

UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA ELETTRICA E TECNOLOGIE DELL'INFORMAZIONE



CORSO DI LAUREA IN INFORMATICA
INSEGNAMENTO DI INGEGNERIA DEL SOFTWARE I
ANNO ACCADEMICO 2023/2024

Specifica, progettazione, implementazione e validazione del Sistema Informativo *"DietiDeals24"*

Autori:

GRUPPO INGSW2324_38

Tommaso SIMIOLI

MATRICOLA N86/757

Luca ESPOSITO

MATRICOLA N86/2565

Riccardo RESCIGNO

MATRICOLA N86/2602

Docente:

Prof. Sergio DI MARTINO

Indice

Capitolo I – Descrizione del progetto	5
1.1 Introduzione	5
Capitolo II – Modello Funzionale	6
2.1 Documento dei Requisiti Software.....	6
2.1.1 Requisiti funzionali.....	6
2.1.2 Requisiti non funzionali	10
2.1.3 Requisiti di dominio	11
2.1.4 Modellazione dei casi d’uso	12
2.1.5 Tabelle di Cockburn dei casi d’uso	13
2.1.5.1 Creazione asta	13
2.1.5.2 Presenta offerta asta	14
2.1.5.3 Effettuare l’accesso	15
2.1.5.4 Visualizzazione del profilo.....	17
2.1.6 Mock-up dell’applicazione.....	18
2.1.6.1 Schermata home	18
2.1.6.2 Schermata ricerca	19
2.1.6.3 Schermata del profilo	20
2.1.6.4 Schermata asta inversa	21
2.1.6.5 Schermata asta silenziosa	22
2.1.6.6 Schermata asta al ribasso.....	23
2.1.6.7 Schermata creazione asta del venditore	24
2.1.6.8 Schermata creazione asta inversa utente	25
2.2 Glossario.....	26
Capitolo III – Modelli di Dominio	27
3.1 Documento dei Requisiti Software.....	27
3.1.1.1 Diagrammi delle classi.....	27
3.1.1.2 Diagrammi di sequenza	29
3.1.1.3 Diagrammi di stato	33
3.1.1.4 Diagramma di attività – Accesso alla piattaforma	34
Capitolo IV – Documento del sistema.....	35
4.1 Documento architettura	35
4.2 Client Layer (Front-End)	35

4.3 Architettura server	36
4.3.1 Web Server Layer	36
4.3.2 Application Server Layer	36
4.3.3 Database Layer	36
4.4 Servizi Cloud Utilizzati	37
4.4.1 Microsoft Azure.....	37
4.4.2 Docker Hub	37
4.4.3 SendGrid	Errore. Il segnalibro non è definito.
Capitolo V – Testing	37
5.1 Documento di testing	37
5.1.1 Codice xUnit per il testing dei metodi.....	37
5.1.1.1 Analisi Classi di Equivalenza – Creazione asta	38
5.1.1.2 Piano di Test: Servizio Aste - Creazione Asta	43
5.1.1.3 Analisi Classi di Equivalenza – Accettazione offerta (Dettagliata).....	47
5.1.1.4 Piano di Test: Servizio Aste - Accettazione Offerta	49
5.1.1.5 Analisi Classi di Equivalenza (Dettagliata) – Creazione offerta	52
5.1.1.6 Piano di Test: Servizio Aste - Creazione Offerta	56
5.1.1.7 Analisi Classi di Equivalenza – Recupero aste utente	59
5.1.1.8 Piano di Test: Servizio Aste - Recupero Aste Utente	63
5.2 Strategia di Test e dipendenze utilizzate per il piano di test.....	65
5.2.1 Strategia di Test Adottata.....	65
5.2.3 Dipendenze Chiave Utilizzate nel Testing	65

Capitolo I – Descrizione del progetto

1.1 Introduzione

Si vuole realizzare un sistema informativo distribuito denominato Deal 24 (oppure DietiSeals24) il cui scopo è quello di fornire una piattaforma che consenta di partecipare e pubblicare aste.

L'applicativo consentirà agli utenti di effettuare ricerche di aste e di visualizzare dei dettagli relativi alla stessa.

La ricerca potrà essere effettuata in diversi modi: mediante parole chiave (ad esempio tutte le aste che contengono la parola “computer” nel nome), oppure con la possibilità di filtrare le aste per categoria (ad esempio informatica, giocattoli, servizi, etc.).

Gli utenti finali potranno registrarsi alla piattaforma la quale implicherà l'accesso a funzioni aggiuntive, quali:

- Avere accesso ad un profilo personalizzabile
L'utente avrà la possibilità di modificare il proprio profilo
- Partecipare alle aste correttamente attive
L'utente può presentare un'offerta per una o più aste attive

Capitolo II – Modello Funzionale

2.1 Documento dei Requisiti Software

2.1.1 Requisiti funzionali

Vengono ora elencati i requisiti funzionali ovvero tutti i servizi (o funzioni) che il sistema deve offrire.

ID	APP_REQF01
NOME	Registrazione alla piattaforma
DESCRIZIONE	Il sistema deve consentire ad un utente non registrato di potersi registrare alla piattaforma indicando username, e-mail, nome, cognome e password.

ID	APP_REQF02
NOME	Registrazione alla piattaforma come venditore
DESCRIZIONE	Il sistema deve consentire ad un utente, venditore, non registrato di potersi registrare alla piattaforma indicando username, e-mail, nome, cognome e password.

ID	APP_REQF03
NOME	Accesso alla piattaforma
DESCRIZIONE	Il sistema deve consentire ad un utente registrato di poter effettuare l'accesso alla piattaforma includendo username e password.

ID	APP_REQF04
NOME	Accesso alla piattaforma
DESCRIZIONE	Il sistema deve consentire ad un utente registrato come venditore di poter effettuare l'accesso alla piattaforma includendo username e password.

ID	APP_REQF05
NOME	Effettuare una ricerca e filtraggio risultati
DESCRIZIONE	Il sistema deve consentire ad un <i>utente</i> di poter effettuare una ricerca delle aste. La ricerca avviene mediante l'inserimento di una parola chiave o categoria. I risultati possono essere filtrati per tipo di aste.

ID	APP_REQF06
NOME	Visualizzazione delle aste attive
DESCRIZIONE	Il sistema deve consentire ad un utente di poter visualizzare nel dettaglio le aste attive in modo da poter consultare le informazioni della stessa.

ID	APP_REQF07
NOME	Visualizzazione e partecipazione delle aste attive
DESCRIZIONE	Il sistema deve consentire ad un utente di poter visualizzare nel dettaglio le aste attive in modo da poter consultare le informazioni e poter partecipare alla stessa presentando un'offerta.

ID	APP_REQF08
NOME	Visualizzazione delle aste attive
DESCRIZIONE	Il sistema deve consentire ad un utente di poter visualizzare nel dettaglio le aste attive in modo da poter consultare le informazioni della stessa.

ID	APP_REQF09
NOME	Visualizzazione del profilo
DESCRIZIONE	Il sistema deve consentire ad un utente registrato di poter visualizzare il proprio profilo o del venditore, nel quale sono presenti le seguenti informazioni: nome, cognome, età, numero di aste, paese di appartenenza, link proprio sito web.

2.1.2 Requisiti non funzionali

ID	APP_REQNF01
NOME	Performance di ricerca
DESCRIZIONE	Il sistema deve mostrare i risultati di una ricerca effettuata entro 3 secondi, in quasi il 90% dei casi.
ID	APP_REQNF02
NOME	Usabilità dell'applicazione
DESCRIZIONE	Un utente deve riuscire ad utilizzare la piattaforma in tutte le sue funzionalità, dopo circa una media di 2 ore di utilizzo.
ID	APP_REQNF03
NOME	Limitazioni dell'accesso ai dati
DESCRIZIONE	Il sistema non deve avere accesso diretto al database.
ID	APP_REQNF04
NOME	Policy password
DESCRIZIONE	Il sistema deve obbligare un utente ad inserire una password di almeno 8 caratteri.
ID	APP_REQNF05
NOME	Flessibilità del back-end
DESCRIZIONE	Il sistema deve essere flessibile alla possibilità di eventuali cambiamenti evolutivi del back-end e quindi essere disaccoppiato rispetto al front-end.

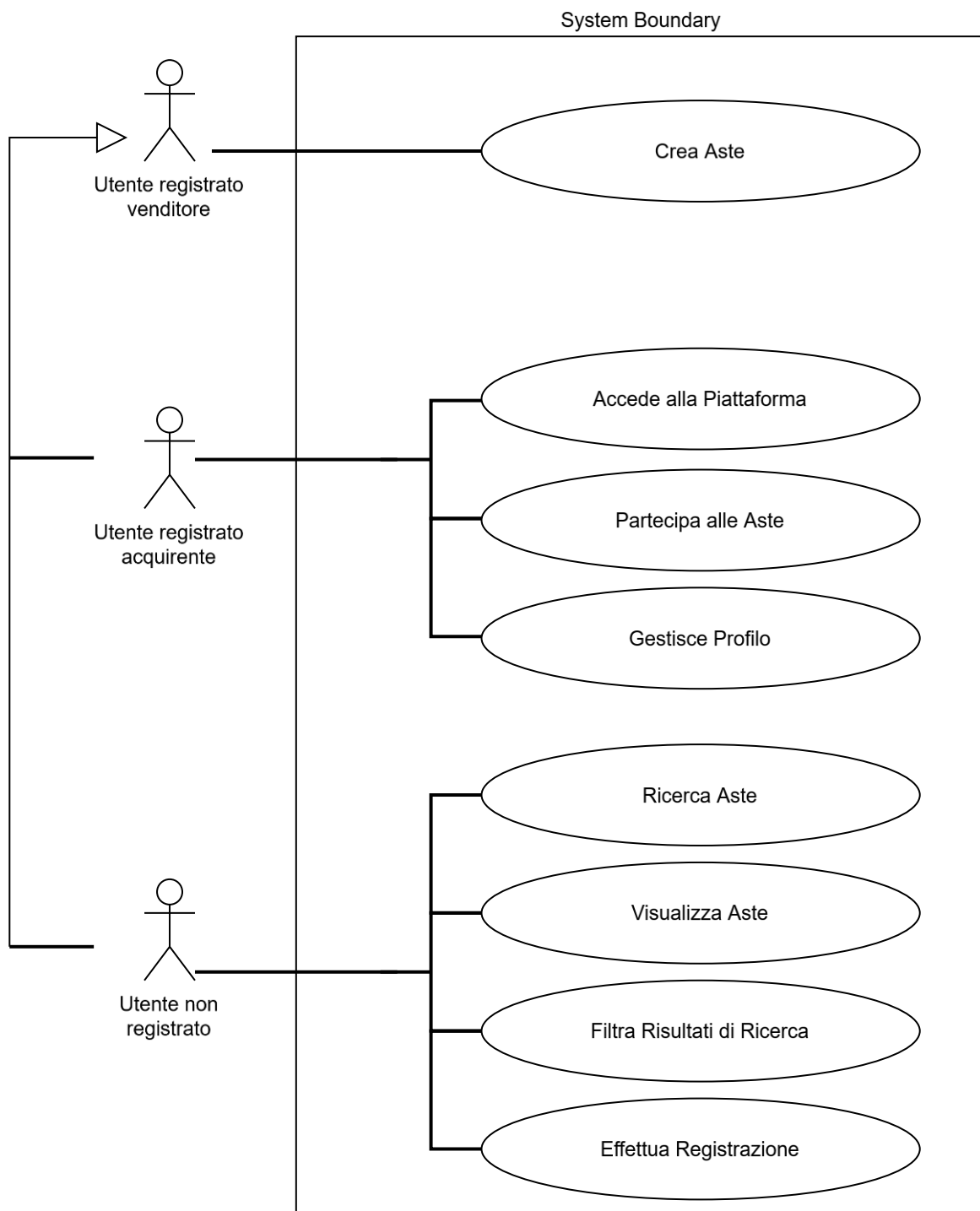
2.1.3 Requisiti di dominio

Vengono elencati i requisiti di dominio, ovvero i vincoli generali ai quali l'applicativo deve attenersi

ID	APP_REQD01
NOME	GDPR
DESCRIZIONE	Il sistema deve essere conforme al GDPR (Regolamento Generale sulla Protezione dei Dati), relativo al trattamento dei dati personali e privacy dell'utente.
ID	APP_REQD02
NOME	Standard ISO/IEC 27000
DESCRIZIONE	Il sistema deve essere conforme quanto più è possibile allo standard ISO/IEC 27000 che comprende una serie di norme ISO per la sicurezza e gestione delle informazioni.

2.1.4 Modellazione dei casi d'uso

Di seguito viene riportato lo Use Case Diagram relativo al sistema.



2.1.5 Tabelle di Cockburn dei casi d'uso

Di seguito vengono elencate le tabelle di Cockburn relativi ai casi d'uso dello Use Case Diagram riportato nel precedente paragrafo.

2.1.5.1 Creazione asta

USE CASE #1	Crea asta		
Goal in Context	L'utente vuole creare un'asta		
Preconditions	L'utente deve essere registrato		
Success End Condition	L'utente riesce a creare un'asta		
DESCRIPTION	Step	Utente	Sistema
	1	L'utente preme sul pulsante per creare un'asta in base al tipo che desidera	
	2		Mostra il mockup_create_deals
	3	L'utente compila tutti i campi della form e clicca su crea asta	
	4		Crea l'asta e la mostra mockup_deals_down/reverse/silent
SUBVARIATION #1	Step	Utente	Sistema
(Branching #4)			

2.1.5.2 Presenta offerta asta

USE CASE #2	Partecipa alle aste		
Goal in Context	L'utente vuole partecipare ad un'asta		
Preconditions	L'utente deve essere autenticato		
Success End Condition	L'utente riesce a partecipare ad un'asta attiva		
DESCRIPTION	Step	Utente	Sistema
	1	Seleziona l'asta a cui desidera partecipare	
	2		Mostra uno dei mockup_deals_reverse/down/silent_logged
	3	Invia l'offerta a seconda del tipo di asta selezionata	
	4		Mostra un messaggio del corretto invio dell'offerta
SUBVARIATION #1	Step	Utente	Sistema
(Branching #3) L'utente non compila tutti i campi	1		Mostra un messaggio su quale campo bisogna compilare
	2		Torna allo step 3 dello scenario principale

2.1.5.3 Effettuare l'accesso

USE CASE #3	Accede alla piattaforma		
Goal in Context	L'utente vuole accedere all'applicazione		
Preconditions	L'utente non è autenticato		
Success End Condition	L'utente riesce ad accedere all'applicazione		
DESCRIPTION	Step	Utente	Sistema
	1	L'utente accede all'applicazione	
	2		Mostra la schermata di autenticazione mockup_login
	3	Inserire Username e Password e premere Accedi	
	4		Mostra la schermata il mockup_home_logged
EXTENSION #1 L'utente preme "Home" annullando l'operazione	Step	Utente	Sistema
	3A	Preme torna alla "Home"	
	4a		Ritorna alla schermata mockup_home_logged

SUBVARIATION #1 L'utente non compila uno o entrambi i campi	Step	Utente	Sistema
	1		Posiziona il cursore nel campo non compilato e lo evidenzia
	2		Torna allo step 3 dello scenario principale
SUBVARIATION #2 L'utente compila i campi con dati errati	Step	Utente	Sistema
	1		Mostra un messaggio di errore
	2		Torna allo step 3 dello scenario principale

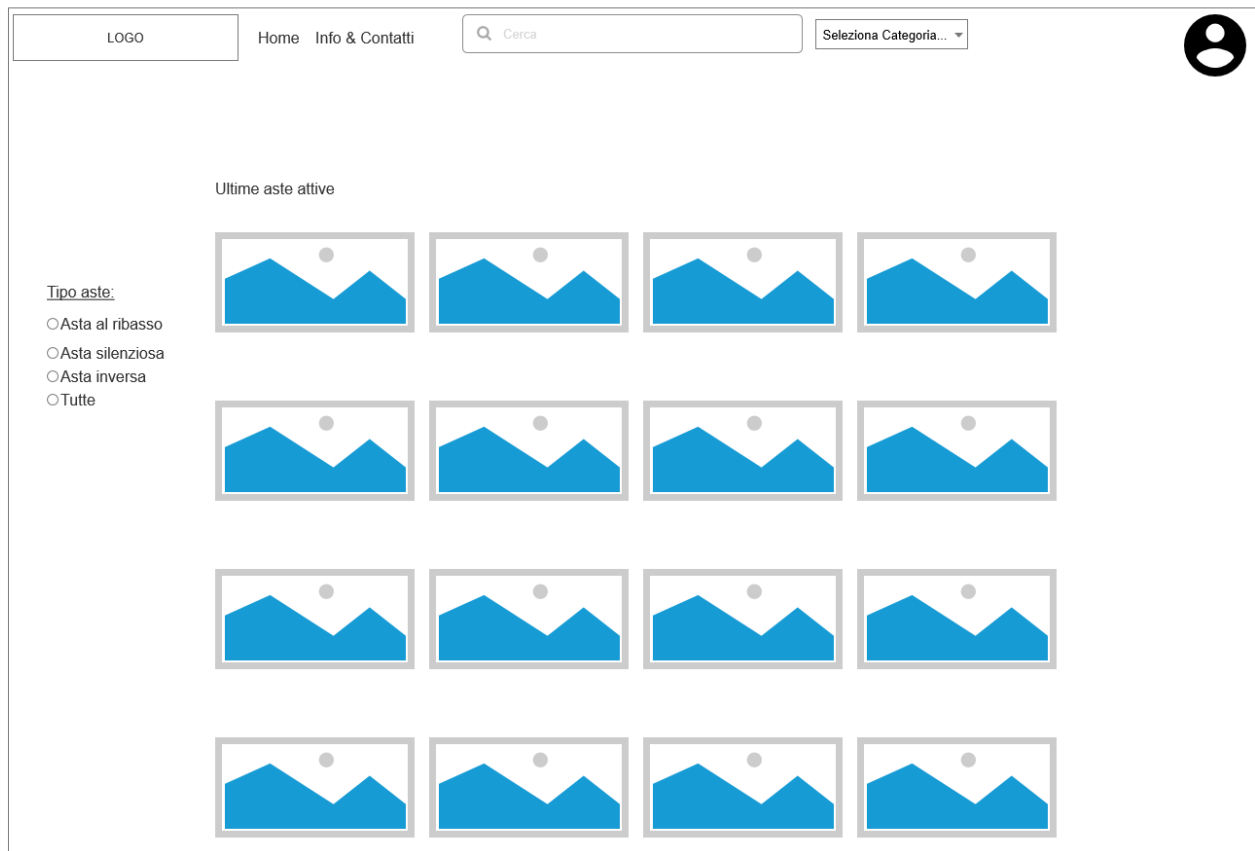
2.1.5.4 Visualizzazione del profilo

USE CASE #4	Visualizza il profilo		
Goal in Context	L'utente vuole visualizzare il proprio profilo		
Preconditions	L'utente è regolarmente autenticato		
Success End Condition	L'utente riesce a visualizzare il proprio profilo		
DESCRIPTION	Step	Utente registro	Sistema
	1	Preme il pulsante del proprio profilo dalla schermata home mockup_home_logged	
	2		Mostra la schermata del proprio profilo mockup_user_logged

2.1.6 Mock-up dell'applicazione

Vengono ora mostrati i mock-up dell'applicazione.

2.1.6.1 Schermata home




2.1.6.2 Schermata ricerca

LOGO

Home Info & Contatti

Cerca

Seleziona Categoria... ▾



Tipo aste:

☐Asta al ribasso

☐Asta silenziosa

☐Asta inversa

☐Tutte

Aste al ribasso / Silenziose / Inverse / Servizi

Prezzo:

Tipologia di asta:

Tempo restante:

19


2.1.6.3 Schermata del profilo


LOGO

Home Info & Contatti

Cerca




Seleziona Categoria...





Nome:
Cognome:
Età:
Pese:
Regione:
Provincia:

Interessi:
Hobby:
Professione:
Titolo di Studio:
Luoghi Visitati:
Sito Web:





LOGO

Home Info & Contatti

Cerca

Seleziona Categoria...








Nome:
Cognome:
Età:
Pese:
Regione:
Provincia:

Interessi:
Hobby:
Professione:
Titolo di Studio:
Luoghi Visitati:
Sito Web:

Modifica




Profilo seller mod


2.1.6.4 Schermata asta inversa

LOGO

Home Info & Contatti

Seleziona Categoria...





Spese di spedizione:

Peso:

Dimensioni:

Specifiche Tecniche:

Tipo asta:

Categoria:

User Profile

L'offerta scadrà il xx/xx/2024

50€


Fai un'offerta


2.1.6.5 Schermata asta silenziosa

LOGO

Home Info & Contatti

Seleziona Categoria... ▼





Spese di spedizione:

Peso:

Dimensioni:

Specifiche Tecniche:

Tipo asta:

Categoria:

[Seller Profile](#)

L'offerta scadrà il xx/xx/2024


Fai un'offerta


2.1.6.6 Schermata asta al ribasso

LOGO

Home Info & Contatti

Seleziona Categoria... ▼





Spese di spedizione:

Peso:

Dimensioni:

Specifiche Tecniche:

Tipo asta:


Categoria:

Seller Profile


2.1.6.7 Schermata creazione asta del venditore

LOGO

Home Info & Contatti

 Cerca

Seleziona Categoria... ▼



Titolo:

Prezzo:

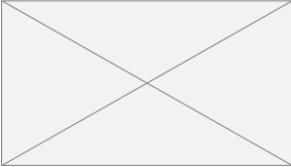
Data fine asta:

Tipologia asta:

Asta al ribasso ▼

Categoria:

Seleziona una categoria... ▼



Carica immagine

Inserisci una descrizione:


Invia

2.1.6.8 Schermata creazione asta inversa utente

LOGO

Home Info & Contatti

Seleziona Categoria... ▼



Titolo:

Prezzo:

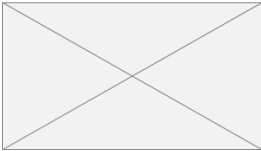
Data fine asta:

Tipologia asta:

Asta inversa ▼

Categoria

Seleziona una categoria... ▼



Carica immagine

Inserisci una descrizione:

Invia

Crea asta seller

2.2 Glossario

Termine		Descrizione
1	Utente	Con il termine utente si intende indifferentemente sia un utente registrato che non.
2	Standard ISO/IEC 27000	La serie ISO/IEC 27000 "Information Security Management Systems (ISMS) Family of Standards" (anche nota, in Italia, come famiglia di norme SGSI, "Sistemi di Gestione per la Sicurezza delle Informazioni") è uno standard di sicurezza informatica redatto dalla ISO. Raggruppa un insieme di norme internazionali che si prefiggono di proteggere le informazioni che vengono mantenute ed elaborate da un'organizzazione. Attraverso questa famiglia di norme, le organizzazioni possono sviluppare ed implementare un proprio sistema per la gestione della sicurezza informatica per le informazioni finanziarie, la proprietà intellettuale ed i dati degli addetti, di clienti o di terzi. Le informazioni vengono protette da possibili attacchi informatici, errori umani, calamità naturali o da qualsiasi altra vulnerabilità che si può presentare durante l'utilizzo di un sistema informatico.
3	GDPR	Il regolamento generale sulla protezione dei dati in sigla RGPD (o GDPR in inglese General Data Protection Regulation). Con questo regolamento, la Commissione europea si propone come obiettivo quello di rafforzare la protezione dei dati personali di cittadini dell'Unione europea (UE) e dei residenti nell'UE, sia all'interno che all'esterno dei confini dell'UE, restituendo ai cittadini il controllo dei propri dati personali, semplificando il contesto normativo che riguarda gli affari internazionali, unificando e rendendo omogenea la normativa privacy dentro l'UE.

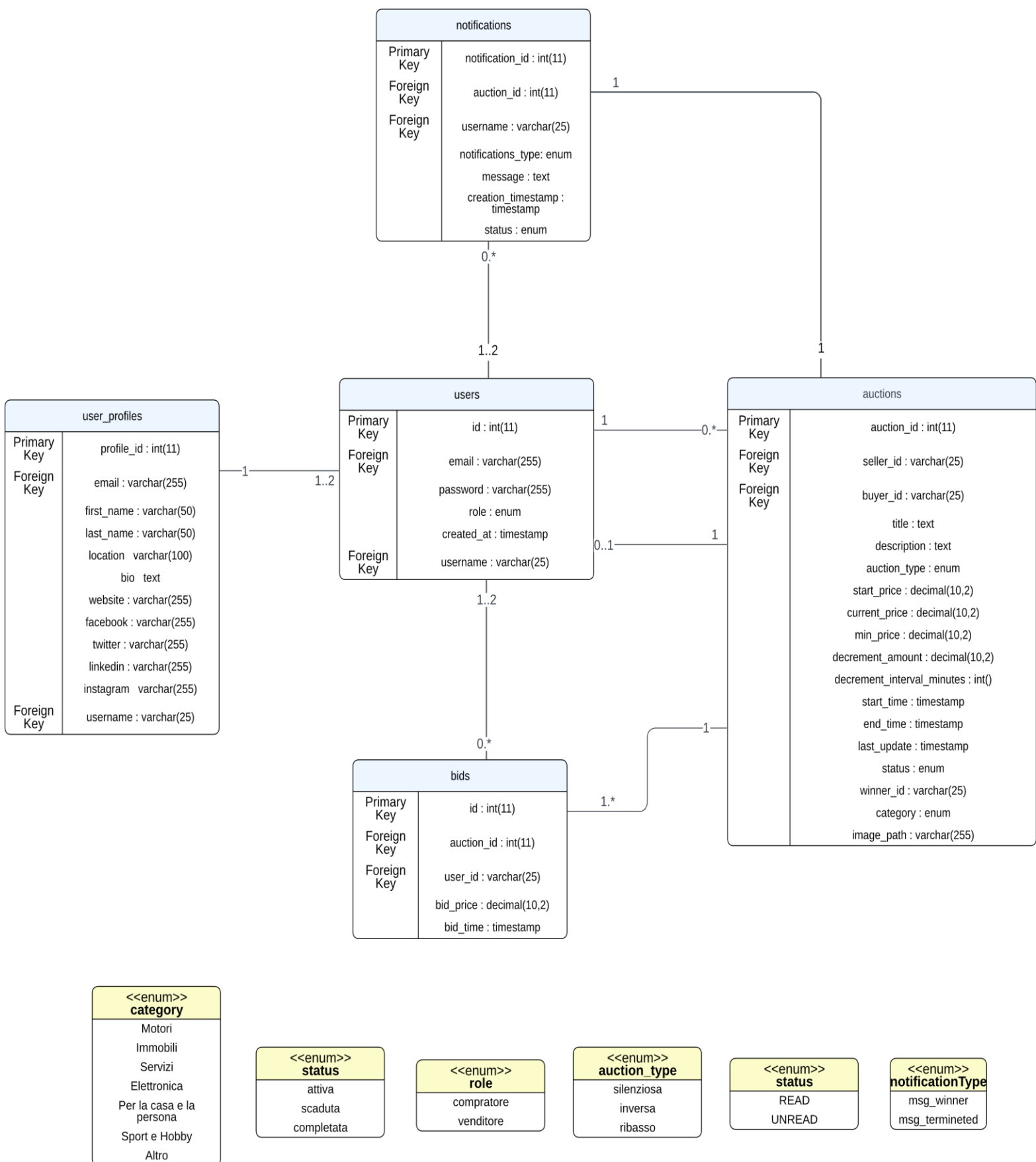
Capitolo III – Modelli di Dominio

3.1 Documento dei Requisiti Software

3.1.1 Modelli di Dominio

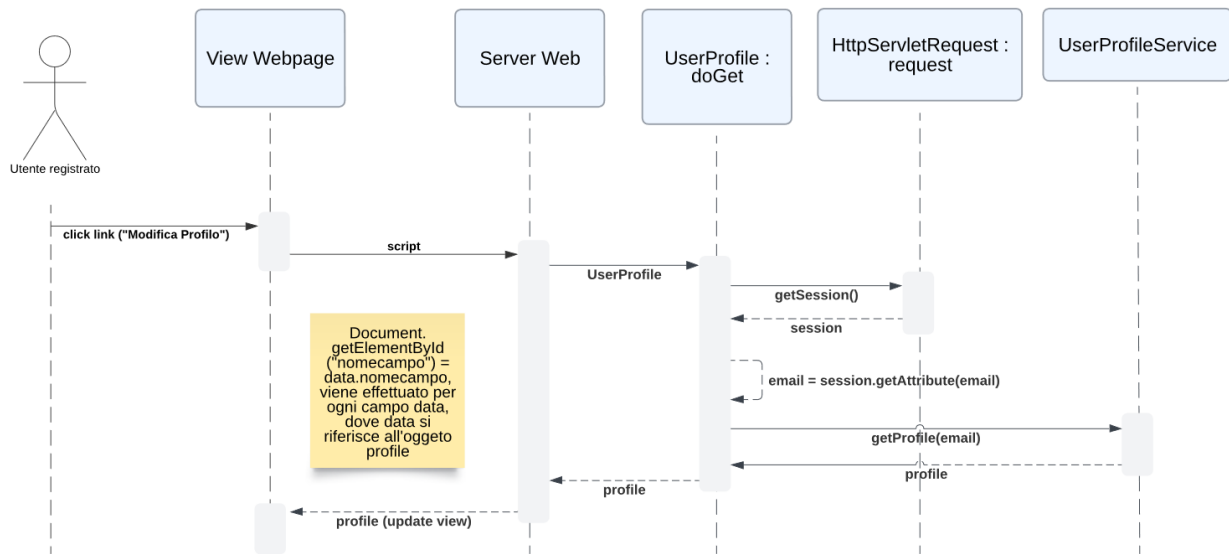
3.1.1.1 Diagrammi delle classi

Vengono di seguito elencati i Class Diagram relativi alle funzionalità che l'applicazione deve offrire.

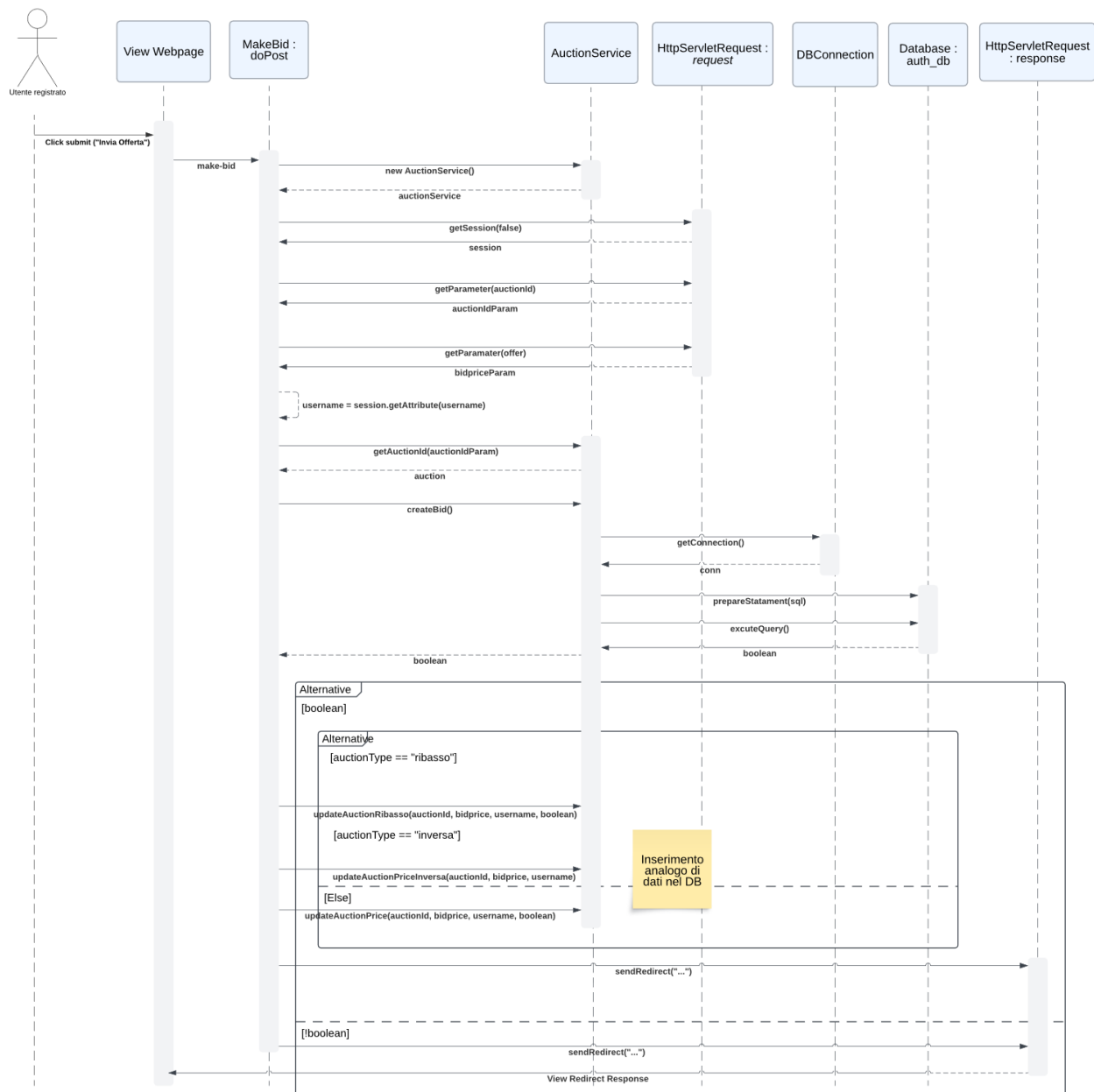


3.1.1.2 Diagrammi di sequenza

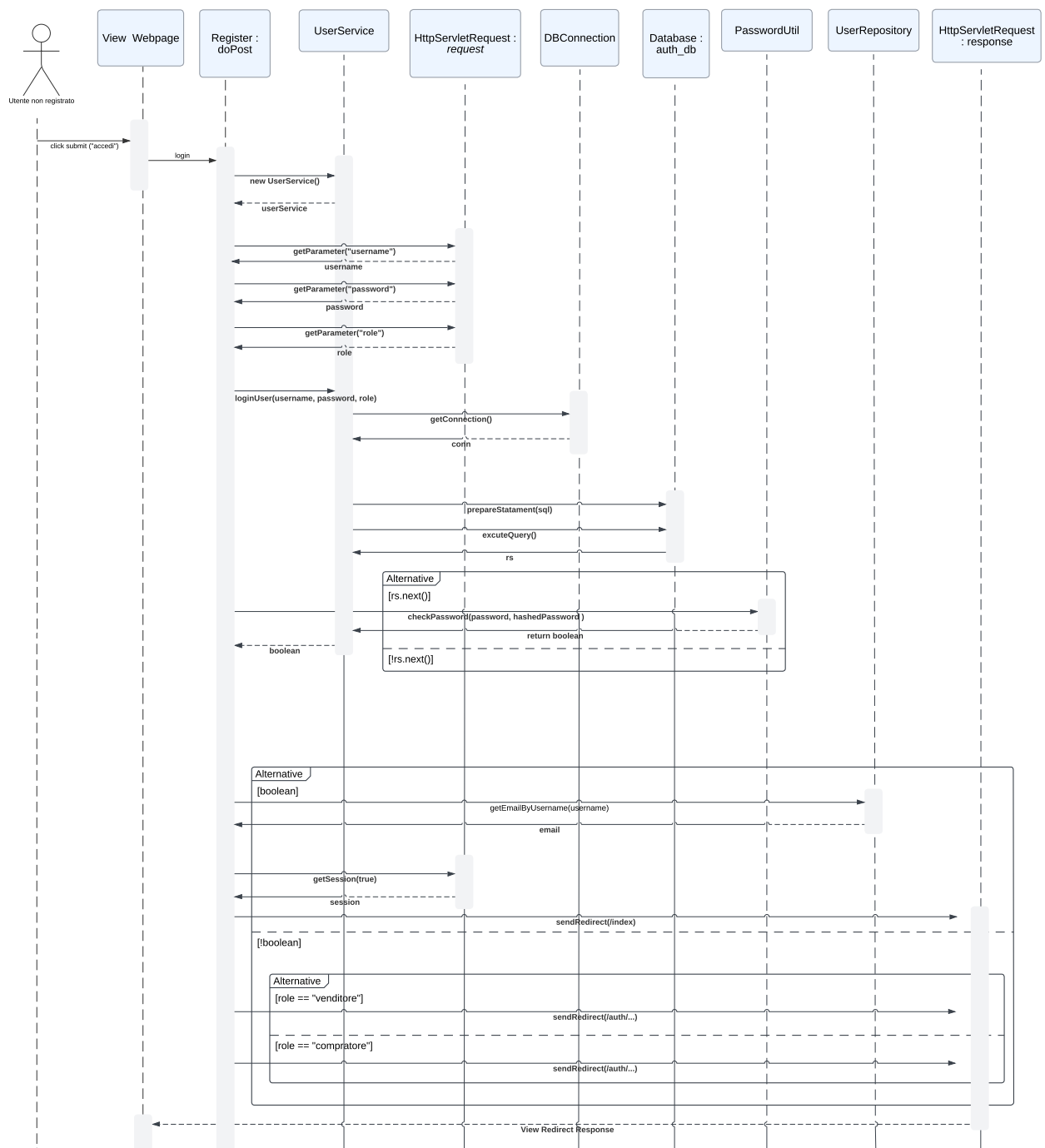
Vengono di seguito elencati i Sequence Diagram relativi alle funzioni che l'applicazione deve offrire.



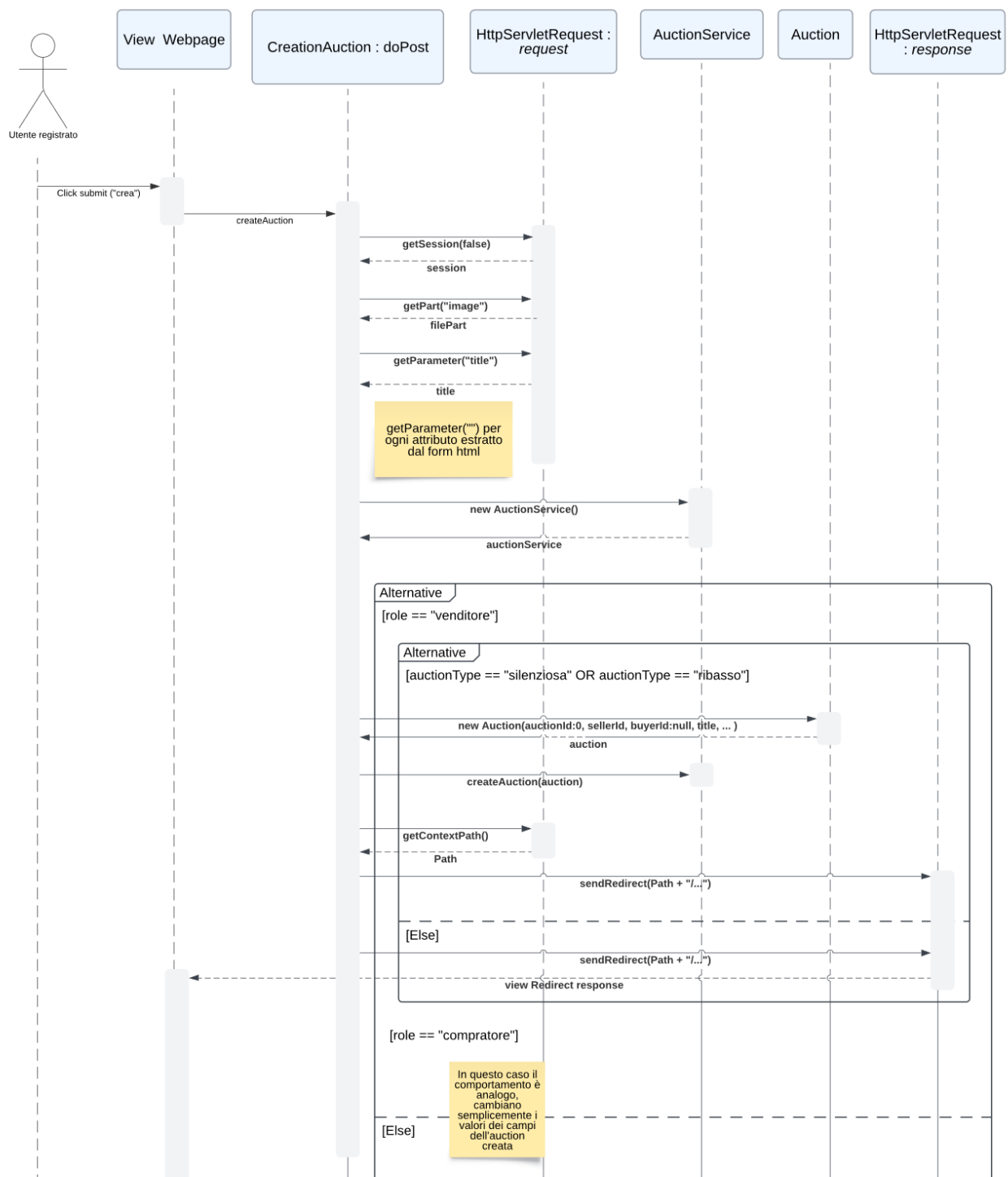
Visualizza profilo



Partecipa alle aste

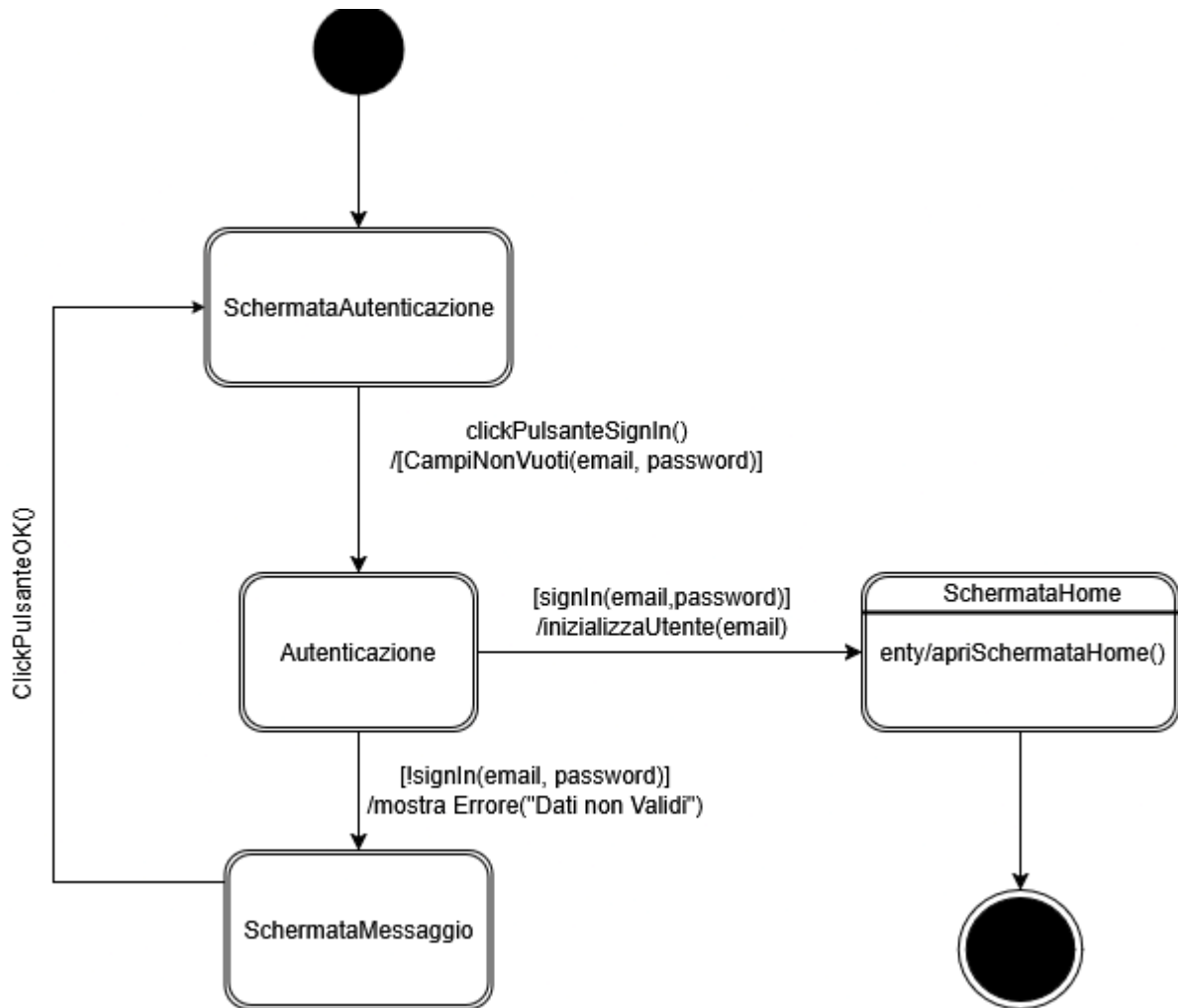


Effettua accesso all'applicazione

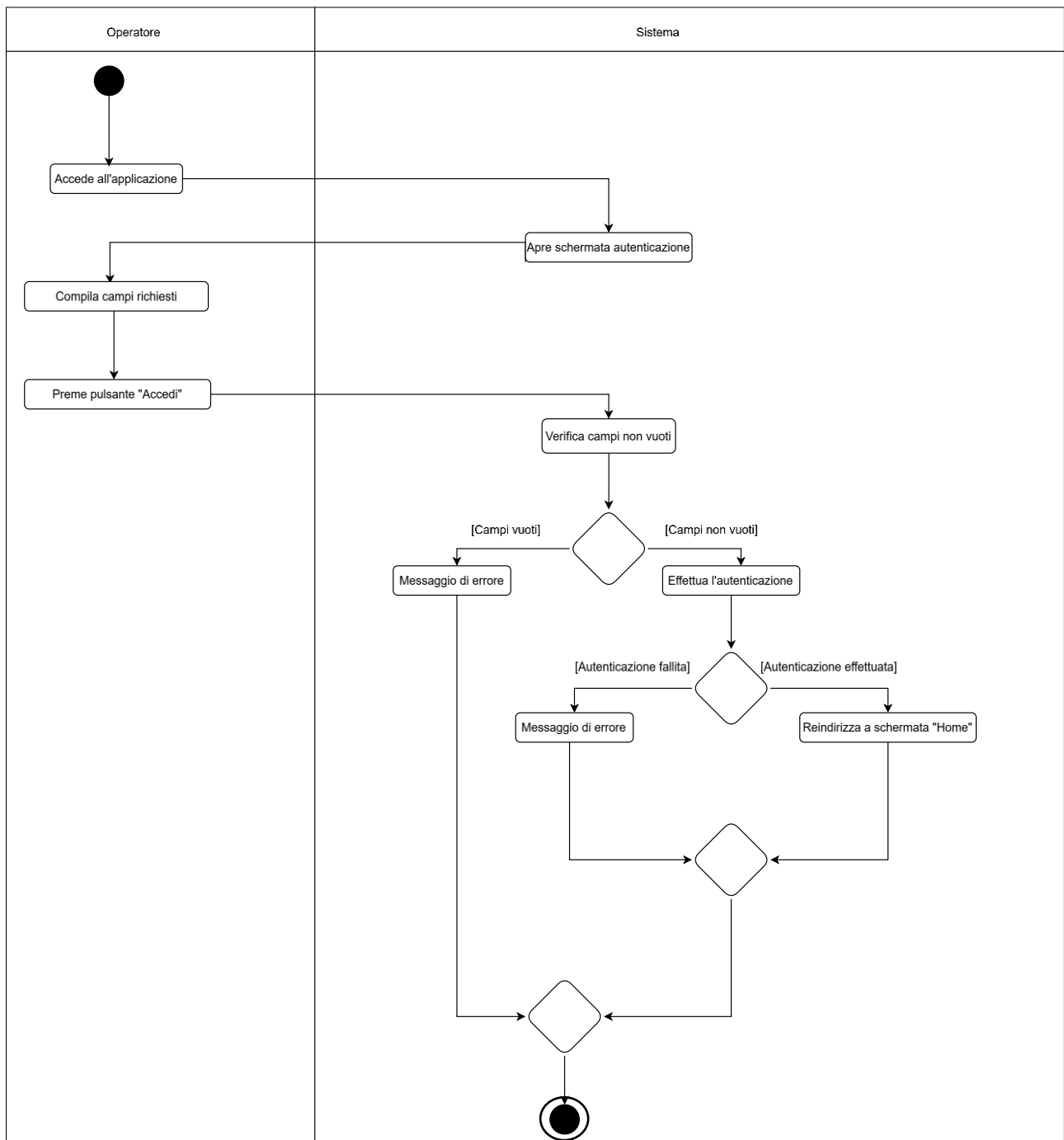


Crea Asta

3.1.1.3 Diagrammi di stato



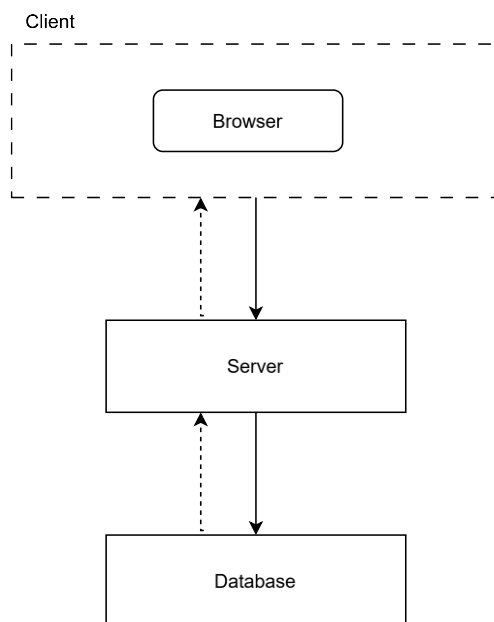
3.1.1.4 Diagramma di attività – Accesso alla piattaforma



Capitolo IV – Documento del sistema

4.1 Documento architettura

L'architettura esterna del sistema è suddivisa in tre differenti layers, ognuno dei quali può accedere esclusivamente al layer immediatamente sottostante, rendendo l'architettura chiusa.



I client inviano eventuali richieste al back-end che li elabora e li propaga al database. Il back-end offre un'interfaccia sicura per l'indicizzazione e il recupero dei dati in quanto non consente chiamate dirette sul database, filtra eventuali richieste dannose, differenzia i diversi tipi di client che fanno richieste o che vogliono compiere una determinata azione.

4.2 Client Layer (Front-End)

L'interfaccia utente è realizzata con HTML e Bootstrap, che garantiscono una visualizzazione responsiva e moderna su qualsiasi dispositivo. JavaScript è stato impiegato non solo per aggiornare dinamicamente alcuni contenuti HTML, ma anche per eseguire validazioni e controlli sui campi dei form per salvaguardare gli input e migliorare l'esperienza utente. In aggiunta, JavaScript viene utilizzato per eseguire fetch di dati provenienti dalle servlet, consentendo un'interazione più dinamica con il server. In questo contesto, Thymeleaf viene impiegato per la renderizzazione dinamica delle pagine, integrando i dati provenienti dal back-end direttamente nel markup HTML, e mantenendo il codice organizzato e modulare.

4.3 Architettura server

4.3.1 Web Server Layer

Apache funge da server HTTP principale, gestendo tutte le richieste in ingresso. È configurato per servire file statici (CSS, JavaScript, immagini) e per fungere da reverse proxy, instradando le richieste dinamiche verso il server applicativo Tomcat. Questa configurazione migliora le performance e la sicurezza, isolando i file statici dal back-end e garantendo un routing efficiente delle richieste.

4.3.2 Application Server Layer

Sul server Tomcat viene distribuita l'applicazione Java, che viene sviluppata e gestita tramite Maven per il build e la gestione delle dipendenze. Le servlet in Tomcat elaborano le richieste dal front-end, interagendo con i modelli (model) Java per la gestione, la validazione e la manipolazione dei dati. Tecnologie come jBCrypt vengono utilizzate per la gestione sicura delle password, mentre MySQL Connector/J (compatibile anche con MariaDB) permette la connessione al database. Gson si occupa della serializzazione/deserializzazione JSON per facilitare la comunicazione tramite API, le quali sono implementate come servlet, e permettono operazioni CRUD. Thymeleaf integra i dati nei template HTML per generare pagine dinamiche da restituire al client.

4.3.3 Database Layer

I dati dell'applicazione sono gestiti da MariaDB, un sistema di gestione di database relazionali, mentre phpMyAdmin viene utilizzato per l'amministrazione e il monitoraggio del database. Le servlet in Tomcat si collegano a MariaDB per eseguire operazioni di lettura e scrittura, garantendo che i dati siano correttamente archiviati e accessibili, e il tutto viene integrato nei template HTML renderizzati e inviati al client.

4.4 Servizi Cloud Utilizzati

L'architettura dell'applicazione si basa su diversi servizi cloud per garantire scalabilità, accessibilità e gestione efficiente delle risorse.

4.4.1 Microsoft Azure

Microsoft Azure è la piattaforma cloud utilizzata per il lavoro in team, permettendo di applicare modifiche da diversi dispositivi in modo flessibile. Inoltre, consente di esporre l'applicazione su un dominio pubblico, rendendola accessibile ovunque.

L'applicazione è ospitata su una macchina virtuale Ubuntu su Azure, che fornisce un ambiente stabile e gestito, ottimizzando le risorse e garantendo la scalabilità automatica in base al carico di lavoro. Il monitoraggio delle performance e i sistemi di sicurezza avanzati, come firewall e crittografia, assicurano stabilità e protezione dei dati.

4.4.2 Docker Hub

Docker Hub è utilizzato per la gestione e distribuzione delle immagini container, semplificando il deployment dell'applicazione. Grazie alla containerizzazione, l'intero stack è portabile e replicabile su diversi ambienti.

Sulla macchina virtuale Ubuntu, l'applicazione è eseguita tramite docker-compose, che gestisce la configurazione e l'orchestrazione dei container, garantendo un'esecuzione coerente e facilitando aggiornamenti e gestione delle versioni.

Capitolo V – Testing

5.1 Documento di testing

5.1.1 Codice xUnit per il testing dei metodi

“Il codice è incluso nella build consegnata. Ci è sembrato superfluo aggiungere gli screen all'interno del documento.”

5.1.1.1 Analisi Classi di Equivalenza – Creazione asta

Funzionalità: Creazione Asta (`AuctionService.createAuction`)

Questa tabella identifica le Classi di Equivalenza per i parametri di input rilevanti del metodo `createAuction`. Una classe di equivalenza raggruppa valori di input che si prevede vengano trattati dal sistema in modo simile. L'obiettivo è selezionare almeno un valore da ogni classe per progettare test efficaci (secondo la strategia Black-Box).

Nome CE	Parametro (Campo di `Auction`)	Descrizione Valori / Condizione	Effetto Atteso (<code>createAuction`</code>)
TitleValid	<code>title</code>	Stringa non vuota e non nulla	Successo (se altri validi)
TitleInvalid_Null	<code>title</code>	<code>null</code>	Fallimento (`false`) Viola vincolo NOT NULL DB
TypeValid	<code>auctionType</code>	Una tra "ribasso", "silenziosa", "inversa"	Successo (se altri validi)
TypeInvalid_Null	<code>auctionType</code>	<code>null</code>	Fallimento (`false`) Viola vincolo NOT NULL DB
TypeInvalid_Other	<code>auctionType</code>	Stringa diversa da quelle valide	Fallimento (`false`) Viola vincolo ENUM/CHECK DB

Nome CE	Parametro (Campo di `Auction`)	Descrizione Valori / Condizione	Effetto Atteso (`createAuction`)
StartPriceValid	<code>startPrice</code>	<code>BigDecimal >= 0</code> (o <code>null</code> se permesso)	Successo (se altri validi)
StartPriceInvalid_Neg	<code>startPrice</code>	<code>BigDecimal < 0</code>	Fallimento (<code>false`</code>) Viola vincolo <i>CHECK >= 0</i>
MinPriceValid_Null	<code>minPrice</code>	<code>null</code>	Successo (se altri validi)
MinPriceValid_Ok	<code>minPrice</code>	<code>BigDecimal >= 0</code> E <code>minPrice <= startPrice</code> (se <code>startPrice</code> non è <code>null</code>)	Successo (se altri validi)
MinPriceInvalid_Neg	<code>minPrice</code>	<code>BigDecimal < 0</code>	Fallimento (<code>false`</code>) Viola vincolo <i>CHECK >= 0</i>
MinPriceInvalid_TooHigh	<code>minPrice</code>	<code>minPrice > startPrice</code> (quando entrambi non nulli)	Fallimento (<code>false`</code>) Viola vincolo <i>CHECK min <= start</i>
DecrAmtValid_Ribasso	<code>decrementAmount</code>	(Tipo 'ribasso') <code>BigDecimal > 0</code>	Successo (se altri validi)

Nome CE	Parametro (Campo di `Auction`)	Descrizione Valori / Condizione	Effetto Atteso (`createAuction`)
DecrAmtValid_Other	<code>decrementAmount</code>	(Tipo != 'ribasso') <code>null</code>	Successo (se altri validi)
DecrAmtInvalid_NonPos	<code>decrementAmount</code>	(Tipo 'ribasso') <code>BigDecimal <= 0</code>	Fallimento (`false`) Viola vincolo CHECK > 0
DecrIntValid_Ribasso	<code>decrementIntervalMinutes</code>	(Tipo 'ribasso') <code>int</code> compreso tra 1 e 720	Successo (se altri validi)
DecrIntValid_Other	<code>decrementIntervalMinutes</code>	(Tipo != 'ribasso') `0` (o valore ignorato dal CHECK)	Successo (se altri validi)
DecrIntInvalid_Low	<code>decrementIntervalMinutes</code>	(Tipo 'ribasso') `int < 1`	Fallimento (`false`) Viola vincolo CHECK >= 1
DecrIntInvalid_High	<code>decrementIntervalMinutes</code>	(Tipo 'ribasso') `int > 720`	Fallimento (`false`) Viola vincolo CHECK <= 720
StartTimeValid	<code>startTime</code>	<code>Timestamp</code> non nullo	Successo (se altri validi)
StartTimeInvalid_Null	<code>startTime</code>	<code>null</code>	Fallimento (`false`) Viola vincolo NOT NULL DB

Nome CE	Parametro (Campo di `Auction`)	Descrizione Valori / Condizione	Effetto Atteso (<code>createAuction`</code>)
EndTimeValid	<code>endTime</code>	<code>Timestamp</code> non nullo E <code>`>=</code> <code>startTime + 1 ora`</code>	Successo (se altri validi)
EndTimeInvalid_Null	<code>endTime</code>	<code>null</code>	Fallimento (`false`) <i>Viola vincolo NOT NULL DB</i>
EndTimeInvalid_TooSoon	<code>endTime</code>	<code>Timestamp < startTime + 1 ora`</code>	Fallimento (`false`) <i>Viola vincolo CHECK tempo</i>
CategoryValid	<code>category</code>	Stringa valida dall'enum/elenco consentito ("Motori", ecc.)	Successo (se altri validi)
CategoryInvalid	<code>category</code>	<code>null</code> (se non permesso) o stringa non nell'enum	Fallimento (`false`) <i>Viola vincolo DB</i>
SellerIdLogic	<code>sellerId</code>	<code>null</code> per 'inversa', non <code>null</code> altrimenti	Successo (se altri validi)
BuyerIdLogic	<code>buyerId</code>	non <code>null</code> per 'inversa', <code>null</code> altrimenti	Successo (se altri validi)

Nome CE	Parametro (Campo di `Auction`)	Descrizione Valori / Condizione	Effetto Atteso (`createAuction`)
StatusLogic	<code>status</code>	Valore valido ('attiva', 'scaduta') o gestito dal service	Successo (se altri validi)

5.1.1.2 Piano di Test: Servizio Aste - Creazione Asta

Test ID:

AS_CA_T01

Test Name:

Test Funzionalità Creazione Asta (`createAuction`)

Test Description:

Verificare la corretta gestione della creazione di aste tramite il metodo `createAuction`, includendo casi validi per diversi tipi d'asta (ribasso, silenziosa, inversa) e la gestione di input invalidi secondo le regole di business e i vincoli di integrità definiti (campi obbligatori, prezzi validi, intervalli, ordine temporale).

Dettaglio Test Case

Input (Scenario Specifico)	Risultato Desiderato	Risultato Ottenuto
Casi di Successo		
Creazione Asta 'ribasso' valida	Il metodo `createAuction` restituisce `true`.	✓
Creazione Asta 'silenziosa' valida	Il metodo `createAuction` restituisce `true`.	✓
Creazione Asta 'inversa' valida	Il metodo `createAuction` restituisce `true`.	✓

Input (Scenario Specifico)	Risultato Desiderato	Risultato Ottenuto
Verifica stato Asta 'silenziosa' valida	`createAuction` ritorna `true` E l'asta recuperata via servizio (`getAuctionById`) corrisponde ai dati input.	✓
Verifica stato Asta 'ribasso' valida	`createAuction` ritorna `true` E l'asta recuperata via servizio corrisponde ai dati input (campi decremento validi).	✓
Verifica stato Asta 'inversa' valida	`createAuction` ritorna `true` E l'asta recuperata via servizio corrisponde ai dati input (`sellerId=null`, `buyerId` ok).	✓
Casi di Fallimento (Input Invalidi)		
Input `title = null`	Il metodo `createAuction` restituisce `false` (Violazione `NOT NULL`).	✓
Input `auctionType = null`	Il metodo `createAuction` restituisce `false` (Violazione `NOT NULL`).	✓
Input `startTime = null`	Il metodo `createAuction` restituisce `false` (Violazione `NOT NULL`).	✓

Input (Scenario Specifico)	Risultato Desiderato	Risultato Ottenuto
Input `endTime = null`	Il metodo `createAuction` restituisce `false` (Violazione `NOT NULL`).	✓
Input `startPrice < 0`	Il metodo `createAuction` restituisce `false` (Violazione `CHECK >= 0`).	✓
Input `minPrice < 0`	Il metodo `createAuction` restituisce `false` (Violazione `CHECK >= 0`).	✓
Input `decrementAmount <= 0` (per tipo 'ribasso')	Il metodo `createAuction` restituisce `false` (Violazione `CHECK > 0`).	✓
Input `decrementIntervalMinutes < 1` (per tipo 'ribasso')	Il metodo `createAuction` restituisce `false` (Violazione `CHECK >= 1`).	✓
Input `decrementIntervalMinutes > 720` (per tipo 'ribasso')	Il metodo `createAuction` restituisce `false` (Violazione `CHECK <= 720`).	✓
Input `endTime < startTime + 1 ora`	Il metodo `createAuction` restituisce `false` (Violazione `CHECK time_order`).	✓

Input (Scenario Specific)	Risultato Desiderato	Risultato Ottenuto
Input `minPrice > startPrice`	Il metodo `createAuction` restituisce `false` (Violazione `CHECK min_le_start`).	✓

5.1.1.3 Analisi Classi di Equivalenza – Accettazione offerta (Dettagliata)

Funzionalità: Accettazione Offerta (`AuctionService.acceptBid`)

Questa tabella identifica le Classi di Equivalenza per i parametri di input e le condizioni di stato rilevanti per il metodo `acceptBid`.

Nome CE	Parametro / Stato	Descrizione Valori / Condizione	Effetto Atteso (<code>acceptBid</code>)
AuctionId_Exists	<code>auctionId</code> (Input)	ID di un'asta esistente nel DB	Successo (se altri validi)
AuctionId_NotExists	<code>auctionId</code> (Input)	ID di un'asta non esistente nel DB	Fallimento (<code>false</code>) Logica Service (probabilmente <code>UPDATE</code> non modifica righe o controllo preliminare fallisce)
BidId_Exists	<code>bidId</code> (Input)	ID di un'offerta esistente nel DB	Successo (se altri validi)
BidId_NotExists	<code>bidId</code> (Input)	ID di un'offerta non esistente nel DB	Fallimento (<code>false</code>) Logica Service (<code>UPDATE</code> non modifica righe basandosi sul <code>bidId</code>)
AuctionStatus_Active	Stato Asta (pre-call)	Lo stato dell'asta	Successo (se altri validi, porta a stato 'completata')

Nome CE	Parametro / Stato	Descrizione Valori / Condizione	Effetto Atteso ('acceptBid')
		('auctionId') è 'attiva'	
AuctionStatus_Completed	Stato Asta (pre-call)	Lo stato dell'asta ('auctionId') è 'completata'	Successo ('true') <i>Comportamento attuale: UPDATE potrebbe essere idempotente o non controllare lo stato iniziale.</i>
AuctionStatus_Other	Stato Asta (pre-call)	Lo stato dell'asta ('auctionId') è 'scaduta' o altro	Fallimento ('false') <i>Previsto: La logica dovrebbe impedire l'accettazione (Test Case Mancante)</i>
BidAuctionRel_Correct	Relazione Offerta-Asta	L'offerta ('bidId') appartiene all'asta ('auctionId')	Successo (se altri validi)
BidAuctionRel_Incorrect	Relazione Offerta-Asta	L'offerta ('bidId') NON appartiene all'asta ('auctionId')	Fallimento ('false') <i>Logica Service ('UPDATE' non modifica righe a causa della condizione sulla relazione)</i>

5.1.1.4 Piano di Test: Servizio Aste - Accettazione Offerta

Test ID:

AS_AB_T01

Test Name:

Test Funzionalità Accettazione Offerta (`acceptBid`)

Test Description:

Verificare la corretta gestione dell'accettazione di offerte tramite il metodo `acceptBid`, includendo casi validi e la gestione di input/stati invalidi (asta/offerta esistente, relazione corretta, stato asta).

Dettaglio Test Case

Input (Scenario Specifico / Metodo Test)	Risultato Desiderato	Risultato Ottenuto
Casi di Successo		
<code>testAcceptBid_Success</code>	Il metodo `acceptBid` restituisce `true`. L'asta viene aggiornata a 'completata' con il vincitore e il prezzo corretti.	✓

Input (Scenario Specifico / Metodo Test)	Risultato Desiderato	Risultato Ottenuto
Casi di Fallimento / Limite		
<code>testAcceptBid_Failure_AuctionNotFound</code>	Il metodo `acceptBid` restituisce `false`.	✓
<code>testAcceptBid_Failure_BidNotFound</code>	Il metodo `acceptBid` restituisce `false`. Lo stato dell'asta originale rimane invariato.	✓
<code>testAcceptBid_Failure_BidForWrongAuction</code>	Il metodo `acceptBid` restituisce `false`. Lo stato dell'asta target rimane invariato.	✓
<code>testAcceptBid_EdgeCase_AuctionAlreadyCompleted</code>	Il primo `acceptBid` restituisce	✓

Input (Scenario Specifico / Metodo Test)	Risultato Desiderato	Risultato Ottenuto
	<p>`true`. Il secondo `acceptBid` restituisce ancora `true`. Lo stato finale è 'completata'.</p>	
<p>(Scenario Mancante): Tentativo `acceptBid` su Asta 'scaduta'</p>	<p>Il metodo `acceptBid` dovrebbe restituire `false`.</p>	

5.1.1.5 Analisi Classi di Equivalenza (Dettagliata) – Creazione offerta

Funzionalità: Creazione Offerta (`AuctionService.createBid`)

Questa tabella identifica le Classi di Equivalenza per i parametri di input e le condizioni di stato rilevanti per il metodo `createBid`.

Nome CE	Parametro / Stato	Descrizione Valori / Condizione	Effetto Atteso (<code>createBid</code>)
AuctionId_ValidActive	<code>auctionId</code> (Input) & Stato Asta	ID di un'asta esistente E con stato 'attiva'	Successo (se altri validi)
AuctionId_Invalid_Not Found	<code>auctionId</code> (Input)	ID di un'asta non esistente nel DB	Fallimento ('false') <i>Validazione Service (asta non trovata)</i>
AuctionId_Invalid_Not Active	Stato Asta (<code>status</code>)	Stato dell'asta con <code>'auctionId'</code> esistente è 'scaduta' o 'completata'	Fallimento ('false') <i>Validazione Service (stato non attivo)</i>
BidPrice_Valid_Positive	<code>bidPrice</code> (Input)	Valore numerico <code>> 0</code>	Successo (se altri validi)

Nome CE	Parametro / Stato	Descrizione Valori / Condizione	Effetto Atteso ('createBid')
BidPrice_Invalid_NonPositive	<code>bidPrice</code> (Input)	Valore numerico <code><= 0</code>	Fallimento ('false') <i>CHECK DB / Validazione Service</i>
BidPrice_Valid_Ribasso	<code>bidPrice</code> vs Asta	(Tipo 'ribasso') <code>bidPrice <= auctions.current_price</code>	Successo (se altri validi)
BidPrice_Invalid_Ribasso	<code>bidPrice</code> vs Asta	(Tipo 'ribasso') <code>bidPrice > auctions.current_price</code>	Fallimento ('false') <i>Trigger / Validazione Service</i>
BidPrice_Valid_Inversa	<code>bidPrice</code> vs Asta	(Tipo 'inversa') <code>bidPrice < auctions.current_price</code>	Successo (se altri validi)
BidPrice_Invalid_Inversa	<code>bidPrice</code> vs Asta	(Tipo 'inversa') <code>bidPrice >= auctions.current_price</code>	Fallimento ('false') <i>Trigger / Validazione Service</i>
UserId_Valid	<code>userId</code> (Input)	Stringa non nulla e non vuota	Successo (se altri validi)

Nome CE	Parametro / Stato	Descrizione Valori / Condizione	Effetto Atteso ('createBid')
UserId_Invalid_Null	<code>userId</code> (Input)	<code>null</code>	Fallimento ('false') NOT NULL DB / Validazione Service
UserId_Invalid_Empty	<code>userId</code> (Input)	Stringa vuota ("")	Fallimento ('false') CHECK DB / Validazione Service
Cooldown_Respected	Tempo dall'ultima offerta utente	Tempo trascorso >= Tempo Cooldown (es. 300s) O Prima offerta	Successo (se altri validi)
Cooldown_Violated	Tempo dall'ultima offerta utente	Tempo trascorso < Tempo Cooldown (es. 300s)	Fallimento ('false') Validazione Service (Cooldown)
User_NotWinner_Inversa	<code>userId</code> vs <code>auctions.winner_id</code>	(Tipo 'inversa') <code>userId != auctions.winner_id</code> OR <code>auctions.winner_id</code> è NULL	Successo (se altri validi)
User_IsWinner_Inversa	<code>userId</code> vs <code>auctions.winner_id</code>	(Tipo 'inversa') <code>userId ==</code>	Fallimento ('false') Validazione Service

Nome CE	Parametro / Stato	Descrizione Valori / Condizione	Effetto Atteso (<code>createBid</code>)
		<code>auctions.winner_id</code>	<i>(Utente già vincitore)</i>

5.1.1.6 Piano di Test: Servizio Aste - Creazione Offerta

Test ID:

AS_CB_T01

Test Name:

Test Funzionalità Creazione Offerta (`createBid`)

Test Description:

Verificare la corretta gestione della creazione di offerte tramite il metodo `createBid`, includendo casi validi e la gestione di input/stati invalidi secondo le regole di business e i vincoli di integrità (asta esistente/attiva, prezzo valido per tipo, username valido, cooldown, blocco vincitore inversa).

Dettaglio Test Case

Input (Scenario Specifico)	Risultato Desiderato	Risultato Ottenuto
Casi di Successo		
Offerta valida (prezzo > 0, utente valido) su asta 'silenziosa' attiva	Il metodo `createBid` restituisce `true`.	✓
Verifica stato dopo offerta valida	`createBid` ritorna `true` E `getBidsByAuctionId` restituisce l'offerta con i dati corretti.	✓
Casi di Fallimento (Input/Stato Invalidi)		

Input (Scenario Specifico)	Risultato Desiderato	Risultato Ottenuto
Input `auctionId` inesistente	Il metodo `createBid` restituisce `false`.	✓
Input `bidPrice = 0`	Il metodo `createBid` restituisce `false`.	✓
Input `bidPrice < 0`	Il metodo `createBid` restituisce `false`.	✓
Input `username = null`	Il metodo `createBid` restituisce `false`.	✓
Input `username = ""` (stringa vuota)	Il metodo `createBid` restituisce `false`.	✓
Offerta su asta non attiva (es. 'scaduta')	Il metodo `createBid` restituisce `false`.	✓
Offerta ripetuta prima della fine del cooldown (300s)	Il metodo `createBid` restituisce `false`.	✓
Offerta su asta 'inversa' da utente già vincitore	Il metodo `createBid` restituisce `false`.	✓

Input (Scenario Specific)	Risultato Desiderato	Risultato Ottenuto
Offerta con `prezzo > prezzo corrente` su asta 'ribasso'	Il metodo `createBid` restituisce `false`.	✓
Offerta con `prezzo >= prezzo corrente` su asta 'inversa'	Il metodo `createBid` restituisce `false`.	✓

5.1.1.7 Analisi Classi di Equivalenza – Recupero aste utente

Funzionalità: Recupero Aste Utente (`AuctionService.getUserAuctions`)

Questa tabella identifica le Classi di Equivalenza per i parametri di input del metodo `getUserAuctions`. Considera sia i parametri espliciti che le condizioni implicite dei dati nel database.

Nome CE	Parametro / Stato	Descrizione Valori / Condizione	Effetto Atteso (<code>getUserAuctions</code>)
UserRole_Seller_Valid	<code>userId</code> , <code>userRole</code>	<code>userId</code> di un venditore esistente, <code>userRole="venditore"</code>	Successo (restituisce aste 'ribasso'/silenziosa' di quel venditore)
UserRole_Buyer_Valid	<code>userId</code> , <code>userRole</code>	<code>userId</code> di un compratore esistente, <code>userRole="compratore"</code>	Successo (restituisce aste 'inversa' di quel compratore)
UserRole_Mismatch	<code>userId</code> , <code>userRole</code>	<code>userId</code> di venditore con <code>'userRole="compratore"</code> (o viceversa)	Lista Vuota
UserId_Invalid	<code>userId</code>	ID utente non esistente nel DB	Lista Vuota

Nome CE	Parametro / Stato	Descrizione Valori / Condizione	Effetto Atteso (`getUserAuctions`)
Keyword_Valid_Match	keyword	Stringa presente (case-insensitive) in titolo, descrizione o categoria di almeno un'asta	Successo (restituisce solo aste corrispondenti)
Keyword_Valid_NoMatch	keyword	Stringa non presente in nessuna asta recuperata inizialmente	Lista Vuota
Keyword_Valid_NullEmpty	keyword	null o stringa vuota	Successo (nessun filtro per keyword applicato)
Category_Valid_Match	category	Stringa corrispondente (case-insensitive) a una categoria esistente	Successo (restituisce solo aste di quella categoria)
Category_Valid_NoMatch	category	Stringa non corrispondente a nessuna categoria delle aste recuperate	Lista Vuota

Nome CE	Parametro / Stato	Descrizione Valori / Condizione	Effetto Atteso (`getUserAuctions`)
Category_Valid_NullEmpty	<code>category</code>	<code>null</code> o stringa vuota	Successo (nessun filtro per categoria applicato)
FilterType_Valid_Match	<code>filterType</code>	Stringa contenente (case-insensitive) un tipo d'asta esistente tra quelle recuperate	Successo (restituisce solo aste di quel tipo/i)
FilterType_Valid_NoMatch	<code>filterType</code>	Stringa non contenente nessun tipo d'asta tra quelle recuperate	Lista Vuota
FilterType_Valid_NullEmpty	<code>filterType</code>	<code>null</code> o stringa vuota	Successo (nessun filtro per tipo applicato)
Paging_Valid_First	<code>page,</code> <code>pageSize</code>	<code>page = 1, pageSize > 0</code>	Successo (restituisce i primi `pageSize` risultati)

Nome CE	Parametro / Stato	Descrizione Valori / Condizione	Effetto Atteso (<code>getUserAuctions`</code>)
Paging_Valid_Middle	<code>page,</code> <code>pageSize</code>	<code>page > 1</code> , tale che <code>`(page-1)*pageSize < totalResults`</code>	Successo (restituisce la porzione corretta di risultati)
Paging_Invalid_Beyond	<code>page,</code> <code>pageSize</code>	<code>page</code> tale che <code>`(page-1)*pageSize >= totalResults`</code>	Lista Vuota
Paging_Invalid_PageSize	<code>page,</code> <code>pageSize</code>	<code>page <= 0</code> o <code>pageSize <= 0</code>	Successo (ma usa valori di default <code>`page=1`</code> , <code>`pageSize=10`</code> nel service)

5.1.1.8 Piano di Test: Servizio Aste - Recupero Aste Utente

Test ID:

AS_GA_T01

Test Name:

Test Funzionalità Recupero Aste Utente (`getUserAuctions`)

Test Description:

Verificare la corretta gestione del recupero e filtraggio delle aste per un utente specifico tramite il metodo `getUserAuctions`, considerando ruolo, keyword, categoria, tipo asta e paginazione.

Dettaglio Test Case

Input (Scenario Specifico)	Risultato Desiderato	Risultato Ottenuto
Casi di Successo (Risultati Attesi)		
Filtro per Ruolo Venditore (`SELLER_ID_1`)	Restituisce la lista corretta (9) di aste non-inverse per quel venditore.	✓
Filtro per Ruolo Compratore (`BUYER_ID_1`)	Restituisce la lista corretta (2) di aste inverse per quel compratore.	✓
Filtro per Keyword Unica (case-insensitive)	Restituisce la singola asta corrispondente.	✓
Filtro per Keyword Comune (case-insensitive)	Restituisce tutte le aste (9) del venditore che contengono la keyword comune.	✓

Input (Scenario Specifico)	Risultato Desiderato	Risultato Ottenuto
Filtro per Categoria (case-insensitive)	Restituisce le aste corrette (3) per quella categoria e venditore.	✓
Filtro per Tipo Asta (es. "ribasso")	Restituisce le aste corrette (2) per quel tipo e venditore.	✓
Filtro Combinato (Keyword, Categoria, Tipo)	Restituisce le aste corrette (2) che soddisfano tutti i filtri.	✓
Paginazione (Pagina 1, 2 con risultati)	Restituisce il numero corretto di aste per pagina (5, 4) senza sovrapposizioni.	✓
<i>Casi con Risultato Atteso: Lista Vuota</i>		
Paginazione (Pagina 3, oltre i risultati)	Restituisce una lista vuota (0 risultati).	✓
Caso Nessun Risultato (Filtri non corrispondenti)	Restituisce una lista vuota.	✓

5.2 Strategia di Test e dipendenze utilizzate per il piano di test

Questo documento delinea l'approccio strategico adottato per la verifica funzionale dei metodi in AuctionService (come createAuction, createBid, acceptBid, getUserAuctions) e descrive le principali tecnologie e librerie impiegate nel processo di testing. L'obiettivo primario è garantire l'affidabilità, la correttezza e la robustezza del servizio rispetto ai requisiti funzionali e alle regole di business definite.

5.2.1 Strategia di Test Adottata

L'approccio principale scelto è quello dell'**Integration Testing**. I test verificano l'interazione tra il codice del servizio (AuctionService) e il livello di persistenza, assicurando che le operazioni CRUD (Create, Read, Update, Delete) e la logica di business che coinvolgono i dati funzionino come previsto.

Per la progettazione dei casi di test specifici, è stata utilizzata una strategia **Black-Box**, focalizzandosi sugli input e sugli output attesi dei metodi del servizio senza necessariamente conoscere i dettagli implementativi interni. In particolare, è stata applicata la seguente tecnica:

- **Analisi delle Classi di Equivalenza:** ogni parametro di input e condizione di stato rilevante (es. tipo di asta, stato dell'asta, validità dei prezzi, relazione tra entità) è stato suddiviso in classi di valori dove vengano trattati dal sistema in modo simile (classi valide e invalide). Questo permette di ridurre il numero di test necessari, selezionando almeno un rappresentante per ogni classe per ottenere una buona copertura sistematica degli input.

L'ambiente di test è stato configurato per garantire **isolamento** e **ripetibilità**: Viene utilizzato un database **in-memory H2** che viene creato, popolato con dati di test specifici (@BeforeEach tramite DBQueryHelper) e distrutto/ricreato tra i test (gestito da BaseIntegrationTest), assicurando che i test non interferiscano tra loro e non dipendano da uno stato preesistente del database. La classe BaseIntegrationTest fornisce un setup comune per la gestione del ciclo di vita del database e altre configurazioni condivise tra le classi di test di integrazione.

5.2.3 Dipendenze Chiave Utilizzate nel Testing

- **JUnit 5 (Jupiter):**
Uso: Framework principale per la scrittura e l'esecuzione dei test in Java. Fornisce le annotazioni (@Test, @BeforeAll, @BeforeEach, @DisplayName, etc.) per definire la struttura dei test, i metodi di setup/teardown e le asserzioni (Assertions.*) per verificare i risultati attesi.
Scelta: Standard de-facto per il testing in Java, moderno, flessibile e ben integrato con build tools come Maven.

- **Maven (con Surefire Plugin):**

Uso: Build tool utilizzato per gestire il ciclo di vita del progetto, incluse la compilazione, la gestione delle dipendenze e l'esecuzione dei test. Il plugin maven-surefire-plugin è responsabile specifico dell'esecuzione dei test JUnit durante la fase test del build Maven.

Scelta: Strumento di build ampiamente diffuso nell'ecosistema Java, facilita la gestione delle dipendenze e l'automazione del processo di build e test.

- **H2 Database Engine:**

Uso: Database relazionale SQL, scritto in Java, che può operare in modalità in-memory. Viene utilizzato per creare un database temporaneo e isolato per ogni esecuzione dei test di integrazione.

Scelta: Ideale per i test perché è veloce, non richiede installazione separata, si integra facilmente in un'applicazione Java e permette una pulizia completa dello stato tra i test, garantendo l'isolamento. Permette di testare la logica SQL e le interazioni JDBC senza dipendere da un database esterno.

- **Utility Custom nel package com.hello.utils:**

Uso: Set di classi di supporto sviluppate ad-hoc per facilitare i test:

- DBConnection: Fornisce un metodo statico getConnection() che, in base alla proprietà di sistema 'env', restituisce una connessione al DB. Se 'env' è 'test', delega a TestDBConfig per ottenere una connessione H2; altrimenti, si connette al DB (MySQL) originario utilizzato dall'app, non quello simulato in memoria.
- TestDBConfig: Definisce i parametri di connessione (URL JDBC jdbc:h2:mem:testdb;DB_CLOSE_DELAY=-1, utente, password) per il database H2 in-memory utilizzato per i test e fornisce il metodo statico getConnection() per ottenere la connessione a questo DB.
- BaseIntegrationTest: Classe base astratta per i test di integrazione. Utilizza metodi JUnit @BeforeAll e @AfterAll per invocare DBQueryHelper al fine di creare/distuggere lo schema del database H2 prima e dopo l'esecuzione dei test in una classe che la estende.
- DBQueryHelper: Classe helper con metodi statici per l'interazione diretta con il DB H2 via JDBC (usando TestDBConfig). Include metodi per la gestione dello schema strutturale (createTables, dropTables), l'inserimento/aggiornamento di dati di test (createTestAuction, createTestBid, setAuctionStatus) e il recupero di dati per le verifiche (getAuctionFromDbById).
- AuctionTestDataFactory: Utilizza la libreria JavaFaker per generare dati realistici e validi per oggetti Auction (createRealisticAuction). Include anche metodi specifici per creare e persistere set di dati necessari a scenari di test particolari (es. setupAuctionsForGetUserAuctionsTest).
- AuctionTestAssertions: Fornisce metodi di asserzione statici e personalizzati (assertAuctionFieldsEqual) per confrontare in modo dettagliato e robusto oggetti Auction all'interno dei test, gestendo tipi specifici come BigDecimal e Timestamp.

Scelta: Create per ridurre la duplicazione di codice nei test, migliorare la leggibilità e l'affidabilità, centralizzando le operazioni ripetitive di setup, teardown, interazione DB, creazione dati e asserzioni specifiche del dominio.

In conclusione, la strategia adottata combina efficacemente l'***integration testing** (per validare l'interazione servizio-DB), l'***analisi delle classi di equivalenza** (per una copertura

sistematica degli input), e l'automazione fornita da **JUnit** e **Maven**. L'uso mirato del database **H2 in-memory** garantisce un ambiente di test isolato, veloce e ripetibile. Le **utility custom** specifiche del progetto ottimizzano ulteriormente il processo, rendendo i test più leggibili ed efficienti nel verificare la correttezza dei metodi `createAuction`, `createBid`, `acceptBid` e `getUserAuctions` di `AuctionService` rispetto ai requisiti definiti.