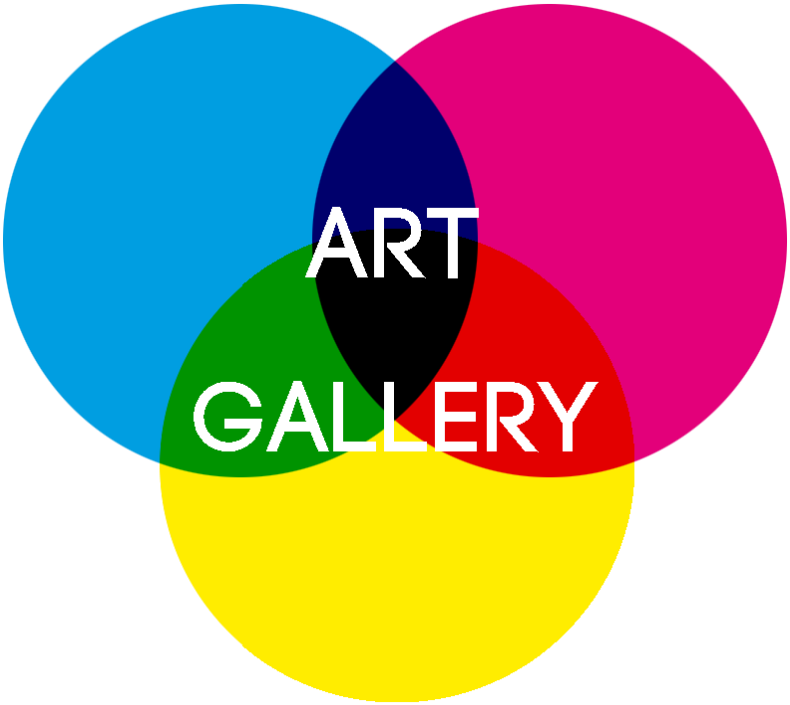
Università degli Studi Di Salerno
Corso di Ingegneria del Software

ArtGallery
System Design Document
Versione 2.0



DATA 13/12/2020

Coordinatori del progetto

Nome
De Lucia Andrea
Pecorelli Fabiano

Partecipanti al progetto

Nome	**Matricola**
Emanuele Fittipaldi	05121-05816
Fedele Mauro	05121-04496
Giuseppe Martinelli	05121-05972

Scritto da	Emanuele Fittipaldi, Giuseppe Martinelli, Fedele Mauro

REVISION HISTORY

Data	Versione	Descrizione	Autore
22/11/2020	1.0	1. Introduzione	Fedele Mauro
23/11/2020	1.0	2. Architettura del software corrente	Fedele Mauro
23/11/2020	1.0	3.1 Overview	Fedele Mauro
24/11/2020	1.0	3.2 Decomposizione in sottosistemi	Emanuele Fittipaldi
24/11/2020	1.0	UML-WAE-x	Emanuele Fittipaldi
25/11/2020	1.0	Glossario dei servizi dei sottosistemi	Emanuele Fittipaldi
26/11/2020	1.0	3.3 Mapping hw/sw	Fedele Mauro
26/11/2020	1.0	3.7 Boundary Condition	Fedele Mauro
26/11/2020	1.0	3.6 Controllo del software globale	Fedele Mauro
27/11/2020	1.0	Object,Concurrency,Memory,Connectivity Concerns	Emanuele Fittipaldi
28/11/2020	1.0	3.4 Gestione dei dati persistenti	Giuseppe Martinelli
28/11/2020	1.0	3.4.1 Scelta del database	Giuseppe Martinelli
1/12/2020	1.0	3.4.2 Rappresentazione dei dati, 3.4.3, 3.4.4	Giuseppe Martinelli
1/12/2020	1.0	3.5 Sicurezza degli accessi	Giuseppe Martinelli
13/12/2020	2.0	Revisione finale dell’intero documento	Emanuele Fittipaldi

System Design Document

1. Introduzione

1.1 Scopo del sistema

Il progetto “ArtGallery” è finalizzato alla realizzazione di un servizio web relativo all’acquisto di opere d’arte, più nello specifico, di dipinti. Il sistema consentirà agli utenti un’esperienza di acquisto quanto più simile a quella in una galleria d’arte fisica ma a portata di click, così da superare eventuali barriere dovute alla distanza dell’acquirente dalla galleria fisica, e fornendo tutti i dettagli delle opere in vendita nei minimi dettagli. Il sistema fornirà un’interfaccia semplice e guidata per effettuare le varie operazioni di acquisto, creazione di un account e per la gestione dei propri ordini.

1.1 Obiettivi di design

Gli obiettivi di design che ci siamo prefissati derivano sia dai requisiti non funzionali che da una serie di criteri che definiscono un insieme di qualità desiderabili, raggruppati nelle seguenti cinque categorie.

Criteri di affidabilità

Il sistema deve garantire la permanenza dei dati di estrema importanza, quali i dati degli utenti e i dettagli sugli acquisti effettuati da questi ultimi, poiché sono alla base delle funzionalità principali del sistema, tramite l’utilizzo di un database.

La fault tolerance sarà gestita attraverso interfacce grafiche, visualizzabili dall’utente, che gli permettono di comprendere le basi del problema verificatosi e di interagire correttamente con il sistema. Questo vuol dire che sono stati predisposti controlli sui campi più sensibili, come per esempio quelli inerenti alla registrazione e al log-in, quelli inerenti al metodo di pagamento, inserimento delle immagini nel sistema, modifiche ad utenti e ai quadri.

Per quanto riguarda la sicurezza, le informazioni sensibili quali il numero della carta di credito o la password dell’utente sono crittografate tramite crittografia SHA-1 e memorizzate in un database. La comunicazione tra l’utente e il sistema avverrà tramite protocollo HTTPS.

Criteri di performance

I tempi di risposta del sistema devono essere inferiori ad 1 secondo per quanto riguarde le operazioni principali. L’operazione che può richiedere un tempo di risposta più alto riguarda la fase del caricamento di una immagine e di pagamento, in cui il sistema si ritrova a dover mettersi in contatto con il sistema bancario e attendere la risposta sull’esito positivo o negativo della richiesta.

Il sistema deve supportare almeno 100 utenti concorrenti.

Per ogni utente la memorizzazione dei vari dati richiedono pochi kb di spazio, mentre la struttura principale del sito si aggira sui 20MB, bisogna includere lo spazio occupato dalle varie immagini che rappresentano le varie opere, che sarà al massimo di 1MB per ogni immagine caricata.

Ovviamente questi tempi sono assoluti, ma dipendono largamente dalla velocità della connessione con la quale il cliente fa accesso al sistema, il trade-off è fornire delle performance accettabili con una connessione di un utente medio di 100MB/s.

Criteri di costo

Non ci sono costi associati all'installazione del sistema in quanto il sistema proposto è di tipo web based. Non sono previsti neanche costi di amministrazione in quanto questa web application è pensata per essere auto amministrata. Sono previsti invece costi per chiamate di assistenza o preventivi per aggiunte di funzionalità al sistema e manutenzione in generale.

Criteri di mantenimento

Data la natura modulare con la quale il sistema verrà decomposto in sottosistemi autonomi, il sistema si appresta molto agevolmente all'aggiunta di nuovi moduli, permettendogli scalabilità e assecondamento del progresso. In quanto a portabilità il sistema è estremamente prortabile data la sua natura decentralizzata e non richiede installazioni in locale di alcun software.

Criteri di end user

All'accesso al sistema vogliamo che l'utente si trovi davanti una schermata principale che mostri tutte le opere in vendita sul sito tramite un'immagine;

Vogliamo che il sistema sia usufruibile anche da chi non è *Technology literate* e quindi tutti i meccanismi implementativi o che si discostano dall'obiettivo dell'acquisto di dipinti da parte del cliente, devono essere totalmente trasparenti ad esso.

L'utente in caso di necessità di aiuto o informazioni troverà una scheda per i Contatti nella barra di navigazione dove potrà mettersi in contatto con gli amministratori del sito.

1.2 Definizioni, acronimi e abbreviazioni

NFR*x*_*requirement* = Requisito non funzionale, dove *x* è il numero del requisito non funzionale di riferimento e *requirement* è il nome del requisito non funzionale.

Technology Literate = Individuo che dispone di un livello sufficiente di capacità di interazione con dispositivi tecnologi.

RAD = Requirement Analysis Document.

SDD = System Design Document.

Dipinto, Opera, Quadro = Usati in questo contesto per individuare la stessa cosa, ovvero un dipinto su tela, o un disegno.

Categoria = categoria di appartenenza del quadro: Ritratto, Paesaggio, Scena.

DB = Database.

Admin = amministratore del sistema, una figura che è in grado di manipolare direttamente i quadri e gli utenti inseriti nel sistema. Ha un insieme di funzioni che è diverso dall'Utente e ovviamente dall'utente non registrato.

Utente = rappresenta l'utente registrato del quale teniamo traccia nel nostro sistema e il quale può compiere un insieme di azioni diverse da un utente non registrato.

utente = L'utente rappresenta un utente generico che può essere un utente visitatore o un utente registrato.

1.4 Riferimenti

Per individuare gli obiettivi di design e i trade-off abbiamo fatto riferimento ai seguenti requisiti non funzionali descritti nel Requirement Analysis Document:

- NFR1.Usability
- NFR2.Reliability
- NFR4.Supportability
- NFR3.Performance

1.5 Overview

Il sistema verrà realizzato come un sito web dove sarà possibile acquistare opere d'arte, più nello specifico, dipinti. Il sistema consentirà la visualizzazione di tutte le opere in vendita dalla galleria, permettendo anche di visualizzare soltanto le opere appartenenti ad una determinata categoria di interesse. Sarà necessario registrarsi per effettuare un acquisto, mentre chiunque potrà accedere al sistema per visualizzare le opere e gli eventuali dettagli. Ogni utente che acquisterà le opere potrà visualizzare nella propria area utente i propri acquisti recenti e passati. Tramite una nav-bar posta nell'header del sito si potrà accedere alle varie aree del sistema.

2. Architettura del software corrente

Stiamo perseguendo un approccio di progettazione Greenfield, non basandoci quindi su un sistema in particolare già esistente. Abbiamo deciso soltanto di prendere qualche spunto in termini di design dell'interfaccia grafica e sul principio base di una architettura web based, analizzando piattaforme simili presenti in rete come: zoomonart.com, artegante.it, artmajeur.com e singulart.com, ma discostandoci totalmente dall'essere pedissequi al fine di dare spazio alla possibilità di apportare un contributo originale e soprattutto efficiente ed efficace al Sistema da realizzare.

3. Architettura del software proposto

3.1 Overview

Per il sistema ArtGallery è stata scelta l'architettura Model-View-Controller (MVC), che permette di dividere la logica di business da quella di presentazione.

Le responsabilità delle componenti dell'architettura sono:

- **Controller (Servlet):** nel Controller vi è tutta la business logic. Si occupa di processare le richieste dell'utente attraverso la View e reagisce eseguendo delle operazioni che possono riguardare il Model e che portano ad un cambiamento di stato della View.

- **Model:** nel Model sono incapsulate le risorse dell'applicazione. Contiene la logica di business e i metodi di accesso ai dati

- **View (JSP):** Nella View è racchiusa la parte di presentazione del sito. Si occupa di far visualizzare i dati all'utente e gestisce l'interazione fra quest'ultimo e l'infrastruttura sottostante. Attraverso JSP si ottengono i dati dei form. È responsabile della presentazione e si occupa della ricezione dell'input che viene restituito al Controller.

Vantaggi del modello MVC: i vantaggi del modello MVC sono molteplici e riguardano soprattutto la riusabilità del codice, semplificando operazioni di mantenimento e di aggiornamento. Con l'MVC si semplifica la gestione della complessità dividendo un'applicazione in Model, Controller e View. Questo modello risulta funzionare bene per applicazioni Web gestite da team di sviluppatori che necessitano un grado di controllo sul comportamento dell'applicazione.

Svantaggi del modello MVC: questo modello architetturale dipende molto dalla qualità di lavoro del team di progettazione. È un sistema molto articolato che richiede un'ottima dose di coordinazione nello sviluppo.

Comportamento di un'applicazione MVC:

1) L'utente richiede una risorsa tramite una URL (cliccando su un link, inserendo un indirizzo web nel browser, cliccando submit in un form, etc.).

2) Il browser costruisce una HTTP Request con informazioni prese dalla URL e l'invia, tramite internet, al server HTTP interessato (Application server) che la gira all'applicazione MVC ed in particolare alla sua componente Controller. Il Controller riceve dall'Application server la HTTP Request tradotta dall'Application Server in oggetto Java.

3) Il Controller legge l'oggetto di richiesta HTTP, ed utilizza i dati estratti per invocare il codice costituente il Model. In particolare, predispone oggetti java che dovranno contenere le risposte del Model (javabean) ed invia le sue richieste al Model.

4) Il Model, utilizzando i parametri ricevuti, interroga la base di dati a cui è collegato e restituisce i risultati nei javabean. A seconda di tali risultati il Controller invia il javabean ad una particolare componente della View.

5) La View legge i dati dal javabean, li formatta in una pagina web, inserisce tale pagina nell'oggetto di risposta e invia questa all'Application Server.

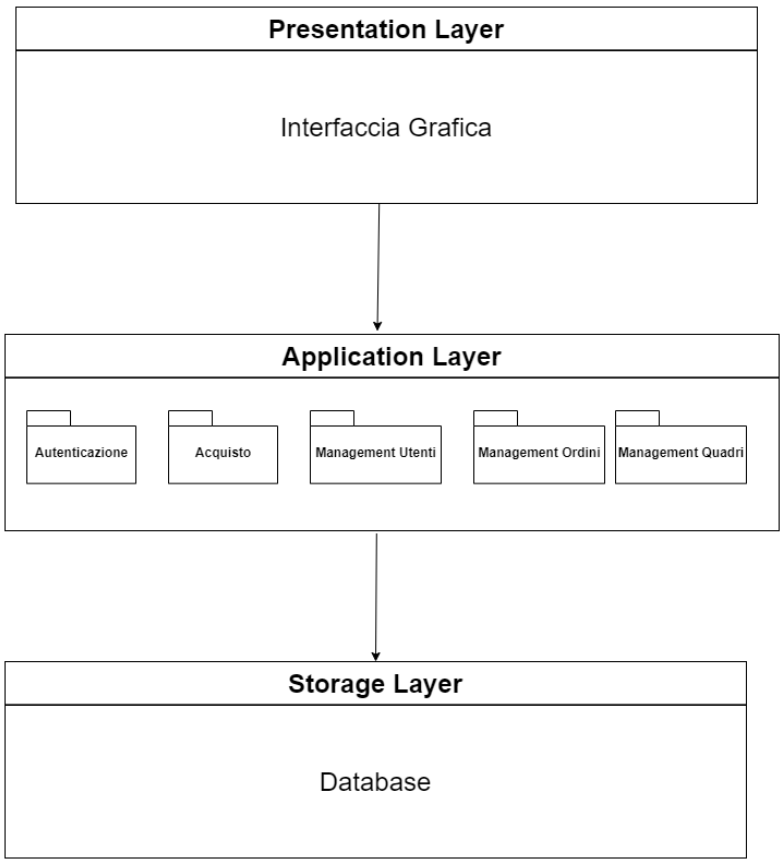
6) L'Application Server traduce l'oggetto di risposta ricevuto dalla View in un messaggio HTTP Response che viene quindi inviato al browser richiedente.

7) Il browser legge il messaggio HTTP Response, interpreta il codice HTML della pagina e mostra a video il risultato finale.

3.2 Decomposizione in sottosistemi

Il sistema è stato decomposto in tre layer:

1) **Presentation Layer**: è il layer a cui tutto ciò che l'utente vede e può interagire afferisce. Questo livello si interfaccia con l'application layer.
2) **Application Layer**: in questo livello troviamo il cuore del sistema. Il sistema è stato decomposto in piccoli sottosistemi autonimi e cooperativi, ognuno dei quali fornisce un insieme di servizi ben definito, al fine di aumentare la coesione e diminuire il più possibile l'accoppiamento. Questo approccio ci garantisce un sistema scalabile, che può reggere aggiornamenti anche dopo il rilascio, facilmente manutenibile e leggibile.
3) **Storage Layer**: In questo layer troviamo il nostro Database. Per approfondimenti vedi Sezione "3.4 Gestione dei dati persistenti"



UML WAE Diagrams

Nei seguenti diagrammi andiamo ad esplorare quello che è il comportamento dinamico del sistema durante l'esecuzione dei casi d'uso.

UML-WAE_Acquisto

Diagramma relativo al caso d'uso dell'acquisto.

Descrizione: Abbiamo una pagina client la quale è un aggregato di elementi HTML/JS. Nel diagramma qui sotto è stato inserito soltanto l'elemento <<form>> per brevità e perché in questo caso,rappresenta l'elemento con il quale stiamo interagendo per iniziare il caso d'uso.

1) Click su Checkout
2) Azione intercettata dal Controller Checkout Control che reindirizza l'Utente sulla pagina dove può inserire i dati relativi all'acquisto.
3) Una volta inseriti i dati, l'utente clicca su submit.
4) Azione intercettata dal Controller SalvaOrdine
5) SalvaOrdine prenderà dalla sessione i quadri che sono inseriti nel carrello
6) Questi quadri sono passati al manager SalvaOrdine che si interfaccerà con il layer di persistenza memorizzare l'ordine.

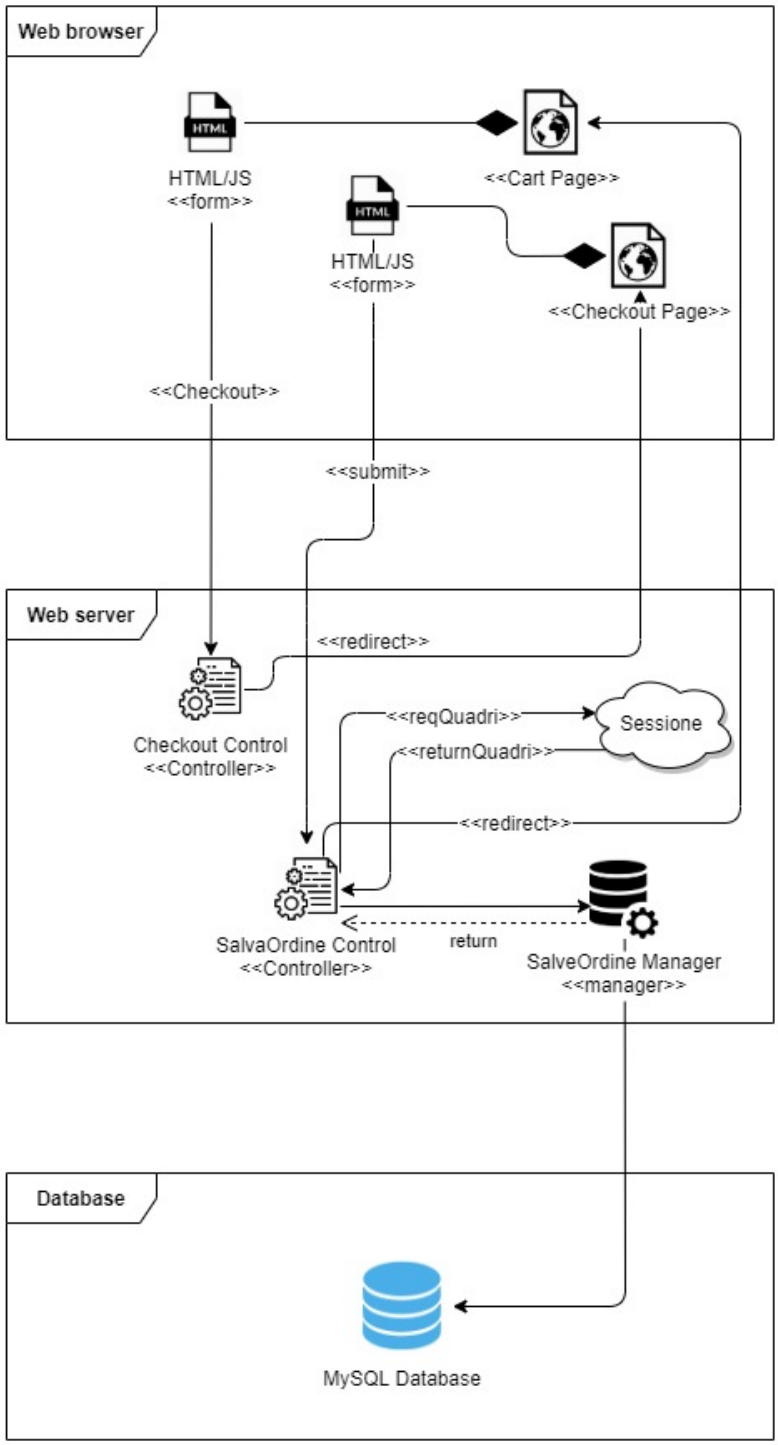
sequence diagram di riferimento (*vedi Requirement Analysis Document- Sezione 3.4.4*):

- **SD_ACQ**

Use case di riferimento (*vedi Requirement Analysis Document – Sezione 3.4.2*):

- **UC_ACQ**

UML-WAE_Acquisto



UML-WAE_AreaPersonale

Diagramma relativo al caso d'uso dell' accesso all'Area personale.

Descrizione: Abbiamo una pagina client la quale è un aggregato di elementi HTML/JS. Nel diagramma qui sotto è stato inserito soltanto l'elemento <<form>> per brevità e perché in questo caso,rappresenta l'elemento con il quale stiamo interagendo per iniziare il caso d'uso.

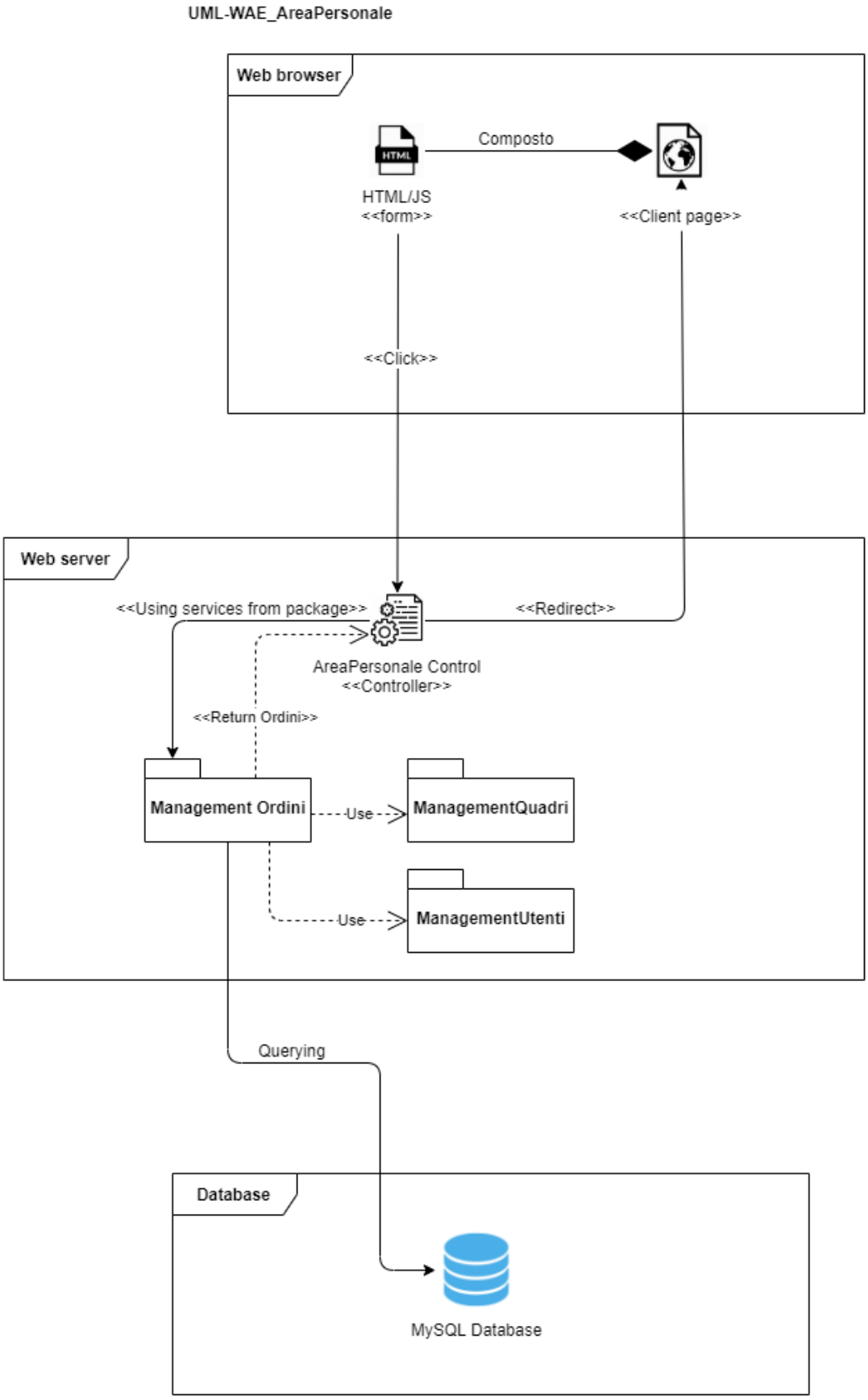
1) Click sul pulsante "Area personale"
2) Azione intercettata dal Controller AreaPersonale
3) Area personale utilizzerà dei servizi forniti dal sottosistema Management Ordini, il quale fornisce dei servizi per effettuare operazioni CRUD sugli ordini. In particolare, verranno usati i servizi che ci consentono di reperire gli ordini. Dato che in un ordine ci sono i Quadri, che vogliamo mostrare nell' Area personale avremo anche bisogno di reperire i quadri usando il servizio di reperimento dei quadri tramite id fornito da Management Quadri. Infine, ovviamente, nell'Area personale vogliamo visualizzare anche i dettagli sull'account dell'utente, e per questa ragione avremo bisogno del servizio offerto da Management Utenti per reperirle.
4) Una volta reperito tutto il necessario, viene restituito tutto al Control AreaPersonale, che si occuperà di reindirizzare l'utente sulla sua Area personale.

sequence diagram di riferimento (*vedi Requirement Analysis Document- Sezione 3.4.4*):

- **SD_AP**

Use case di riferimento (*vedi Requirement Analysis Document – Sezione 3.4.2*):

- **UC_AP**


UML-WAE_AreaPersonale
Web browser
HTML
Composto
HTML/JS
<<form>>
<<Client page>>
<<Click>>
Web server
<<Using services from package>>
<<Redirect>>
AreaPersonale Control
<<Controller>>
<<Return Ordini>>
Management Ordini
Use
ManagementQuadri
Use
ManagementUtenti
Querying
Database
MySQL Database

UML-WAE_LogOut
Diagramma relativo al caso d'uso del LogOut.

Descrizione: Abbiamo una pagina client la quale è un aggregato di elementi HTML/JS. Nel diagramma qui sotto è stato inserito soltanto l'elemento <<form>> per brevità e perché in questo caso,rappresenta l'elemento con il quale stiamo interagendo per iniziare il caso d'uso.

1) Click sul pulsante "LogOut"
2) Azione intercettata dal Controller LogOut Control
3) LogOut Control si occupera di invalidare la sessione. Questo farà terminare la sessione dell'utente.
4) Una volta invalidata la sessione, verrà notificato all'Utente di tale evento e verrà reindirizzato sulla pagina principale del sistema.

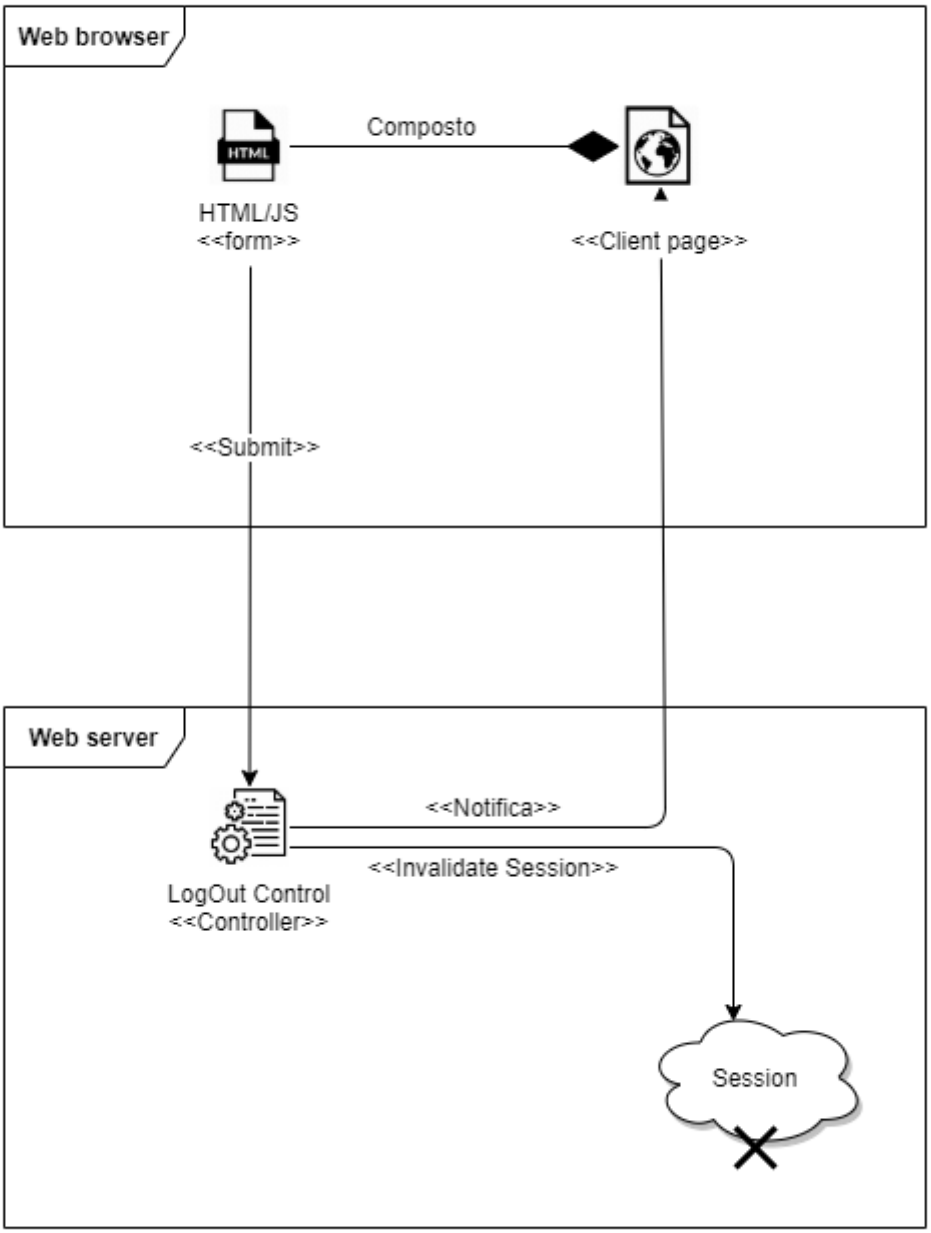
sequence diagram di riferimento (*vedi Requirement Analysis Document- Sezione 3.4.4*):

- **SD_LOGOUT**

Use case di riferimento (*vedi Requirement Analysis Document – Sezione 3.4.2*):

- **UC_LOUT**

UML-WAE_LogOut



UML-WAE_Login/Registration

Diagramma generico per il caso d'uso del Login e della registrazione.

Descrizione: Abbiamo una pagina client la quale è un aggregato di elementi HTML/JS. Nel diagramma qui sotto è stato inserito soltanto l'elemento <<form>> per brevità e perché in questo caso,rappresenta l'elemento con il quale stiamo interagendo per iniziare il caso d'uso.

Caso Login:

1) Click su submit
2) Le informazioni inserite dall'utente nei campi del form vengono passate al Controller Login Control.
3) Login Control si interfaccia con il manager Login Manager che si occuperà di reperire dal Database l'utente.
4) L'Utente reperito viene passato al Controller Login Control il quale si preouccuperà di inserire l'Utente nella sessione in modo da tenerne traccia durante le sue attività con il sistema e per permettere di usare servizi come l'aggiunta di un quadro in un carrello.
5) Viene notificato all'Utente dell'avvenuto Login e viene reindirizzato sulla pagina principale del sistema.

Caso Registrazione:

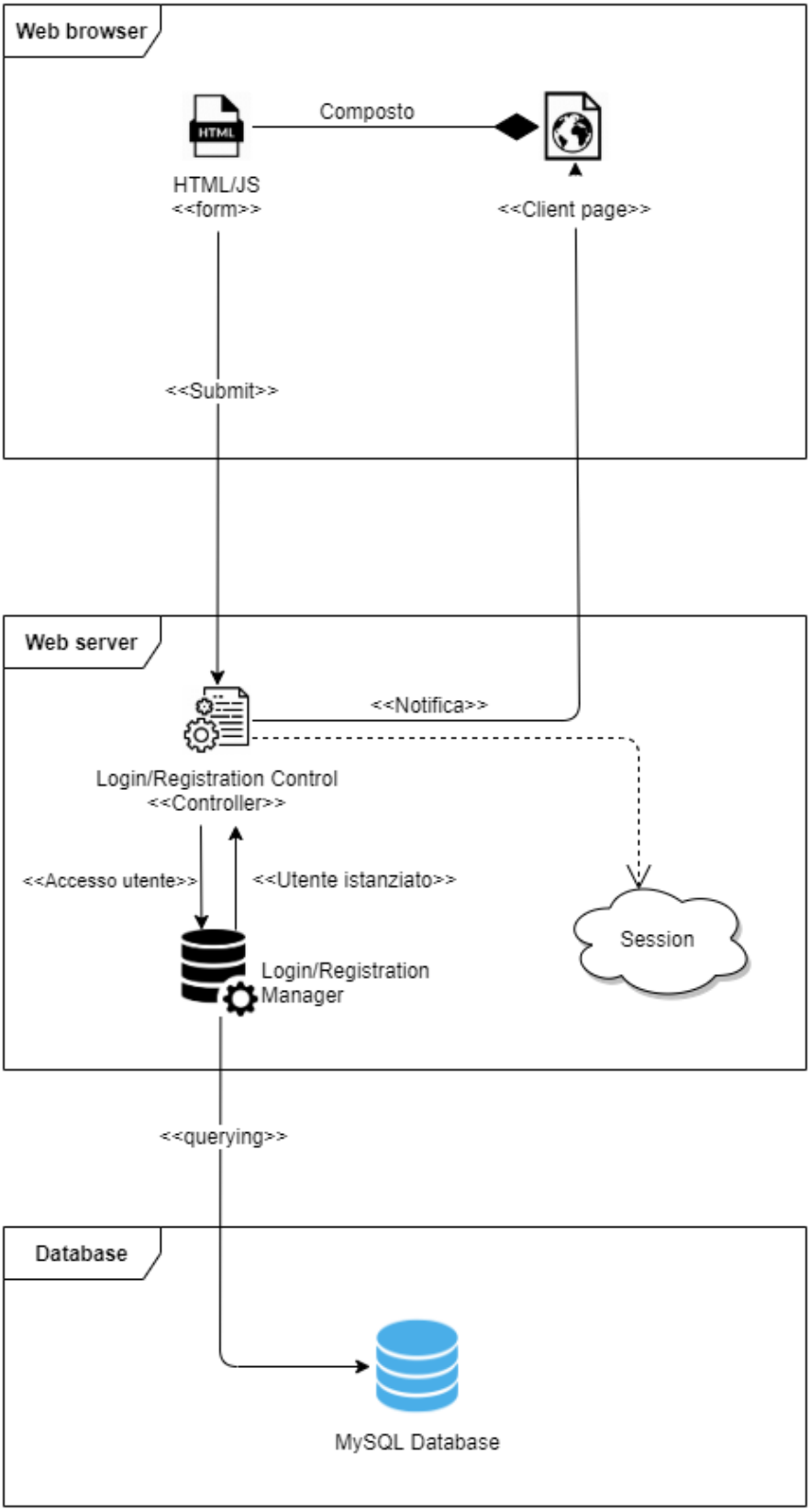
1) A compilazione del form per la registrazione avvenuta, tramite il click su submit parte una nuova azione.
2) Questa azione viene intercettata dal controller Registration Control.
3) Registration Control si interfaccia con il Manager Registration che si preouccuperà di rendere persistente il nuovo utente nel database.
4) L'avvenuta registrazione viene comunicata al Registration Control che propagherà la notifica a video, all'Utente.

sequence diagrams di riferimento (*vedi Requirement Analysis Document- Sezione 3.4.4*):

- **SD_LOGIN**
- **SD_REG**

Use case di riferimento (*vedi Requirement Analysis Document – Sezione 3.4.2*):

- **UC_LIN**
- **UC_REG**


Web browser
HTML
Composto
HTML/JS
<<form>>
<<Client page>>
<<Submit>>
Web server
<<Notifica>>
Login/Registration Control
<<Controller>>
<<Accesso utente>>
<<Utente istanziato>>
Session
Login/Registration
Manager
<<querying>>
Database
MySQL Database

UML-WAE_CRUD

Diagramma generico relativo alle operazioni CRUD su Quadro, Utente e Ordine.

Descrizione: Abbiamo una pagina client la quale è un aggregato di elementi HTML/JS. Nel diagramma qui sotto è stato inserito soltanto l'elemento <<form>> per brevità e perché in questo caso,rappresenta l'elemento con il quale stiamo interagendo per iniziare il caso d'uso.

1) Una azione Crud nel nostro sistema viene intrapresa tramite una azione lanciata al momento del click di un pulsante in un form.
2) Questa azione viene intercettata da un Control che si preoccuperà di interagire con un Manager per richiedere il reperimento, l'aggiunta, la cancellazione o l'update di una informazione
3) L'operazione sarà portata a termine dal manager che si preoccuperà di notificare al Control dell'esito e di passare ad esso eventuali informazioni reperite.
4) Il Control a questo punto propagherà l'informazione all'utente.

Nota: Una piccola nota doverosa nei confronti di Ordine.
Per Ordine non abbiamo predisposto la cancellazione come una operazione attuabile direttamente, bensì, abbiamo trattato la rimozione di un ordine come conseguenza della rimozione di un determinato utente, al fine di garantire il vincolo di integrità referenziale che vige tra utente e ordine all'interno del nostro sistema di persistenza.

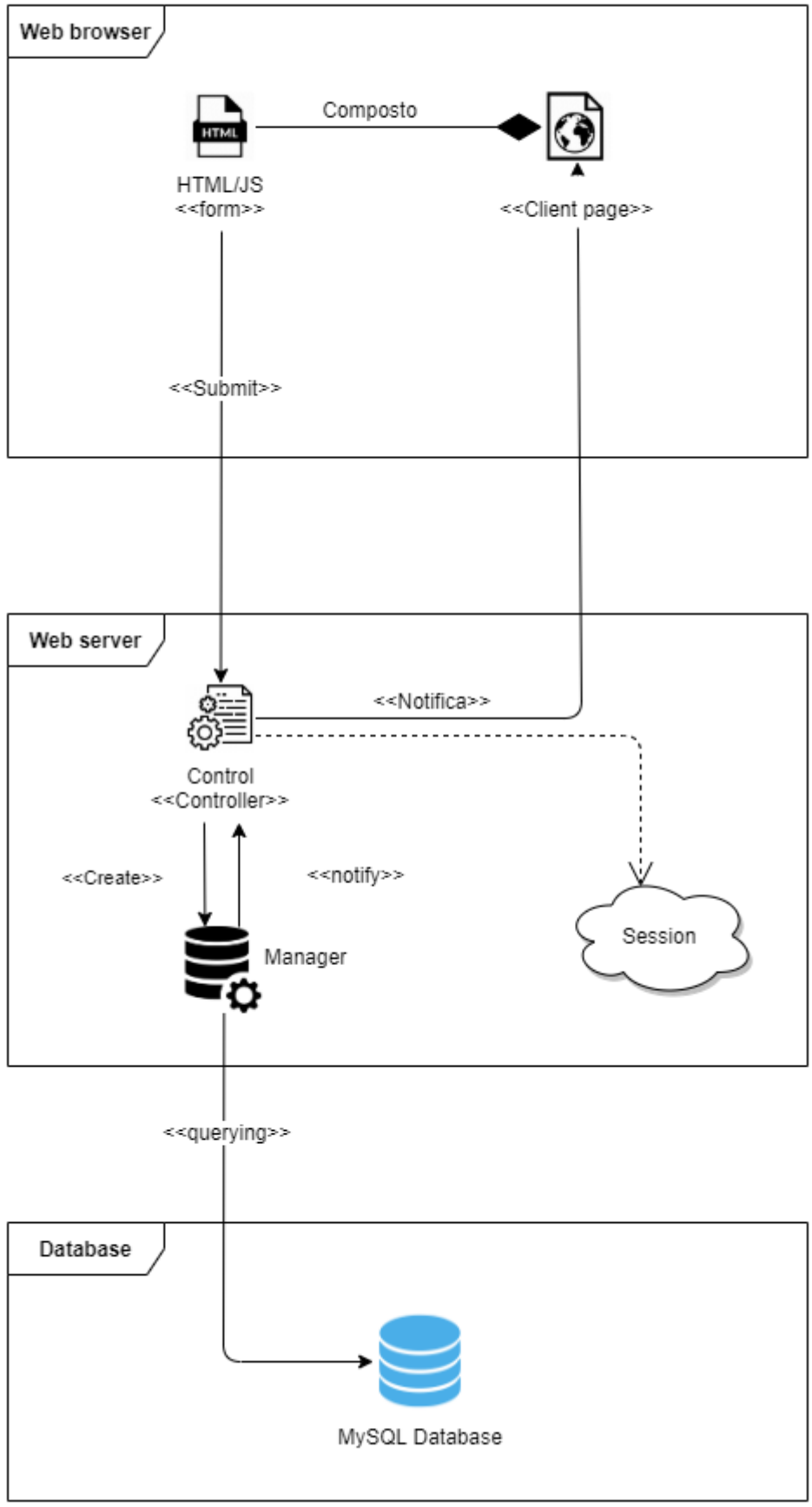
sequence diagrams di riferimento (*vedi Requirement Analysis Document- Sezione 3.4.4*):

- **SD_RIM_DIP**
- **SD_RIM_UTENTE**
- **SD_VQ**
- **SD_MOD_UTENTE**
- **SD_MOSTRA_UTENTI**
- **SD_MOD_DIP**

Use case di riferimento (*vedi Requirement Analysis Document – Sezione 3.4.2*):

- **UC_RIM_DIP**
- **UC_RIM_UTENTE**
- **UC_MOD_DIP**
- **UC_MOSTRA_OPERE**
- **UC_MOSTRA_UTENTI**
- **UC_VQ**

UML-WAE_CRUD


Web browser
HTML
Composto
HTML/JS
<<form>>
<<Client page>>
<<Submit>>
Web server
<<Notifica>>
Control
<<Controller>>
<<Create>>
<<notify>>
Manager
Session
<<querying>>
Database
MySQL Database

3.3 Mapping hardware/software

La divisione del Sistema in sottosistemi è realizzata a livello software, mentre a livello Hardware troviamo la suddivisione è concepita attraverso i tier di Interface, Business e Storage. Questi tier sono divisi nel seguente modo:

- Tier Interface sulla macchia del client
- Tier di Business e Storage sul server.

Objects Mapping Concerns

L'Object model viene mappato sul sistema attraverso l'impiego di un Database relazionale in esecuzione su un Database Server. Di conseguenza le tassonomie e le relazioni tra le varie componenti del sistema sono mappate come relazioni tra tabelle nel database tramite chiavi esterne e vincoli di integrità referenziale.

Concurrency Concerns

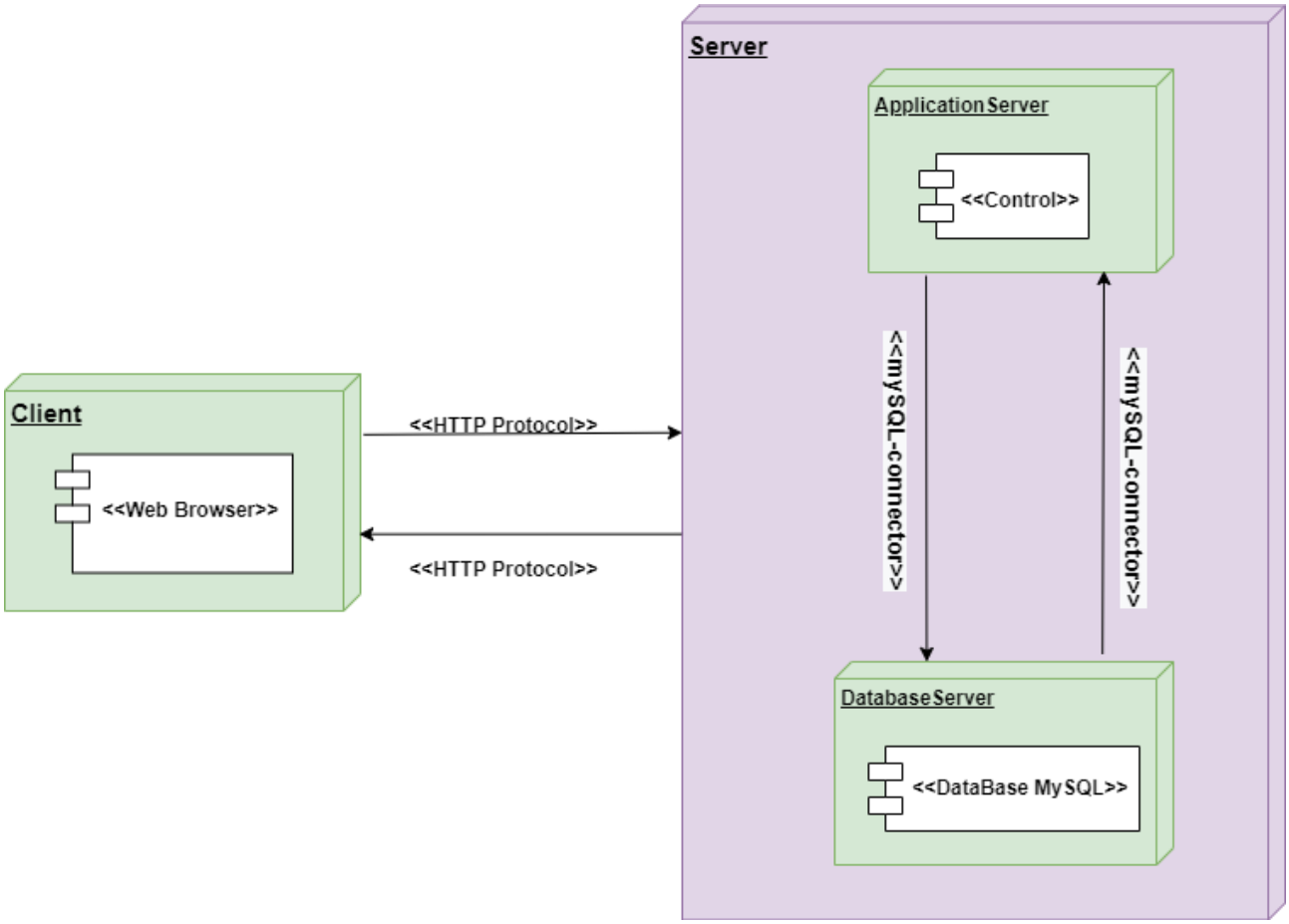
Dato l'impiego di un Applciation Server, la concorrenza è gestita automaticamente dallo stesso, mentre il carico del lavoro a regime viene ripartito automaticamente dal Sistema Operativo tra più thread e core di cui la nostra macchina dispone.

Memory Concerns

La memoria per gestire le operazioni è ampiamente sufficiente in quanto non sono previste attività da parte degli Utenti particolarmente onerose in termini di spazio. L'unica operazione che prevede uno spazio di archiviazione appena notabile è l'inserimento di una immagine a seguito dell'inserimento di un nuovo Quadro nel sistema, la quale operazione può richiedere fino a 2 MB.

Connectivity Concerns

Le comunicazioni tra Client e Server avvengono naturalmente tramite protocollo di livello applicativo HTTP, mentre all'interno del Sistema, tramite jdbc per quanti riguarda l'interazione tra i sottosistemi e l'unità di persistenza, e tramite riferimenti per le interazioni tra i vari sottosistemi.



3.4 Gestione dei dati persistenti

3.4.1 Scelta del Database

Abbiamo deciso di gestire la persistenza dei dati attraverso un Database relazionale. Il DBMS selezionato per aiutarci in questo compito è MySQL.

3.4.2 Rappresentazione dei dati

Le seguenti tabelle descrivono come i dati sono mappati all'interno del DB. Ogni tabella contiene al suo interno: la tipologia di dato da mappare, il nome del dato all'interno del DB e i vari ed eventuali vincoli. Inoltre, le tabelle hanno un tag che le distingue dalle altre.

Tabella Utente

Utente		
Dato	Tipologia	Vincoli
Username	**Char(15)**	**Primary key**
Password	Char(100)	Not null
Nome	VarChar(30)	Not null
Cognome	VarChar(30)	Not null
Nazionalità	VarChar(30)	Not null
E-mail	Char(30)	Not null

Tabella Ordine

Ordine		
Dato	Tipologia	Vincoli
IdOrdine	**Integer**	**Primary key, Auto Incremental**
TipoCarta	VarChar(30)	Not null
NumCarta	Char(30)	Not null
DataAcquisto	Date	Not null
FK_Utente	Char(15)	Foreign Key, Not null

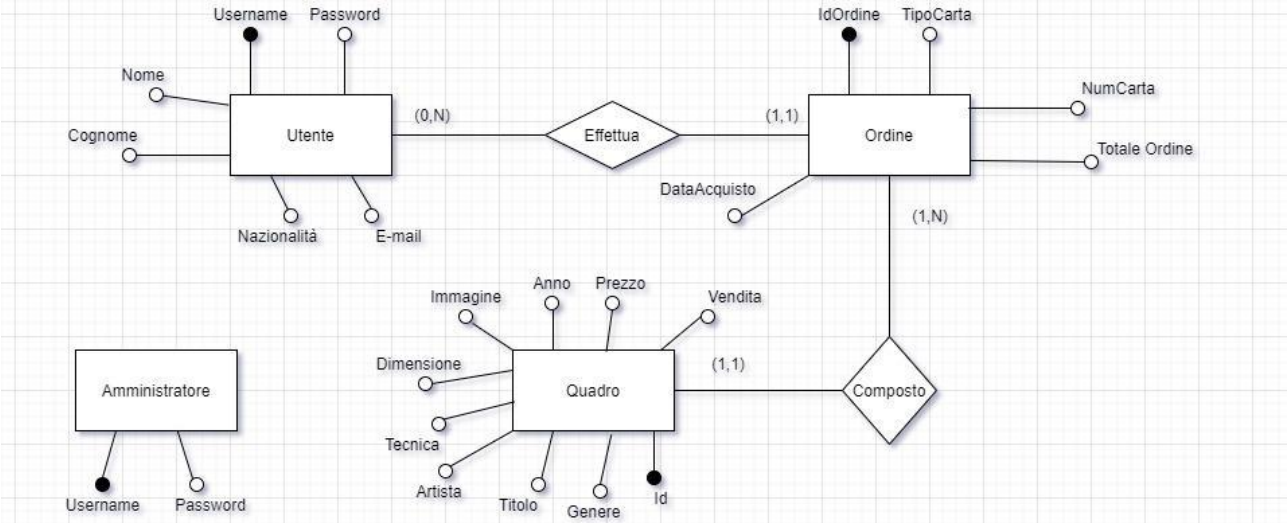
Tabella Quadro

Quadro		
Dato	Tipologia	Vincoli
Id	**Integer**	**Primary key, Auto Incremental**
Genere	VarChar(100)	Not null
Titolo	VarChar(30)	Not null
Artista	VarChar(30)	Not null
Tecnica	VarChar(30)	Not null
Dimensione	VarChar(30)	Not null
Immagine	Char(260)	Not null
Anno	Char(30)	Not null
Prezzo	Double	Not null
FK_IdOrdine	Integer	Foreign Key, Not null

Tabella Amministratore

Amministratore		
Dato	Tipologia	Vincoli
Username	**Char(15)**	**Primary key**
Password	Char(100)	Not null

3.4.3 Diagramma ER e traduzione modello relazionale



Traduzione:

Utente(<u>Username</u>,Password,Nome,Cognome,Nazionalità,E-mail)

Ordine(<u>IdOrdine</u>,TipoCarta,NumCarta,TotaleOrdine,DataAcquisto,**IdUtente**)

Quadro(<u>Id</u>,Genere,Titolo,Artista,Tecnica,Dimensione,Immagine,Anno,Prezzo,Vendita,**IdOrdine**)

Amministratore(<u>Username</u>,Password)

3.4.4 Sicurezza

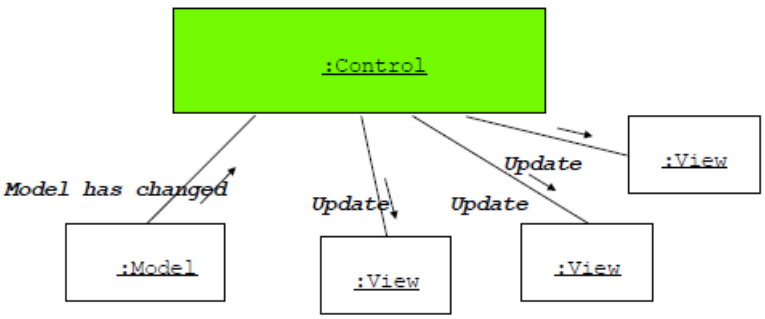
Informazioni particolarmente sensibili, come le password scelte dagli utenti, dagli amimnistratori e i numeri di carta di credito sono protette tramite crittografia SHA-1. Questo rende la comunicazione tra il Database e la logica di business del sistema sicura, in quanto qualsiasi tipo di intercettazione da parte di terzi, durante il il processo di scambio di dati sensibili, renderebbe i dati reperiti illegalmente illegibili e privi di utilità.

3.5 Controllo degli accessi e sicurezza

Attori	Gestione Utenti	Gestione Quadri	Gestione Acquisti
Amministratore	- Rimozione utente - Modifica utente	- Rimozione quadro - Modifica quadro - Inserimento quadro	
Utente registrato	- Log-In nel sistema - Visione pagina personale - Visione degli ordini effettuati in passato - Log-out dal sistema - Creare altri account	- Visionare i quadri - Ricerca filtrata dei quadri	- Aggiungere un quadro al carrello - Eliminare il quadro dal carrello - Comprare i quadri dal carrello
Utente non registrato	- Registrarsi nel sistema	- Visionare i quadri - Ricerca filtrata dei quadri	- Aggiungere un quadro al carrello - Eliminare il quadro dal carrello

3.6 Controllo software globale

Si utilizza un **controllo del flusso esplicito centralizzato** di tipo Event Driven. Il controllo risiede in un dispatcher che chiama le funzioni mediante callback. È molto flessibile, buono per progettare interfacce grafiche ed è facile da estendere.



In un sistema Event Driven, l'acquisizione, la comunicazione, l'elaborazione e la persistenza degli eventi costituiscono i pilastri portanti della soluzione. Il producer genera un evento, come ad

esempio l'acquisto di un quadro e lo rappresenta come messaggio. Questo messaggio è trasmesso dal producer al/ai consumer tramite un canale di eventi. I consumer sono "in ascolto" su questo canale e quando arriva un messaggio lo elaborano.

Un esempio di evento lo abbiamo quando un utente inserisce un oggetto all'interno del proprio carrello in un e-commerce. L'evento avrà un suo stato (id dell'oggetto, prezzo, ecc.…) verrà intercettato da un consumatore che si occuperà di memorizzarlo e/o applicare della logica di Business.

Il controllo del flusso del software è quindi regolato da classi Java che fungono da producer e consumer di eventi, che rispondono in base alle attività di un utente.

Le richieste vengono generate da un client, e tramite una classe dedicata a gestire quel determinato evento, vengono presi in carico eventuali input e vengono inizializzate eventuali richieste da inoltrarle alle classi per lo svolgimento dell'operazione associata a quell'evento.

Una volta ottenuto il risultato dell'operazione, la classe gestore si preoccupa di inoltrarlo al client che aveva inoltrato la richiesta, mediante una sua interazione con il sistema.

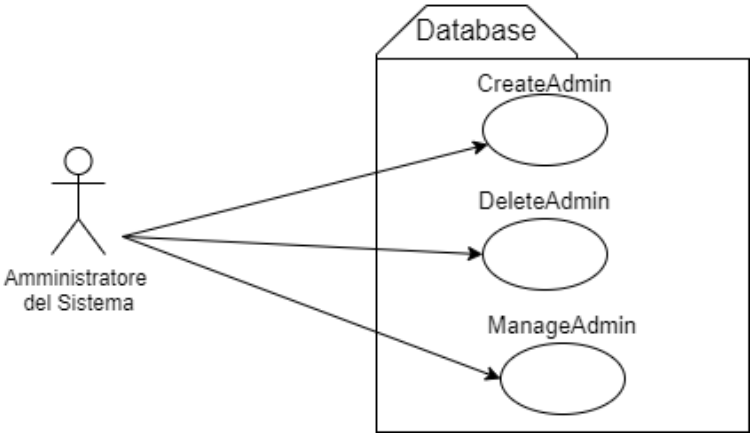
Il sistema software è gestito tramite l'uso di Servlet e JSP. Il server centrale è in grado di intercettare le richieste di un client (web browser) e una volta ricevuta una richiesta, la processa e la smista alla Servlet deputata.

3.7 Boundary conditions

3.7.1 Configurazione

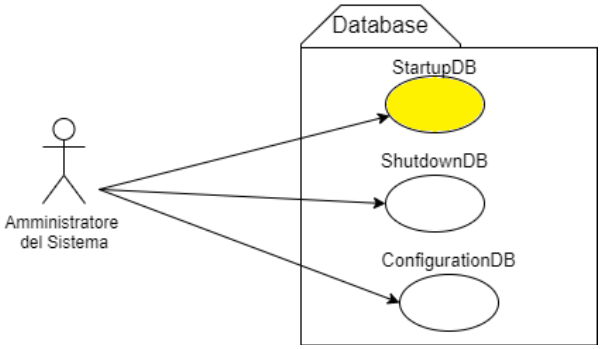
Gli oggetti persistenti presenti nel sistema sono Utente, Quadro e Ordine e Amministratore.
Per quanto riguarda gli oggetti Utente vengono creati nel UC_REG, distrutti nel UC_RIM_UTENTE e modificati nel UC_MOD_UTENTE.
Mentre gli oggetti Quadro vengono creati nel UC_IQ, distrutti nel UC_RIMDIP e modificati nel UC_MOD_DIP.
Gli oggetti Ordine vengono creati nel UC_ACQ e vengono distrutti nel UC_RIM_UTENTE
Gli oggetti di tipo Amministratore sono invece gestiti dall' Amministratore di sistema, non vengono creati tramite una fase di Registrazione, ma sono creati e distrutti tramite database

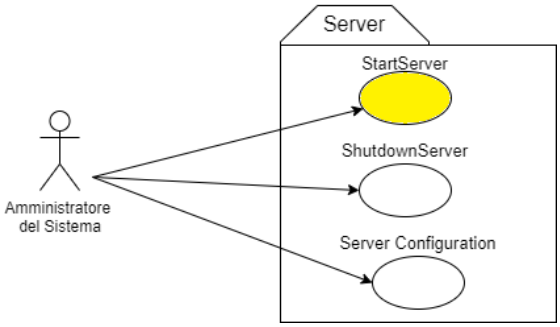
UC_MOD_DIP: Use case modifica di un dipinto
UC_MOD_UTENTE: Use case modifica di un utente
UC_RIM_DIP: Use case rimozione di un dipinto
UC_RIM_UTENTE: Use case rimozione utente
UC_REG: Use case registrazione utente
UC_IQ: Use case Inserimento Quadro
UC_ACQ: Use Case Acquisto



3.7.2 Inizializzazione (start-up)

Non appena il sistema verrà messo online risulterà sempre disponibile.
Lato server, all'inizializzazione del sistema, verrà caricata una pagina di homepage e tramite una richiesta al database verranno caricate tutte le opere disponibili.
All'utente verrà quindi mostrata una hompage con tutte le opere in vendita e un header che comprende le varie sezioni del sito a cui può accedere.

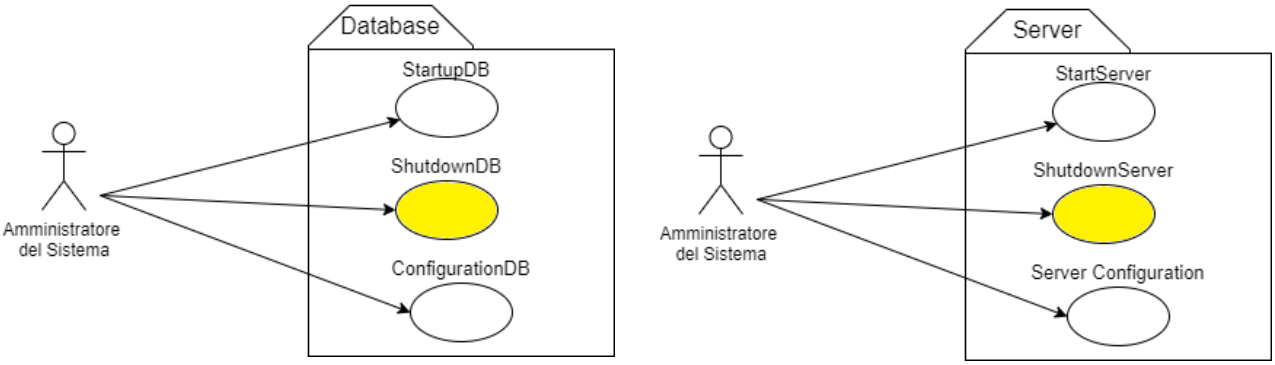




Nome: UC_STARTUP
Attore: Amministratore di sistema
Flusso di eventi
1. Il sistema e tutti i sottosistemi vengono avviati 2. Nel sistema vengono caricate un pool di 10 opere nella pagina principale 3. Viene avviata una sessione
Condizioni in entrata
● Questo caso d'uso inizia quando amministratore di sistema avvia il sistema
Condizioni in uscita
● Questo caso d'uso finisce quando viene avviata una sessione
Eccezioni
● Nessuna.

3.7.3 Terminazione

Il sistema lato server termina solo in caso di guasti o malfunzionamenti.
Il sistema lato client termina alla chiusura della pagina web oppure al logout (dove verrà terminata la sessione relativa all'utente)



Nome: UC_SHUTDOWN
Attore: Amministratore di sistema
Flusso di eventi
1. Le sessioni degli utenti vengono invalidate 2. Viene arrestato il server Database 3. Viene arrestato l'Application Server
Condizioni in entrata
● Questo caso d'uso inizia quando si verifica un malfunzionamento o una terminazione del sistema.
Condizioni in uscita
● Questo caso d'uso finisce quando il sistema non è più accessibile.
Eccezioni
● Nessuna.

3.7.4 Failure

Le failure riguardanti l'hardware possono essere causate da casi eccezionali come guasti o mancanza di connessione che prevedono una conseguente manutenzione per recuperare le funzionalità del sistema.

Le failure relative al software riguardano errori lato server oppure sul database. Vengono gestite tramite eccezioni, che nel migliore dei casi permettono all'utente di continuare o ripetere le proprie azioni, mentre nel caso peggiore il sistema invia un messaggio all'utente per segnalare l'errore, con un suggerimento finale.
Se per un qualche motivo i dati non vengono salvati viene invocata un'eccezione che richiede all'utente di reinserire i dati persistenti, in modo da rendere concluso il task.

Nome: UC_CONNECTION_FAILURE
Attore: Amministratore di sistema
Flusso di eventi
1. Nel sistema si verifica un problema di connessione lato server/database 2. Il sistema tenterà autonomamente di ristabilire la connessione.
Condizioni in entrata
● Questo caso d'uso inizia in caso di problemi di connessione lato server o lato database
Condizioni in uscita
● Questo caso d'uso finisce quando il sistema è riuscito a ristabilire la connessione
Eccezioni
● Nessuna.

Nome: UC_SYSTEM_FAILURE
Attore: Amministratore di sistema
Flusso di eventi
3. Nel sistema si verifica un problema di natura Hardware il quale richiede una rapida assistenza da parte dell'Amministratore di sistema. 4. Viene notificato all'Utente che il sito è temporaneamente in manutenzione
Condizioni in entrata
● Questo caso d'uso inizia in caso di problemi di natura hardware.
Condizioni in uscita
● Questo caso d'uso finisce quando il sistema restituisce un messaggio all'utente informandolo che il sistema è in manutenzione.
Eccezioni
● Nessuna.

Glossario dei servizi dei sottosistemi

Management Utenti

Aggiunta Utente

Aggiunge un Utente nel sistema, fornendo uno username, nome, cognome, password, nazionalità ed e-mail.

Rimozione Utente

Rimozione di un Utente dato l'id dell'utente da rimuovere.

Aggiornamento Utente

Aggiornamento di determinato Utente, fornendo l'Utente con le informazioni alterate.

Reperimento Utente

Reperimento di un determinato Utente, passato un id.

Reperimento di tutti gli Utenti.

Management Quadri

Aggiunta Quadro

Aggiunta di un Quadro fornendo genere, titolo, artista, tecnica, dimensione, immagine, anno, prezzo.

Rimozione Quadro

Rimozione di un Quadro fornendo l'id del quadro da rimuovere.

Aggiornamento Quadro

Aggiornamento di un determinato Quadro, fornendo il Quadro con le informazioni alterate.

Reperimento Quadro

Reperimento di un determinato quadro tramite id
Reperimento di tutti i quadri

Management Ordini

Aggiunta Ordine

Aggiunta di un ordine fornendo il tipo di carta di credito, il num. della carta di credito.

Rimozione Ordine

La rimozione dell'ordine non è un vero e proprio servizio. La elenchiamo qui soltanto perché effettivamente possibile, ma è il risultato del processo di Cascading nel caso di rimozione di un Utente, per non violare il vincolo di Integrità referenziale che vige tra Utente e Ordine.

Reperimento Ordine

Reperimento degli ordini fornendo un Utente.

Acquisto

Operazioni sul Carrello

- Aggiunta di un Quadro nel carrello, fornendo un Quadro.
- Rimozione di un Quadro dal carrello fornendo l'id del Quadro da rimuovere dal carrello.
- Svuotamento del carrello
- Calcolo del totale per i Quadri presenti nel Carrello

Checkout

- Usando congiuntamente i servizi di Management Ordini e Management Quadri e fornendo i Quadri da acquistare e l'Utente che li stà acquistando viene realizzata la fase di Checkout.

Autenticazione

Accesso all'Area Personale

- Usando congiuntamente i servizi di Management Ordini, Management Quadri e Management Utenti, viene realizzato il servizio di Accesso all'Area Personale, il quale, fornendo un Utente, vengono reperiti tutti gli Ordini e di conseguenza tutti i Quadri da esso acquistati, insieme alle informazioni relative al suo account.

Login, Logout e Registrazione

- Usando i servizi di Management Utenti viene creato un nuovo Utente il quale viene memorizzato nel sistema.
- Usando il servizio di reperimento dell'Utente di Management Utenti, viene reperito l'Utente con le credenziali fornite e ne viene effettuato il Login.