UniRent Hub

Test Plan
UniRentHub

Riferimento	C11_TP_V_2.0
Versione	2.0
Data	23/01/2024
Destinatario	Prof.ssa Filomena Ferrucci, Prof. Fabio Palomba
Presentato da	C11 Fries Tech Team
Approvato da	Rocco Iuliano, Simone Della Porta

Team Members

Nome	Cognome	Ruolo	Acronimo	Contatto
Rocco	Iuliano	PM	RI	r.iuliano13@studenti.unisa.it
Simone	Della Porta	PM	SDP	s.dellaporta6@studenti.unisa.it
Antonio	Albanese	TM	AA	a.albanese22@studenti.unisa.it
Francesco Pio	Contaldo	TM	FPC	f.contaldo4@studenti.unisa.it
Cristyan	Esposito	TM	CE	c.esposito175@studenti.unisa.it
Iliano	Fasolino	TM	IF	i.fasolino3@studenti.unisa.it
Marco	Greco	TM	MG	m.greco65@studenti.unisa.it
Giuseppe Pio	Sorrentino	TM	GPS	g.sorrentino101@studenti.unisa.it

Revision History

Data	Versione	Descrizione	Autori
06/12/2023	0.1	Scrittura paragrafi 1, 2, 3, 4, 5, 6	RI, SDP
07/12/2023	0.2	Scrittura paragrafi 7, 8, 9	RI, SDP
10/12/2023	0.3	Scrittura System Test Plan	Tutto il team
12/12/2023	0.4	Check Test Plan	IF, CE
13/12/2023	1.0	Revisione Test Plan	IF, CE
07/01/2024	1.1	Scrittura Unit Test Plan	FPC, CE, IF
08/01/2024	1.2	Scrittura Unit Test Plan	AA, MG, GPS
17/01/2024	2.0	Revisione Test Plan	IF, CE

Sommario

1: Introduzione

UniRentHub si propone di diventare leader nel settore immobiliare in locazione facilitando e velocizzando la ricerca di case per studenti, evitando la ricerca estenuante su diversi siti e social media, rispondendo così all'elevata domanda di alloggio. Il progetto si propone di facilitare la ricerca di coinquilini consentendo così la ripartizione del prezzo d'affitto, aiutando gli utenti sull'aspetto economico. Infine, mira ad offrire un servizio che promuova la sostenibilità ambientale sia attraverso la piattaforma stessa, sia attraverso le opzioni di alloggio disponibili in modo da fornire agli utenti un beneficio economico a lungo termine.

Il presente documento ha l'obiettivo di descrivere le attività di testing pianificate per la piattaforma UniRentHub in modo da garantire il suo corretto funzionamento.

2: System overview

Il software UniRentHub è basato su un'architettura three-tier e la suddivisione del sistema rispetta il modello MVC (Model View Control). Il lato front-end della piattaforma sarà sviluppato tramite HTML, CSS, JS e Python mentre il lato back-end del sito sarà sviluppato in Python e verrà utilizzato Flask come server web in modo da poter gestire le request e le response HTTP. Infine, per la gestione dei dati persistenti, si utilizzerà un database MySQL con il quale sarà possibile interagire grazie a delle librerie per Python e Flask.

3: Relationship with others document

Per realizzare correttamente i test case, si fa riferimento ai seguenti documenti:

- **RAD:** in quanto i test case sono definiti sulla base dei requisiti funzionali, requisiti non funzionali e dei casi d'uso;
- **SDD:** in quanto i test case devono rispettare la suddivisione in sottosistemi descritta dal suddetto documento.

Di seguito i riferimenti ai documenti utili:

- Requirements Analysis Document
- System Design Document

4: Features to be tested

La fase di testing è stata prioritizzata in base alle priorità definite nel RAD per i requisiti funzionali; quindi, verranno testate le funzionalità cardine del sistema software. Inoltre, verranno escluse dalla fase di testing le funzionalità non previste per la prima release del progetto. Quindi le funzionalità che verranno testate sono:

- Gestione annunci
  - Creare annuncio proprietario
- Gestione affitto
  - Prenotazione visita
  - Affitto
  - Recensione
  - Segnalare locatore
- Gestione utente
  - Modifica dati personali

5: Pass/Fail criteria

I test avranno l'obiettivo di individuare delle failure all'interno del sistema software in modo da poter applicare tempestivamente un'attività di correzione.
Il test si considera superato se l'output corrisponde all'output atteso presente nell'oracolo, invece, il test si considera fallito se l'output non corrisponde all'output atteso presente nell'oracolo.

6: Test approach

L'approccio che utilizzeremo per testare il sistema è l'approccio **Bottom-up**. Questa tecnica consiste nel testare dapprima singolarmente le componenti che non hanno bisogno di altre componenti per essere testate (test di unità), poi si procede effettuando il testing sulle componenti del layer superiore includendo le componenti del layer inferiore che ciascuna di essa utilizza (test di integrazione), si ripete ciò finché non vengono testate tutte le componenti (test di sistema). Nell'applicare quest'approccio, si partirà dalle classi DAO poiché il database lo assumiamo corretto. Il budget del progetto prevede che ogni membro del team dovrà effettuare il testing di unità di esattamente un metodo di una classe sviluppata. Quindi, a partire dalle classi DAO, si applicherà un testing di unità tramite la tecnica di **Category Partition** che ci consente di decomporre lo spazio di input in categorie per poi partizionare queste categorie in classi di equivalenza chiamate scelte, ottenendo una Test Suite che include tutti i Test Case per il DAO in questione. Il passo successivo è fare testing di integrazione tra Servlet e DAO, ma si bypasserà questo tipo di testing perché il Category Partition per il testing di sistema copre tutto ciò che ci si aspetta da una Servlet perché in caso di failure riusciamo ad individuare facilmente il fault. Inoltre, la pagina HTML invoca solo la Servlet e i DAO sono già stati testati tramite test di unità. Quindi si passerà direttamente al testing di sistema applicato sulle funzionalità cardine del sistema sempre tramite la tecnica di **Category Partition**. In questo caso il budget del progetto prevede che ogni membro del team dovrà effettuare testing di sistema di esattamente una funzionalità del progetto.
Testing materials:

- **Pytest** per il testing di unità;
- **Selenium** per poter registrare le azioni effettuare dall'utente sulla piattaforma in modo da poter ottenere il metodo di test che consente di testare la funzionalità registrata;
- **Web server Flask** per poter gestire le request e le response HTTP ed eseguire la piattaforma UniRentHub;
- **DBMS** di MySQL per poter interagire con il database;
- Web browser per poter effettuare le richieste al server.

7: Suspension and reinstatement criteria

7.1: Suspension criteria

I test possono essere sospesi solo se viene mostrato un messaggio d'errore dovuto alla definizione del test stesso altrimenti i test non possono essere sospesi finché la loro esecuzione non termina.

7.2: Reinstatement criteria

I test sospesi verranno ripresi andando a rieseguire l'intero test dopo aver corretto il fault.

8: Test deliverables

I documenti di test che verranno prodotti saranno:

- Test Plan;
- Test Case Specification;
- Test Execution Report;
- Test Incident Report;
- Test Summary Report.

9: Testing schedule

Le attività di testing verranno condotte come definite nei capitoli precedenti, quindi subito dopo la fase di system design.

Lo sviluppo dei test avverrà subito dopo aver ultimato la fase di sviluppo. Di conseguenza l'esecuzione dei test avverrà dopo la loro implementazione e verranno prodotti i vari report di test. Ulteriori informazioni sono indicate nei documenti di management.

10: System Test Case

10.1: Gestione Utente

10.1.1: Modifica dati personali

Parametri:	password, università, nome, cognome, numerocarta, mese, anno, cvv
Oggetti dell'ambiente:	Database
Categorie:	**Scelte**
Modifica Password	P1: Password non valida P2: Password valida
Modifica Università	U1: Università valida
Modifica numerocarta	NC1: Numero carta non valido NC2: Numero carta valido
Modifica mese	M1: mese non valido M2: mese valido
Modifica anno	A1: anno valido
CVV	CV1: cvv non valido CV2: cvv valido

Parametro: Password	
Nome categoria	**Scelte per la categoria**
Password	1. Lunghezza < 8 OR > 64 OR Lettera maiuscola < 1 OR Numeri < 1 OR Caratteri speciali < 1 [Errore] 2. Lunghezza >8 OR < 64 OR Lettera maiuscola > = 1 OR Numeri > = 1 OR Caratteri speciali > = 1 [corretto]
Parametro: Università	
Nome categoria	**Scelte per la categoria**
Università	Università [PROPERTY_U_OK]

Parametro: Numero Carta	
Nome categoria	**Scelte per la categoria**
Numero Carta	Lunghezza != 16 [Errore] Lunghezza = 16 [Corretto]
Parametro: Mese	
Nome categoria	**Scelte per la categoria**
Mese	1. Mese/Anno < DataCorrente [Errore] 2. Mese/Anno > DataCorrente [Corretto]
Parametro: Cvv	
Nome categoria	**Scelte per la categoria**
Cvv	1. Lunghezza != 3[Errore] 2. Lunghezza = 3 [Corretto]

Test-Case ID	**Test Frame**	**Esito**
TC_1.1_1	P1	Errore: Lunghezza non valida
TC_1.1_2	P2-U1-NC1	Errore: Lunghezza non valida
TC_1.1_3	P2-U1-NC2-A1-M1	Errore:Data non valida
TC_1.1_4	P2-U1- -NC2-A1-M2-CV1	Errore: Lunghezza non valida
TC_1.1_5	P2-U1- -NC2-A1-M2-CV2	Modifica avvenuta

10.2: Gestione Annunci

10.2.1: Pubblicazione annuncio di affitto

Parametri:	Titolo, citta, provincia, cap, civico, indirizzo, descrizione, Numero Bagni, Numero camere da letto, Pannelli solari, Pannelli fotovoltaici, Tipo alloggio, arredamento, Classe energetica, Numero ospiti, Metriquadri, Prezzo, Servizi compresi, immagini, Periodo minimo
Oggetti dell'ambiente	Database
Categorie:	**Scelte**
Titolo	T1: titolo non valido T2: titolo valido

Citta	C1: citta non valida C2:citta valida
Provincia	P1: provincia non valida P2: provincia valida
Cap	CA1: cap non valido CA2: cap valido
Inserimento Indirizzo	I1: indirizzo non valido I2: indirizzo valido
Civico	CIV1: civico non valido CIV2: civico valido
Inserimento descrizione	D1: descrizione non valida D2: descrizione valida
Inserimento Numero Bagni	NB1: numero bagni non valido NB2: numero bagni valido
Inserimento Numero Camere Letto	NC1: numero camere non valido NC2: numero camere valido
Inserire la presenza di pannelli solari	NPS1: presenza pannelli solari valida
Inserire la presenza di pannelli fotovoltaici	NPF1: presenza pannelli fotovoltaici valida
Inserimento Tipo Alloggio	TP1: Tipo alloggio valido
Inserimento arredamento	A1: presenza arredamento valida
Inserimento Classe Energetica	CE1: presenza classe energetica valida
Inserimento Numero Ospiti	NO1: numero ospiti non valido NO2: numero ospiti valido
Inserisci i Metri Quadri	MQ1: numero metri quadri non valido MQ2: numero metri quadri valido
Prezzo	P1: prezzo non valido P2: prezzo valido
Periodo minimo	PM1 : periodo valido
Servizi	S1: servizio valido
Immagini	IM1: immagini non valide IM2: immagini valide

Parametro: Titolo	
Nome categoria	**Scelte per la categoria**

Titolo	Lunghezza >30 OR Lunghezza<5 [errore] Lunghezza <30 AND Lunghezza >=5 [corretto]
Parametro: Indirizzo	
Nome categoria	**Scelte per la categoria**
Indirizzo	Lunghezza<=5 [errore] Lunghezza>5 [corretto]
Parametro: Citta	
Nome categoria	**Scelte per la categoria**
Citta	Lunghezza <0[errore] Lunghezza>0[corretto]
Parametro: Provincia	
Nome categoria	**Scelte per la categoria**
Provincia	Lunghezza<2[errore] Lunghezza = 2[corretto]
Parametro: CAP	
Nome categoria	**Scelte per la categoria**
CAP	Lunghezza !=5 [errore] Lunghezza == 5 [corretto]
Parametro: Civico	
Nome categoria	**Scelte categoria**
Civico	Valore =0 [errore] Valore != 0 [corretto]
Parametro: Descrizione	
Nome categoria	**Scelte per la categoria**
Descrizione	Lunghezza <30 [errore] Lunghezza >30
Parametro: Numero bagni	
Nome categoria	**Scelte per la categoria**
Numero bagni	Valore<1 [errore] Valore>=1 [corretto]
Parametro: Numero camere da letto	

Nome categoria	**Scelte per la categoria**
Numero camere da letto	Valore<1 Valore>=1 [corretto]
Parametro: Numero ospiti	
Nome categoria	**Scelte per la categoria**
Numero ospiti	Valore<1 [errore] Valore>=1 [corretto]
Parametro: Metri Quadri	
Nome categoria	**Scelte per la categoria**
Metri Quadri	Valore<1 [errore] Valore>=1 [corretto]
Parametro: Prezzo	
Nome categoria	**Scelte per la categoria**
Prezzo	Valore<1 [errore] Valore>=1 [corretto]
Parametro: Periodo Minimo	
Nome categoria	**Scelte per la categoria**
Periodo minimo	Valore<0 (in mesi) [errore] Valore >= 0 [valido]
Parametro: Immagini	
Nome categoria	
Immagini	Numero immagini != 3[errore] Numero immagini = 3 [corretto]

Test-Case ID	**Test Frame**	**Esito**
TC_2.1_1	T1	Errore: lunghezza non valida
TC_2.1_2	T2-C1	Errore: Citta non valida
TC_2.1_3	T2-C2-P1	Errore: Provincia non valida
TC_2.1_4	T2-C2-P2-CA1	Errore: Cap non valido
TC_2.1_5	T2-C2-P2-CA2-I1	Errore: indirizzo non valido

TC_2.1_6	T2-C2-P2-CA2-I2-CIV1	Errore: Numero civico non valido
TC_2.1_7	T2-C2-P2-CA2-I2-CIV2-D1	Errore: la descrizione è troppo breve
TC_2.1_8	T2-C2-P2-CA2-I2-CIV2-D2-NB1	Errore: numero bagni non valido
TC_2.1_9	T2-C2-P2-CA2-I2-CIV2-D2-NB2-NC1	Errore: numero camere non valido
TC_2.1_10	T2-C2-P2-CA2-I2-CIV2-D2-NB2-NC2-NPS1-NPF1-TP1-A1-CE1-NO1	Errore: numero ospiti non valido
TC_2.1_11	T2-C2-P2-CA2-I2-CIV2-D2-NB2-NC2-NPS1-NPF1-TP1-A1-CE1-NO2-MQ1	Errore: numero metri quadri non valido
TC_2.1_12	T2-C2-P2-CA2-I2-CIV2-D2-NB2-NC2-NPS1-NPF1-TP1-A1-CE1-NO2-MQ2-P1	Errore: il prezzo non è valido
TC_2.1_13	T2-C2-P2-CA2-I2-CIV2-D2-NB2-NC2-NPS1-NPF1-TP1-A1-CE1-NO2-MQ2-P2-PM1-S1-IM1	Errore: il numero di immagini non è valido
TC_2.1_14	T2-C2-P2-CA2-I2-CIV2-D2-NB2-NC2-NPS1-NPF1-TP1-A1-CE1-NO2-MQ2-P2-PM1-S1-IM2	Corretto: l'inserimento va a buon fine

10.3: Gestione Affitto

10.3.1: Prenotazione visita alloggio

Parametri:	Data e ora visita
Oggetti dell'ambiente	Database
Categorie:	**Scelte**
Selezione data visita	DV1: data e ora selezionata valida

Parametro: Data Visita	
Nome categoria	**Scelte per la categoria**
Seleziona Data Visita [DV]	Data selezionata[PROPERTY_DV_OK]

Test-Case ID	**Test Frame**	**Esito**
TC_3.1_1	DV1	OK: Data visita confermata con successo

10.3.2: Affitto

Parametri:	Numero Carta, Data Scadenza, CVV, Data Check-in, Data Check-out, Inserimento Data Appuntamento con Locatore
Oggetti dell'ambiente:	Database, Sessione
Categorie:	**Scelte**
Inserimento Numero Carta (PAN)	NC1: numero carta valido NC2: numero carta NON valido
Inserimento Data Scadenza	DS1: data scadenza valida DS2: data scadenza NON valida
Inserimento CVV	IC1: cvv valido IC2: cvv NON valido
Inserimento Data Check-in	DCI1: data check-in valida DCI2: data check-in NON valida
Inserimento Data Check-out	DCO1: data check-out valida DCO2: data check-out NON valida
Periodo	P1: Periodo valido P2: Periodo non valido

Parametro: Numero carta	
Nome categoria	**Scelte per la categoria**
Numero Carta	1. Lunghezza > 16 OR Lunghezza < 16 [errore] 2. Lunghezza = 16 [corretto]
Parametro: Data Scadenza	
Nome categoria	**Scelte per la categoria**
Data Scadenza	1. DataCorrente < Data Scadenza [corretto] 2. DataCorrente >= DataScadenza [errore]
Parametro: CVV	
Nome categoria	**Scelte per la categoria**
CVV	1. Lunghezza < 3 OR Lunghezza > 3 [errore] 2. Lunghezza = 3 [corretto]
Parametro: Data Check-In	
Nome categoria	**Scelte per la categoria**
Data Check-In	1. DataCorrente > Data Check-In [errore] 2. DataCorrente <= Data Check-In [corretto]
Parametro: Data Check-Out	
Nome categoria	**Scelte per la categoria**
Data Check-Out	1. Data Check-in >= Data Check-Out [errore] 2. Data Check-in < Data Check-Out [corretto]
Parametro: Periodo	
Nome categoria	**Scelte per la categoria**
Periodo	1. Durata < periodo minimo[errore] 2. Durata >periodo minimo[corretto]

Test-Case ID	**Test Frame**	**Esito**
TC_3.2_1	NC2	Errore: Numero carta non valido
TC_3.2_2	NC1-DS2	Errore: Data scadenza non valida
TC_3.2_3	NC1-DS1- IC2	Errore: CVV non valido

TC_3.2_4	NC1-DS1-IC1-DCI2	Errore: Data Check-in non valida
TC_3.2_5	NC1-DS1-IC1-DCI1-DCO2	Errore: Data Check-out non valida
TC_3.2_6	NC1-DS1-IC1-DCI1-DCO1-P1	Errore: Data Appuntamento con Locatore non valida
TC_3.2_7	NC1-DS1-IC1-DCI1-DCO1-P1	Corretto: Affitto completato con successo

10.3.3: Recensione

Parametri:	Titolo, Descrizione, Periodo, Voto
Oggetti dell'ambiente:	Database
Categorie:	**Scelte**
Titolo	T1: titolo non valido T2: titolo valido
Descrizione	D1: descrizione non valida D2: descrizione valida
Periodo	P1: tempo trascorso non valido P2: tempo trascorso valido
Voto	V1: voto valido

Parametro: Titolo	
Nome categoria	**Scelte per la categoria**
Titolo	Lunghezza >30 OR Lunghezza<5 [errore] Lunghezza <30 AND Lunghezza >=5 [corretto]
Parametro: Descrizione	
Nome categoria	**Scelte per la categoria**
Descrizione	Lunghezza >400 OR Lunghezza<5 [errore] Lunghezza <400 AND Lunghezza >=5 [corretto]
Parametro: Periodo	
Nome categoria	**Scelte per la categoria**
Periodo	DataOggi - DataAcquisto < 30gg [errore] DataOggi - DataAcquisto >30gg [corretto]

Parametro: Titolo	
Nome categoria	**Scelte per la categoria**
Titolo	Lunghezza >30 OR Lunghezza<5 [errore] Lunghezza <30 AND Lunghezza >=5 [corretto]

Test-Case ID	**Test Frame**	**Esito**
TC_3.3_1	T1	Errore: lunghezza titolo non valida
TC_3.3_2	T2-D1	Errore: lunghezza descrizione non valida
TC_3.3_3	T2-D2-P1	Errore: Periodo non valido
TC_3.3_4	T2-D2-P2-V1-P2	Recensione inviata

10.3.4: Segnalazione locatore

Parametri:	Motivo, Segnalazione, Descrizione
Oggetti dell'ambiente:	Database
Categorie:	**Scelte**
Motivo Segnalazione	M1: Motivo Segnalazione non selezionato M2: Motivo Segnalazione selezionato
Descrizione	D1 Descrizione non valida D2 Descrizione valida

Parametro: Motivo Segnalazione	
Nome categoria	**Scelte per la categoria**
Motivo Segnalazione	1. Selezionare opzioni <1 [errore] 2. Selezionare opzioni >1 [errore] 3. Selezionare opzioni=1 [corretto]
Parametro: Descrizione	
Nome categoria	**Scelte per la categoria**
Descrizione	1. Lunghezza ==0 OR > 300 [errore] 2. Lunghezza > 0 AND < 300 [corretto]

Test-Case ID	**Test Frame**	**Esito**

TC_3.4_1	M1	Errore: Nessun parametro selezionato
TC_3.4_2	M1-D2	Errore: motivo segnalazione non selezionato
TC_3.4_3	M2-D2	Errore: più di un motivo selezionato
TC_3.4_4	D1-M3	Errore: Descrizione non conforme
TC_3.4_5	M3-D2	Corretto: Segnalazione inviata con successo

11: Unit Test Cases

11.1: GestioneUtente

11.1.1: ClienteDAO

Metodo:	createCliente(cliente: Cliente)
Parametri:	Cliente cliente
Oggetti dell'ambiente:	Database
Categorie:	**Scelte**
Cliente	• CNPDB: il cliente che stiamo provando a salvare nel database non è presente. • CPDB: il cliente che stiamo provando a salvare nel database è già presente. • CDM: il cliente che stiamo provando a salvare nel database ha dei parametri mancanti • CN: l'istanza è None
Vincoli:	
Test Frame:	
CNPDB	Oracolo: Viene effettuata l'operazione ed inserita una nuova istanza nel database contenente tutti i dati del parametro cliente.
CPDB	Oracolo: Viene generato un'eccezione perché stiamo cercando di salvare un'istanza di Cliente già presente nel database. In questo caso l'operazione non viene effettuata ed il database non subisce modifiche.
CDM	Oracolo: Viene generato un'eccezione perché stiamo cercando di salvare un'istanza di Cliente con dei parametri mancanti. In questo caso l'operazione non viene effettuata ed il database non subisce modifiche.

CN	Oracolo: Viene generato un'eccezione perché il metodo non può essere effettuato in quanto stiamo passando come parametro un'istanza None. In questo caso l'operazione non viene effettuata ed il database non subisce modifiche.

11.1.2: DipendenteDAO

Metodo:	registra_homecheker(Dipendente)
Parametri:	Dipedente dipendente
Oggetti dell'ambiente:	Database
Categorie:	**Scelte**
Registrazione	•RV: l'homecheker che stiamo provando a salvare nel database non è presente e contiene tutti i campi riempiti. •RN: l'istanza è None •RNV: le informazioni relative all'Homecheker non sono valide
Vincoli:	
Test Frame:	
RV	Oracolo: Viene effettuata l'operazione e viene inserito un nuovo Homechecker all'interno del database
RN	Oracolo: Viene generato un errore perché il metodo non può essere effettuato in quanto stiamo passando come parametri delle istanze None. In questo caso l'operazione non viene effettuata ed il database non subisce modifiche.
RNV	Oracolo: Viene generato un errore perché il metodo non può essere effettuato in quanto stiamo passando un dipendente con degli errori sulla definizione degli attributi
RNVDB	Oracolo: Viene generato un errore perché il metodo non può essere effettuato in quanto stiamo passando un dipendente già presente nel database

11.2: GestioneAnnunci

11.2.1: AlloggioDAO

Metodo:	create_alloggio(Alloggio)
Parametri:	Alloggio alloggio
Oggetti dell'ambiente:	Database

Categorie:	**Scelte**
Alloggio	• AV: L'alloggio che stiamo provando a salvare nel database non è presente. • ANN: l'istanza è None • ANV: L'alloggio che vogliamo inserire presenta dei campi mancanti nel form
Vincoli:	
	Test Frame:
AV	Oracolo: Viene effettuata l'operazione ed inserita una nuova istanza nel database contenente tutti i dati passati come parametro.
ANV	Oracolo: Viene generato un errore in quanto stiamo cercando pubblicare un annuncio (e quindi fare un inserimento nel database) con dei campi mancanti. In questo caso l'operazione non viene effettuata ed il database non subisce modifiche.
ANN	Oracolo: Viene generato un errore perché stiamo cercando di pubblicare un annuncio (e quindi fare un inserimento nel database) con l'intera istanza None quindi in questo caso l'annuncio non viene pubblicato e il database non subisce modifiche

11.2.2: PostDAO

Metodo:	createPost(post: Post)
Parametri:	Post post
Oggetti dell'ambiente:	Database
Categorie:	**Scelte**
Post	• PV: il post che vogliamo pubblicare presenta tutti i campi riempiti. • PNV: il post che vogliamo pubblicare presenta dei campi mancanti nel form. • PNN: l'istanza è None.
Vincoli:	
	Test Frame:
PV	Oracolo: Viene effettuata l'operazione ed inserito il post all'interno del database con tutti i dati del parametro del post.

PNV	Oracolo: Viene generato un errore in quanto stiamo cercando di pubblicare un post (e quindi fare un inserimento nel database) con dei campi mancanti. In questo caso l'operazione non viene effettuata ed il database non subisce modifiche.
PNN	Oracolo: Viene generato un errore in quanto stiamo cercando di pubblicare un post (e quindi fare un inserimento nel database) con l'intera istanza None. In questo caso l'operazione non viene effettuata ed il database non subisce modifiche.

11.3: GestioneAffitto

11.3.1: AffitareDAO

Metodo:	creaaffitto(Affittare)
Parametri:	Affitto affitto
Oggetti dell'ambiente:	Database
Categorie:	**Scelte**
Creazione	•C: l'affitto che stiamo creando non è già presente nel DB •CN: l'istanza è None •CNDO: un parametro di affittare è none
Vincoli:	
Test Frame:	
CV	Oracolo: Viene effettuata l'operazione ed inserita una nuova istanza nel database contenente tutti i dati passati come parametro.
CN	Oracolo: Viene generato un errore perché il metodo non può essere effettuato in quanto stiamo passando come parametri dell'stanze None. In questo caso l'operazione non viene effettuata ed il database non subisce modifiche.
CNDO	Oracolo: Viene generato un errore perchè il metodo non può essere effettuato in quanto stiamo passando un parametro none, l'operazione viene quindi annullata e il database non subisce modifiche

11.3.2: RecensioneDAO

Metodo:	recensione_alloggio(recensione: Recensione)
Parametri:	Recensione recensione

Oggetti dell'ambiente:	Database
Categorie:	**Scelte**
Recensione valida	• RV: la recensione che stiamo provando a salvare nel database non è presente. • RN: l'istanza è None • RNV: campi mancanti
Vincoli:	
Test Frame:	
RV	Oracolo: Viene effettuata l'operazione ed inserita una nuova istanza nel database contenente tutti i dati passati come parametro.
RN	Oracolo: Viene generato un errore perché il metodo non può essere effettuato in quanto stiamo passando come parametri dell'stanze None. In questo caso l'operazione non viene effettuata ed il database non subisce modifiche.
RNV	Viene generato un errore perché il metodo non può essere effettuato in quanto stiamo passando come parametro un'istanza con elementi vuoti. In questo caso l'operazione non viene effettuata ed il database non subisce modifiche.