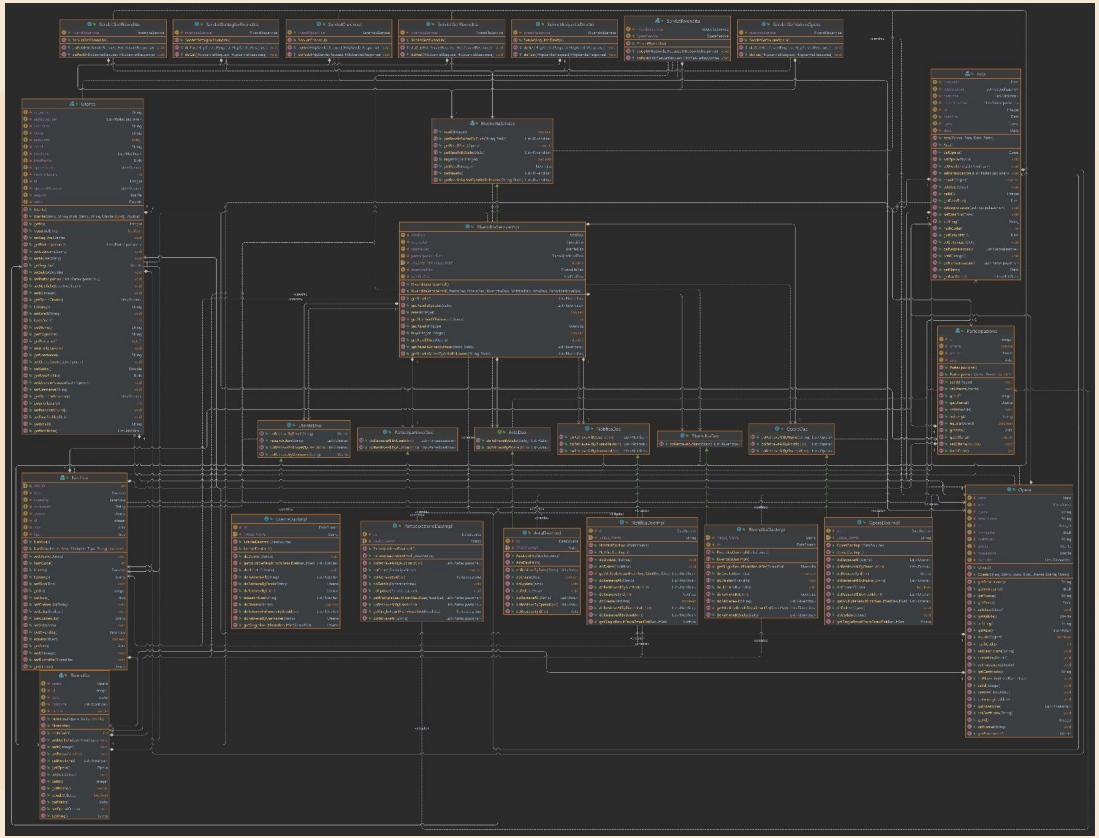
CLASS DIAGRAM-SEGNALAZIONE



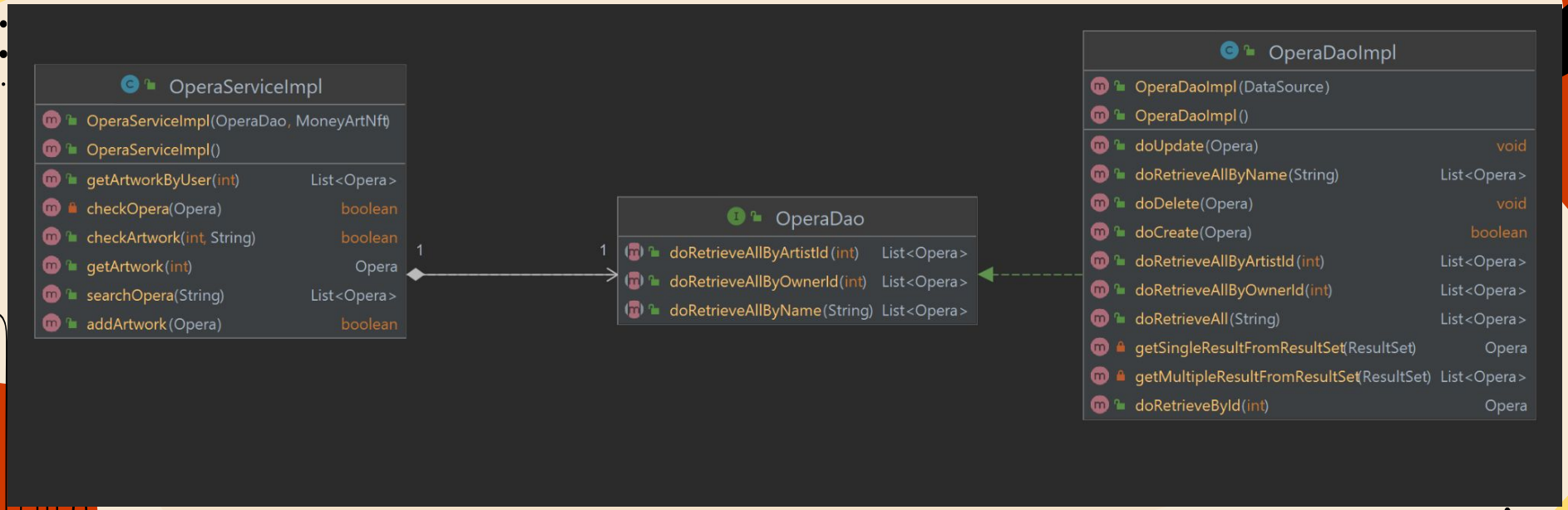
CLASS DIAGRAM-ASTA



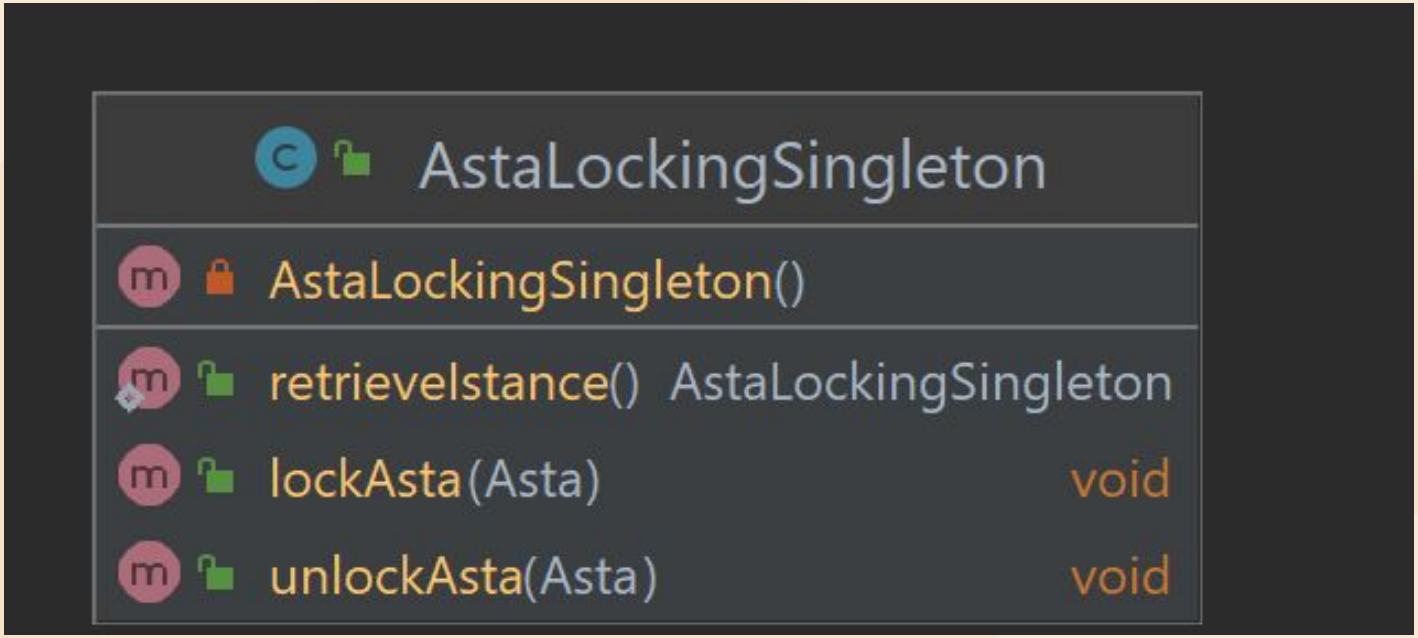
CLASS DIAGRAM-RIVENDITA



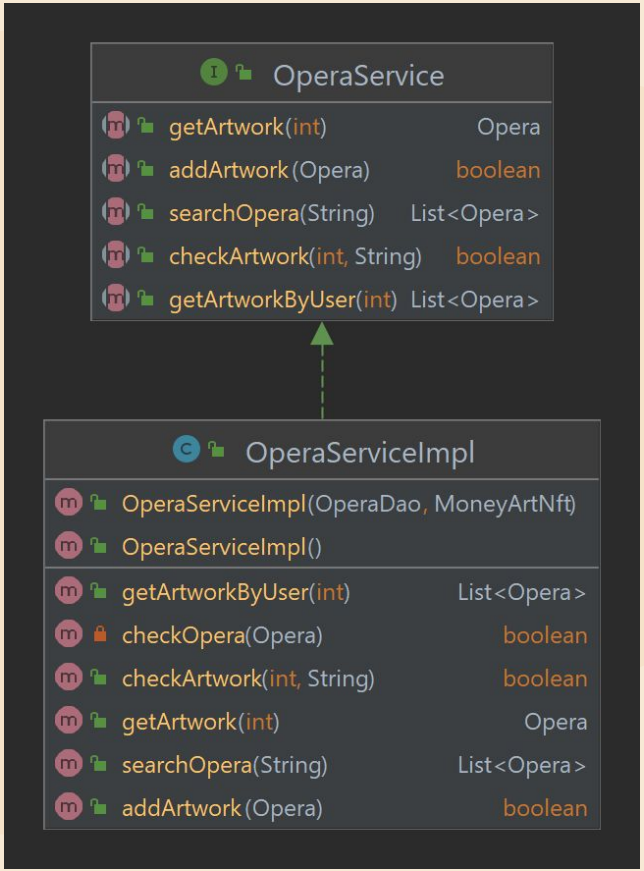
DESIGN PATTERN -> DAO



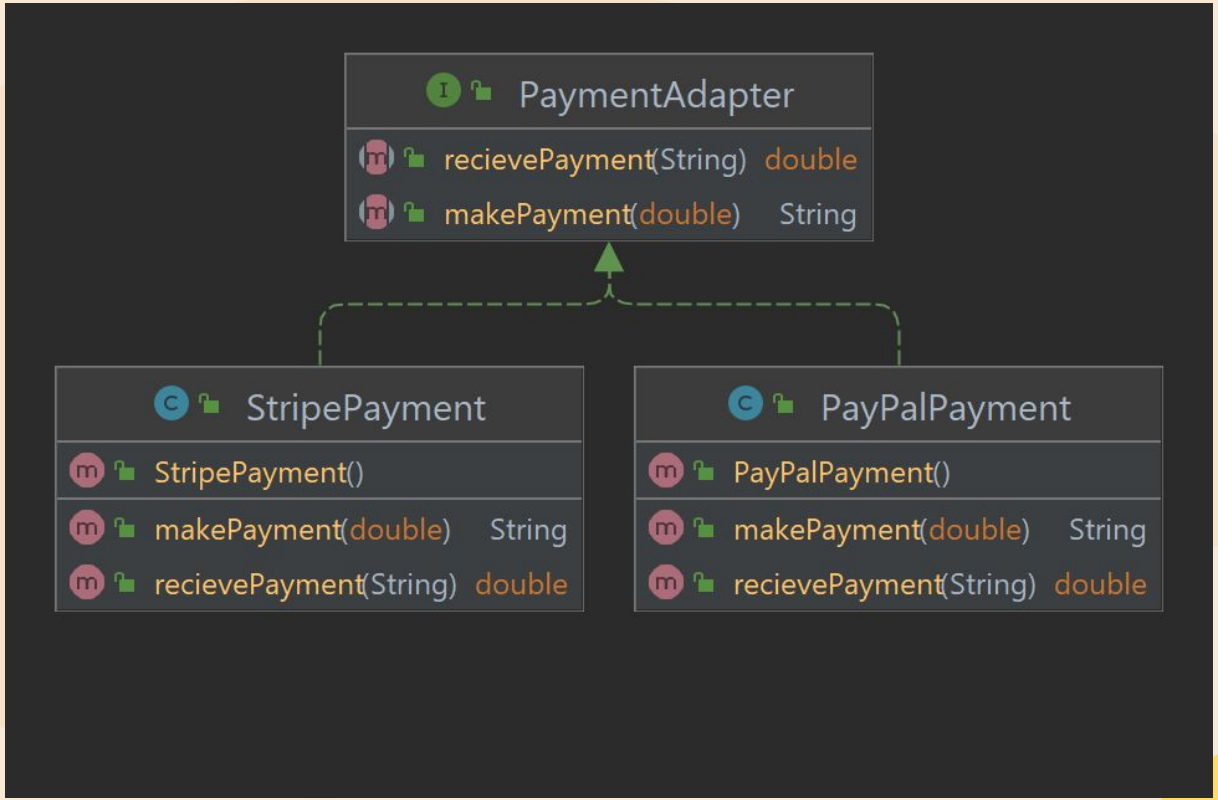
DESIGN PATTERN -> SINGLETON



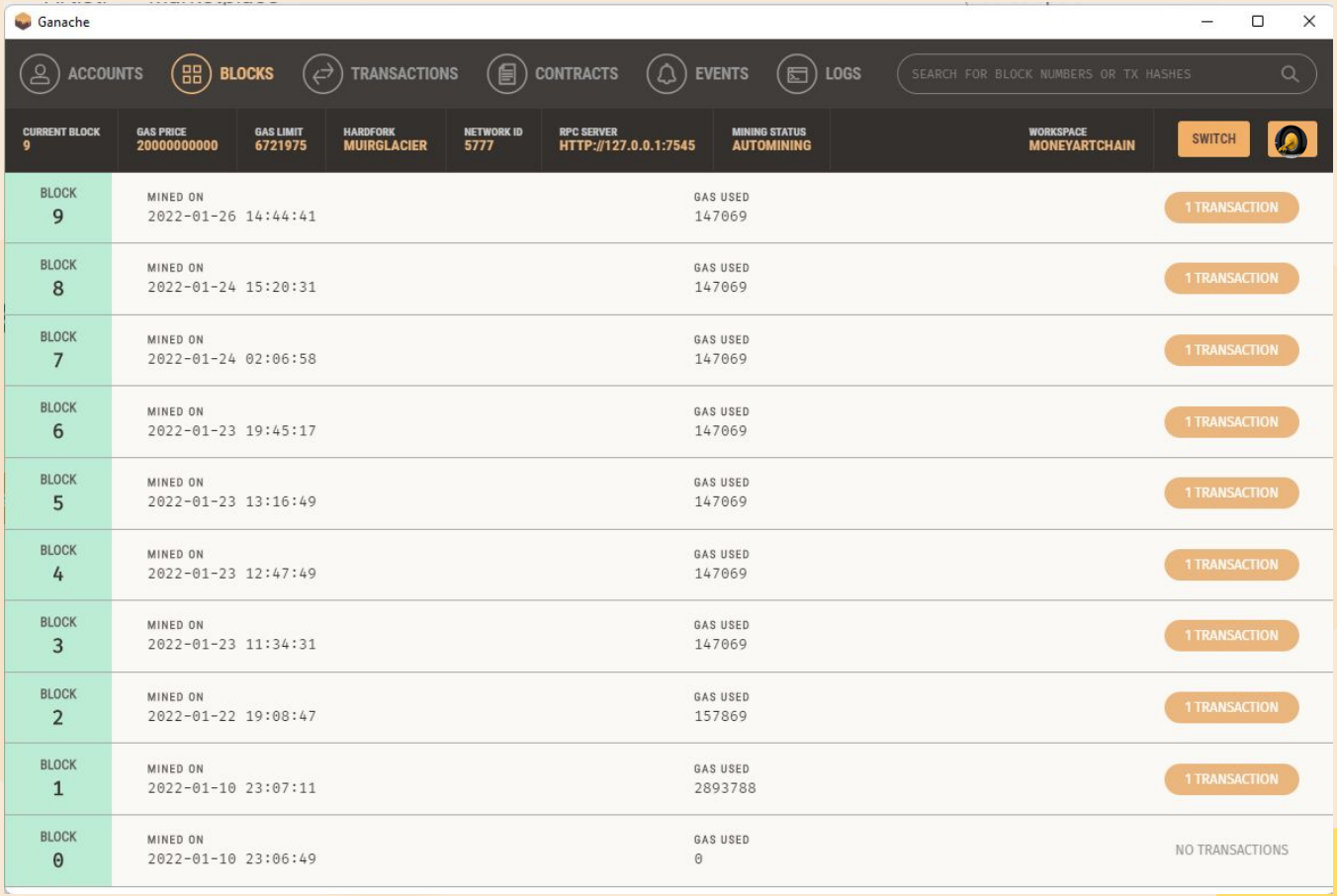
DESIGN PATTERN -> FACADE



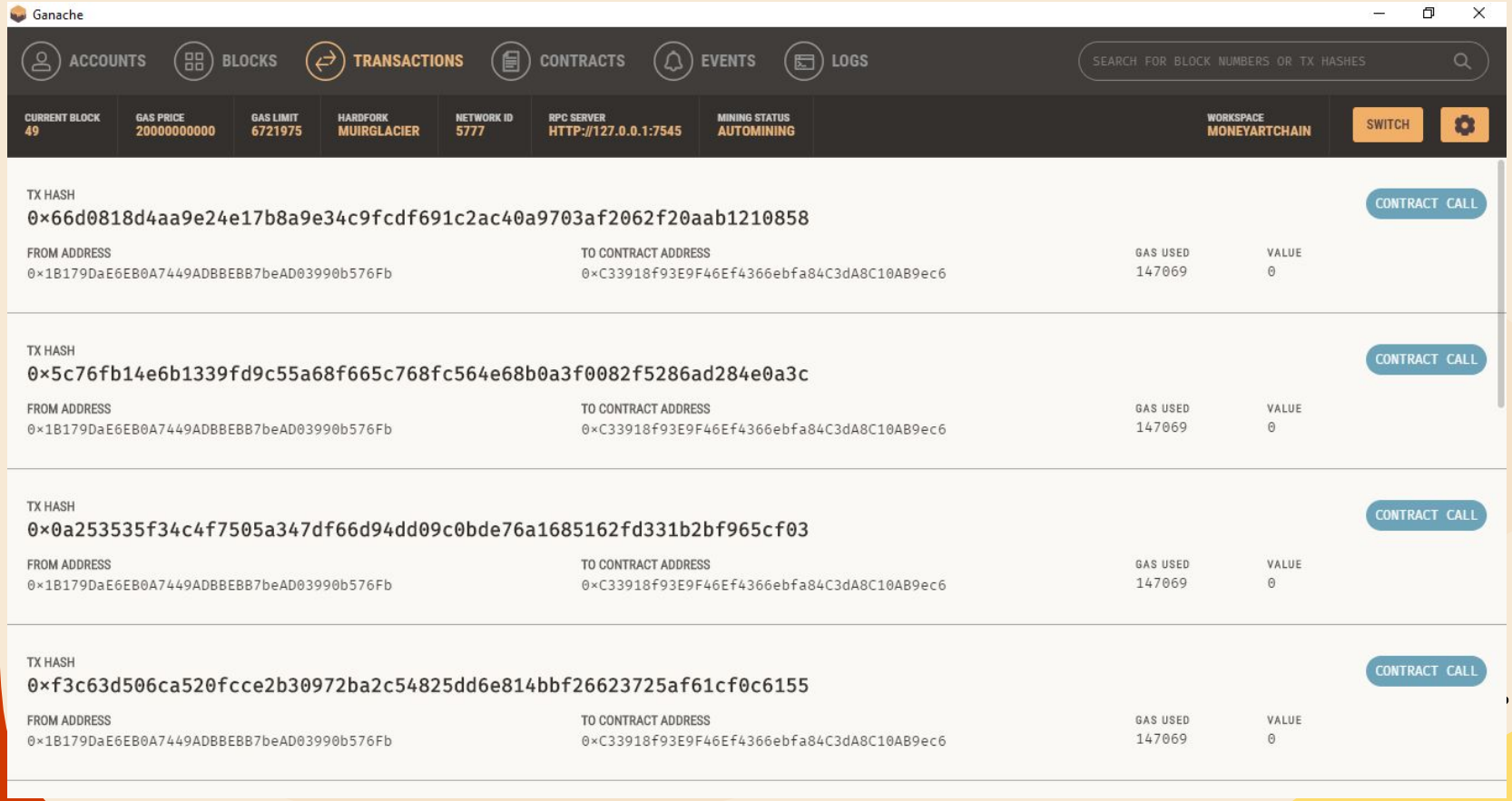
DESIGN PATTERN -> ADAPTER



GANACHE (1)


Ganache
ACCOUNTS
BLOCKS
TRANSACTIONS
CONTRACTS
EVENTS
LOGS
SEARCH FOR BLOCK NUMBERS OR TX HASHES
CURRENT BLOCK 9
GAS PRICE 20000000000
GAS LIMIT 6721975
HARDFORK MUIRGLACIER
NETWORK ID 5777
RPC SERVER HTTP://127.0.0.1:7545
MINING STATUS AUTOMINING
WORKSPACE MONEYARTCHAIN
SWITCH
BLOCK 9 MINED ON 2022-01-26 14:44:41 GAS USED 147069 1 TRANSACTION
BLOCK 8 MINED ON 2022-01-24 15:20:31 GAS USED 147069 1 TRANSACTION
BLOCK 7 MINED ON 2022-01-24 02:06:58 GAS USED 147069 1 TRANSACTION
BLOCK 6 MINED ON 2022-01-23 19:45:17 GAS USED 147069 1 TRANSACTION
BLOCK 5 MINED ON 2022-01-23 13:16:49 GAS USED 147069 1 TRANSACTION
BLOCK 4 MINED ON 2022-01-23 12:47:49 GAS USED 147069 1 TRANSACTION
BLOCK 3 MINED ON 2022-01-23 11:34:31 GAS USED 147069 1 TRANSACTION
BLOCK 2 MINED ON 2022-01-22 19:08:47 GAS USED 157869 1 TRANSACTION
BLOCK 1 MINED ON 2022-01-10 23:07:11 GAS USED 2893788 1 TRANSACTION
BLOCK 0 MINED ON 2022-01-10 23:06:49 GAS USED 0 NO TRANSACTIONS

GANACHE (2)


Ganache
ACCOUNTS
BLOCKS
TRANSACTIONS
CONTRACTS
EVENTS
LOGS
SEARCH FOR BLOCK NUMBERS OR TX HASHES
CURRENT BLOCK 49
GAS PRICE 20000000000
GAS LIMIT 6721975
HARDFORK MUIRGLACIER
NETWORK ID 5777
RPC SERVER HTTP://127.0.0.1:7545
MINING STATUS AUTOMINING
WORKSPACE MONEYARTCHAIN
SWITCH
TX HASH
0×66d0818d4aa9e24e17b8a9e34c9fcdf691c2ac40a9703af2062f20aab1210858
CONTRACT CALL
FROM ADDRESS
0×1B179DaE6EB0A7449ADBBEBB7beAD03990b576Fb
TO CONTRACT ADDRESS
0×C33918f93E9F46Ef4366ebfa84C3dA8C10AB9ec6
GAS USED
147069
VALUE
0
TX HASH
0×5c76fb14e6b1339fd9c55a68f665c768fc564e68b0a3f0082f5286ad284e0a3c
CONTRACT CALL
FROM ADDRESS
0×1B179DaE6EB0A7449ADBBEBB7beAD03990b576Fb
TO CONTRACT ADDRESS
0×C33918f93E9F46Ef4366ebfa84C3dA8C10AB9ec6
GAS USED
147069
VALUE
0
TX HASH
0×0a253535f34c4f7505a347df66d94dd09c0bde76a1685162fd331b2bf965cf03
CONTRACT CALL
FROM ADDRESS
0×1B179DaE6EB0A7449ADBBEBB7beAD03990b576Fb
TO CONTRACT ADDRESS
0×C33918f93E9F46Ef4366ebfa84C3dA8C10AB9ec6
GAS USED
147069
VALUE
0
TX HASH
0×f3c63d506ca520fcce2b30972ba2c54825dd6e814bbf26623725af61cf0c6155
CONTRACT CALL
FROM ADDRESS
0×1B179DaE6EB0A7449ADBBEBB7beAD03990b576Fb
TO CONTRACT ADDRESS
0×C33918f93E9F46Ef4366ebfa84C3dA8C10AB9ec6
GAS USED
147069
VALUE
0

SMART CONTRACT

```
pragma solidity 0.8.11;

import "./nf-token-metadata.sol";
import "./ownable.sol";

contract MoneyArtNft is NFTokenMetadata, Ownable {
  uint256 public tokenCounter;
  string internal tokenURIValue;
  constructor() {
    nftName = "MoneyArt";
    nftSymbol = "MA";
    tokenCounter = 0;
  }
  function create(string calldata _id) external onlyOwner {
    uint256 tokenId = tokenCounter;
    super._mint(msg.sender,tokenId);
    tokenCounter = tokenCounter + 1;
  }

  function setBaseURI(string calldata newToken) external onlyOwner{
    tokenURIValue = newToken;
  }
```

OCL

Nome metodo	+partecipateAuction(Utente utente,Asta asta,double offerta): boolean
Descrizione	questo metodo permette ad un utente di partecipare ad un asta con un offerta
Pre-condizione	**context:** AstaService::partecipateAuctionUtente utente,Asta asta,double offerta): boolean **pre:** asta.getStato() = IN_CORSO and utente.getSaldo() >= offerta and (bestOffer(asta) = null or offerta >= bestOffer(asta).getOfferta())
Post-condizione	**context:** AstaService::partecipateAuctionUtente utente,Asta asta,double offerta): boolean **post:** bestOffer(asta).getOfferta() = offerta and bestOffer(asta).getUtente() = utente and (@pre.bestOffer(asta).getUtente() = null

Video demo