Progetto di Ingegneria del Software

ANNO ACCADEMICO 2017/2018

Destinatario: Sergio di Martino

DOCUMENTO DI TESTING

gruppo 17

Stefano Falangone	N86002437
Alessandro Caputo	N86001934
Crescenzo Cipolletta	N86001866

Documento di Testing

Si desidera testare sia che il progetto sia conforme ai requisiti funzionali specificati, e dunque il progetto risulti essere completo rispetto ad essi, sia la correttezza dei metodi implementati. Per la seconda parte si è utilizzato jUnit andando a stressare quattro classi corrispondenti ai Data Access Object implementati.

System Testing

Il seguente testing plan copre tutte le funzionalità offerte dal sistema all'utente in ogni tab possibile del progetto software.

Test id	1	
Nome test	Cerca Cliente Test	
Descrizione test	Obiettivo di questo test è verificare la funzionalità di ricerca di un cliente	
Input	**Risultato desiderato**	**Risultato ottenuto**
Inserire "Alessandro" nel campo "Nome" e premere "Cerca"	Mostra la riga nei risultati Nome: Alessandro Cognome: Caputo eMail: alecaputo97@hotmail.it Codice fiscale: CPTLSN97H23F839Q Data di nascita: 23 giugno 1997	
Inserire nel campo email "aaa" e premere "Cerca"	Non trova niente e riporta il messaggio *"ERRORE: la mail non è nel formato testo@dominio.testo"*	
Inserire nel campo codice fiscale "aaa" e premere "Cerca"	Non trova niente e riporta il messaggio *"ERRORE: Il codice fiscale deve essere di 16 caratteri"*	
Note	Il database deve contenere nella table Cliente **"**Nome: Alessandro Cognome: Caputo eMail: alecaputo97@hotmail.it Codice fiscale: CPTLSN97H23F839Q Data di nascita: 23 giugno 1997". L'utente deve essere nella tab "Clienti".	

Test id	2	
Nome test	Inserisci Cliente Test	
Descrizione test	Obiettivo di questo test è verificare la funzionalità di inserimento di un cliente	
Input	**Risultato desiderato**	**Risultato ottenuto**
Inserire Nome=”Alberto” Cognome=”Rossi”,CodiceFiscale=”JLMRFN53T47A448U,eMail=”Alberto@gmail.it”Data di nascita=”7 aprile 1995” e cliccare su Inserisci	Mostra il messaggio “Cliente inserito correttamente” e ora i dati del cliente si trovano nel database.	
Inserire Nome=”Alberto” Cognome=”Rossi”,CodiceFiscale=”CPTLSN97H23F839Q,eMail=”Alberto@gmail.it”Data di nascita=”7 aprile 1995” e cliccare su Inserisci	Mostra il messaggio *“ERRORE: email già esistente” senza salvare i dati del cliente*	
Inserire Nome=”Alberto” Cognome=”Rossi”,CodiceFiscale=”JLMRFN53T47A448U,eMail=”alecaputo97@hotmail.it”Data di nascita=”7 aprile 1995” e cliccare su Inserisci	Mostra il messaggio *“ERRORE: Codice Fiscale già esistente” senza salvare i dati del cliente*	
Non inserire nessun campo e cliccare su Inserisci	Mostra il messaggio *“ERRORE: almeno uno dei campi è vuoto” senza salvare i dati del cliente*	
Inserire Nome=”Alberto” Cognome=”Rossi”,CodiceFiscale=”aaa,eMail=”Alberto@gmail.it”Data di nascita=”7 aprile 1995” e cliccare su Inserisci	Mostra il messaggio *“ERRORE: la mail non è nel formato testo@dominio.testo” senza salvare i dati del cliente*	
Inserire Nome=”Alberto” Cognome=”Rossi”,CodiceFiscale=”CPTLSN97H23F839Q,eMail=”aaa”Data di nascita=”7 aprile 1995” e	*“ERRORE: il codice fiscale non è di 16 caratteri” senza salvare i dati del cliente*	

cliccare su Inserisci		
Note	Il database deve contenere nella table Cliente **“**Nome: Alessandro Cognome: Caputo eMail: alecaputo97@hotmail.it Codice fiscale: CPTLSN97H23F839Q Data di nascita: 23 giugno 1997”. L’utente deve essere nella tab “Clienti”.	

Test id	3	
Nome test	Modifica Cliente Test	
Descrizione test	Obiettivo di questo test è verificare la funzionalità di modifica di un cliente	
Input	**Risultato desiderato**	**Risultato ottenuto**
Selezionare la riga con codice fiscale “CPTLSN97H23F839Q” e premere Modifica. Dopo il riempimento dei campi, si modifichi il nome da “Alessandro” a “Francesco” e si prema “Conferma”	Mostra il messaggio “Cliente modificato correttamente” e modifica il nome del cliente con quel codice fiscale nel Database	
Selezionare la riga con codice fiscale “CPTLSN97H23F839Q” e premere Modifica. Dopo il riempimento dei campi, si modifichi la mail da “alecaputo97@hotmail.it” a “antonino@gmail.com” e si prema “Conferma”	Mostra il messaggio “ ERRORE: email già esistente” e non modifica la mail	
Selezionare la riga con codice fiscale “CPTLSN97H23F839Q” e premere Modifica. Dopo il riempimento dei campi, si modifichi la mail da “alecaputo97@hotmail.it” a “” e si prema “Conferma”	Mostra il messaggio “ERRORE: almeno un campo è vuoto” e non modifica la mail	

Selezionare la riga con codice fiscale “CPTLSN97H23F839Q” e premere Modifica. Dopo il riempimento dei campi, si modifichi la mail da “alecaputo97@hotmail.it” a “alecaputo@hotmail” e si prema “Conferma”	Mostra il messaggio “ERRORE: la mail non è nel formato testo@dominio.testo” e non modifca la mail	
Note	Il database deve contenere nella table Cliente **“**Nome: Alessandro Cognome: Caputo eMail: alecaputo97@hotmail.it Codice fiscale: CPTLSN97H23F839Q Data di nascita: 23 giugno 1997”. L’utente deve essere nella tab “Clienti”.	

Test id	4	
Nome test	Elimina Cliente Test	
Descrizione test	Obiettivo di questo test è verificare la funzionalità di eliminazione di un cliente	
Input	**Risultato desiderato**	**Risultato ottenuto**
Selezionare la riga con codice fiscale “JLMRFN53T47A448U” e premere su elimina,successivamente premere “SI” nel popup	Mostra il messaggio “Cliente eliminato correttamente” e il cliente selezionato è stato eliminato dal database	
Selezionare la riga con codice fiscale “CPTLSN97H23F839Q” e premere su elimina,successivamente premere “NO” nel popup	Si chiude il popup e non succede niente	
Note	Il database deve contenere nella table Cliente **“**Nome: Alessandro Cognome: Caputo eMail: alecaputo97@hotmail.it Codice fiscale: CPTLSN97H23F839Q Data di nascita: 23 giugno 1997” e **“**Nome: Alberto	

	Cognome: Rossi eMail: alberto@gmail.com Codice fiscale: JLMRFN53T47A448U Data di nascita: 7 aprile 1995” L’utente deve essere nella tab “Clienti”.

Test id	5	
Nome test	Visualizza statistiche cliente Test	
Descrizione test	Obiettivo di questo test è verificare la funzionalità di visualizzazione delle statistiche di un cliente	
Input	**Risultato desiderato**	**Risultato ottenuto**
Inserire “CPTLSN97H23F839Q” nel campo “Codice Fiscale”, dopodiché selezionare la riga con quel codice fiscale e fare click sul tasto “Statistiche”	Mostra una schermata con le statistiche relative a quel cliente	
Note	Il database deve contenere nella table Cliente **“**Nome: Alessandro Cognome: Caputo eMail: alecaputo97@hotmail.it Codice fiscale: CPTLSN97H23F839Q Data di nascita: 23 giugno 1997” e si deve essere nella tab “Cliente”. L’utente deve essere nella tab “Clienti”.	

Test id	6	
Nome test	Cerca Evento Test	
Descrizione test	Obiettivo di questo test è verificare la funzionalità di ricerca di un un evento	
Input	**Risultato desiderato**	**Risultato ottenuto**
Inserire “Reputation stadium	Mostra la riga nei risultati	

tour” nel campo “Nome” e clickare sul tasto “Cerca”	Nome: Reputation stadium tour Luogo: Croke park Data: 15 giugno 2018 Data Inserimento: 31 gennaio 2018 Prezzo iniziale: 75 Prezzo finale: 600 Tipo: Concerto Max Posti: 60000 Prezzo corrente: (valore numerico compreso tra 75 e 600 se l’evento non è già avvenuto, altrimenti “Non disponibile”)	
Inserire “aaa” nel campo “Prezzo iniziale” e clickare sul tasto “Cerca”	Non trova niente e riporta il messaggio *“ERRORE:* Prezzo iniziale, finale e numero di spettatori devono essere valori numerici!*”*	
Inserire “aaa” nel campo “Prezzo finale” e clickare sul tasto “Cerca”	Non trova niente e riporta il messaggio *“ERRORE:* Prezzo iniziale, finale e numero di spettatori devono essere valori numerici!*”*	
Inserire “aaa” nel campo “Massimo postii” e clickare sul tasto “Cerca”	Non trova niente e riporta il messaggio *“ERRORE:* Prezzo iniziale, finale e numero di spettatori devono essere valori numerici!*”*	
Note	Il database deve contenere nella table Evento “Nome: Reputation stadium tour Luogo: Croke park Data: 15 giugno 2018 Data Inserimento: 31 gennaio 2018 Prezzo iniziale: 75 Prezzo finale: 600 Tipo: Concerto Max Posti: 60000” e nella table Luogo “Nome:Croke park Città: Dublino Indirizzo: Jones' rd drumcondra Stato: Irlanda”. L’utente deve essere nella tab “Eventi”.	

Test id	7	
Nome test	Inserisci Evento Test	
Descrizione test	Obiettivo di questo test è verificare la funzionalità di inserimento di un evento	
Input	**Risultato desiderato**	**Risultato ottenuto**
Inserire Nome=”U2” Prezzo Iniziale=”40”,Prezzo finale=”200”, massimo posti=”40000”, Luogo=”San paolo”, Tipo=”Concerto”, Data=”2 maggio 2018” e cliccare su Inserisci	Mostra il messaggio “Evento inserito correttamente” e ora i dati dell’evento si trovano nel database.	
Inserire Nome=”Reputation stadium tour” Prezzo Iniziale=”40”,Prezzo finale=”200”, massimo posti=”40000”, Luogo=”San paolo”, Tipo=”Concerto”, Data=”2 maggio 2018” e cliccare su Inserisci	Mostra il messaggio “ERRORE: evento già esistente” senza salvare i dati dell’evento	
Inserire Nome=”U2” Prezzo Iniziale=”200”,Prezzo finale=”80”, massimo posti=”40000”, Luogo=”Croke park”, Tipo=”Concerto”, Data=”2 maggio 2018” e cliccare su Inserisci	Mostra il messaggio “ERRORE: Prezzo finale minore del prezzo iniziale” senza salvare i dati dell’evento	
Inserire Nome=”U2” Prezzo Iniziale=””,Prezzo finale=”80”, massimo posti=”40000”, Luogo=”Croke park”, Tipo=”Concerto”, Data=”2 maggio 2018” e cliccare su Inserisci	Mostra il messaggio “ERRORE: almeno uno dei campi è vuoto” senza salvare i dati dell’evento	
Inserire Nome=”U2” Prezzo Iniziale=”aa”,Prezzo finale=”80”, massimo posti=”4000”, Luogo=”Croke park”, Tipo=”Concerto”, Data=”29 aprile 2018” e	Mostra il messaggio “ERRORE: Prezzo iniziale,finale e numero di spettatori devono essere	

cliccare su Inserisci	valori numerici! ” senza salvare i dati dell’evento	
Inserire Nome=”U2” Prezzo Iniziale=”200”,Prezzo finale=”aa”, massimo posti=”40000”, Luogo=”Croke park”, Tipo=”Concerto”, Data=”2 maggio 2018” e cliccare su Inserisci	Mostra il messaggio “ERRORE: Prezzo iniziale,finale e numero di spettatori devono essere valori numerici! ” senza salvare i dati dell’evento	
Inserire Nome=”U2” Prezzo Iniziale=”200”,Prezzo finale=”600”, massimo posti=”aa”, Luogo=”Croke park”, Tipo=”Concerto”, Data=”2 maggio 2018” e cliccare su Inserisci	Mostra il messaggio “ERRORE: Prezzo iniziale,finale e numero di spettatori devono essere valori numerici! ” senza salvare i dati dell’evento	
Note	Il database deve contenere nella table Evento “Nome: Reputation stadium tour Luogo: Croke park Data: 15 giugno 2018 Data Inserimento: 31 gennaio 2018 Prezzo iniziale: 75 Prezzo finale: 600 Tipo: Concerto Max Posti: 60000” e nella table Luogo “Nome:Croke park Città: Dublino Indirizzo: Jones' rd drumcondra Stato: Irlanda”. L’utente deve essere nella tab “Eventi”.	

Test id	8	
Nome test	Modifica Evento Test	
Descrizione test	Obiettivo di questo test è verificare la funzionalità di modifica di un evento	
Input	**Risultato desiderato**	**Risultato ottenuto**
Selezionare la riga con		

nome “Reputation stadium tour” e premere Modifica. Dopo il riempimento dei campi, si modifichi il prezzo finale da “600” a “400” e si prema “Conferma”	Mostra il messaggio “Evento modificato correttamente” e modifica il prezzo finale dell’evento con quel nome nel Database	
Selezionare la riga con nome “Reputation stadium tour” e premere Modifica. Dopo il riempimento dei campi, si modifichi il prezzo finale da “600” a “10” e si prema “Conferma”	Mostra il messaggio “ ERRORE: prezzo finale minore del prezzo iniziale” e non modifica il prezzo finale dell’evento con quel nome nel Database	
Selezionare la riga con nome “Reputation stadium tour” e premere Modifica. Dopo il riempimento dei campi, si modifichi il prezzo iniziale da “75” a “aa” e si prema “Conferma”	Mostra il messaggio “ ERRORE: prezzo iniziale non è un numero” e non modifica il prezzo iniziale dell’evento con quel nome nel Database	
Selezionare la riga con nome “Reputation stadium tour” e premere Modifica. Dopo il riempimento dei campi, si modifichi il prezzo finale da “600” a “aa” e si prema “Conferma”	Mostra il messaggio “ ERRORE: prezzo finale non è un numero” e non modifica il prezzo finale dell’evento con quel nome nel Database	
Selezionare la riga con nome “Reputation stadium tour” e premere Modifica. Dopo il riempimento dei campi, si modifichi il massimo spettatori da “60000” a “aa” e si prema “Conferma”	Mostra il messaggio “ ERRORE: massimo spettatori non è un numero” e non modifica il massimo spettatori dell’evento con quel nome nel Database	
Selezionare la riga con nome “Reputation stadium tour” e premere Modifica. Dopo il riempimento dei campi, si modifichi il massimo spettatori da “60000” a “-20” e si prema “Conferma”	Mostra il messaggio “ ERRORE: massimo spettatori deve essere maggiore di 0” e non modifica il massimo spettatori dell’evento con quel nome nel Database	
Selezionare la riga con nome “Reputation stadium	Mostra il messaggio “ ERRORE: prezzo iniziale e	

tour” e premere Modifica. Dopo il riempimento dei campi, si modifichi il prezzo finale da “75” a “-20” e si prema “Conferma”	finale devono essere >=0” e non modifica il prezzo iniziale dell’evento con quel nome nel Database	
Selezionare la riga con nome “Reputation stadium tour” e premere Modifica. Dopo il riempimento dei campi, si modifichi il prezzo finale da “600” a “-20” e si prema “Conferma”	Mostra il messaggio “ ERRORE: prezzo iniziale e finale devono essere >=0” e non modifica il prezzo finale dell’evento con quel nome nel Database	
Note	Il database deve contenere nella table Evento “Nome: Reputation stadium tour Luogo: Croke park Data: 15 giugno 2018 Data Inserimento: 31 gennaio 2018 Prezzo iniziale: 75 Prezzo finale: 600 Tipo: Concerto Max Posti: 60000” e nella table Luogo “Nome:Croke park Città: Dublino Indirizzo: Jones' rd drumcondra Stato: Irlanda”. L’utente deve essere nella tab “Eventi”.	

Test id	9	
Nome test	Elimina Evento Test	
Descrizione test	Obiettivo di questo test è verificare la funzionalità di eliminazione di un cliente	
Input	**Risultato desiderato**	**Risultato ottenuto**
Selezionare la riga con nome “Finley” e premere su elimina, successivamente premere “SI” nel popup	Mostra il messaggio “Evento eliminato correttamente” e l’evento selezionato viene eliminato dal database	
Selezionare la riga con nome “Finley” e premere su elimina, successivamente premere “NO” nel popup	Chiude il popup senza procedere con l’eliminazione	

Selezionare la riga con nome “Reputation stadium tour” e premere su elimina, successivamente premere “SI” nel popup	Mostra messaggio “ERRORE: Evento non eliminabile, c’è almeno un biglietto venduto” e non elimina l’evento selezionato dal database	
Note	Il database deve contenere nella table Evento “Nome: Reputation stadium tour Luogo: Croke park Data: 15 giugno 2018 Data Inserimento: 31 gennaio 2018 Prezzo iniziale: 75 Prezzo finale: 600 Tipo: Concerto Max Posti: 60000”, “Nome: Finley Luogo: Croke park Data: 15 giugno 2018 Data Inserimento: 29 aprile 2018 Prezzo iniziale: 20 Prezzo finale: 400 Tipo: Concerto Max Posti: 2000” e nella table Luogo “Nome:Croke park Città: Dublino Indirizzo: Jones' rd drumcondra Stato: Irlanda”. L’utente deve essere nella tab “Eventi”.	

Test id	10	
Nome test	Visualizza statistiche evento Test	
Descrizione test	Obiettivo di questo test è verificare la funzionalità di visualizzazione delle statistiche di un evento	
Input	**Risultato desiderato**	**Risultato ottenuto**
Inserire “Reputation stadium tour” nel campo “Nome”, dopodiché selezionare la riga con quel nome e fare click sul tasto “Statistiche”	Mostra una schermata con le statistiche relative a quell’evento	

Note	Il database deve contenere nella table Evento “Nome: Reputation stadium tour Luogo: Croke park Data: 15 giugno 2018 Data Inserimento: 31 gennaio 2018 Prezzo iniziale: 75 Prezzo finale: 600 Tipo: Concerto Max Posti: 60000”. L’utente deve essere nella tab “Eventi”.	

Test id	11	
Nome test	Cerca Luogo Test	
Descrizione test	Obiettivo di questo test è verificare la funzionalità di ricerca di un luogo	
Input	**Risultato desiderato**	**Risultato ottenuto**
Inserire “Croke park” nel campo “Nome” e premere “Cerca”	Mostra la riga nei risultati Nome: Croke park Città: Dublino Stato: Irlanda Indirizzo: Jones ‘rd drumcondra	
Note	Il database deve contenere nella table Luogo “Nome: Croke park Città: Dublino Stato: Irlanda Indirizzo: Jones ‘rd drumcondra”. L’utente deve essere nella tab “Luoghi”.	

Test id	12	
Nome test	Inserisci Luogo Test	
Descrizione test	Obiettivo di questo test è verificare la funzionalità di inserimento di un luogo	
Input	**Risultato desiderato**	**Risultato ottenuto**
Inserire Nome:”Arena di Verona” Città: “Verona” Stato: “Italia” Indirizzo: “arena di verona, 97” e cliccare su Inserisci	Mostra il messaggio “Luogo inserito correttamente” e ora i dati del luogo si trovano nel database.	
Inserire Nome:”Croke park” Città: “Verona” Stato: “Italia” Indirizzo: “arena di verona, 97” e cliccare su Inserisci	Mostra il messaggio “ERRORE: luogo già esistente” senza salvare i dati del luogo	
Inserire Nome:”Arena di Verona” Città: “” Stato: “Italia” Indirizzo: “arena di verona, 97” e cliccare su Inserisci	Mostra il messaggio “ERRORE: almeno uno dei campi è vuoto” senza salvare i dati del luogo	
Note	Il database deve contenere nella table Luogo “Nome:Croke park Città: Dublino Indirizzo: Jones' rd drumcondra Stato: Irlanda”. L’utente deve essere nella tab “Luoghi”.	

Test id	13	
Nome test	Modifica Luogo Test	
Descrizione test	Obiettivo di questo test è verificare la funzionalità di modifica di un evento	
Input	**Risultato desiderato**	**Risultato ottenuto**
Selezionare la riga con nome “Croke Park” e	Mostra il messaggio “Luogo	

premere Modifica. Dopo il riempimento dei campi, si modifichi la Città da “Dublino” a “Roma” e si prema “Conferma”	modificato correttamente” e modifica la città del luogo con quel nome nel Database	
Selezionare la riga con nome “Croke Park” e premere Modifica. Dopo il riempimento dei campi, si modifichi la Città da “Dublino” a “” e si prema “Conferma”	Mostra il messaggio “ ERRORE: almeno uno dei campi è vuoto” e non modifica il luogo del luogo con quel nome nel Database	
Note	Il database deve contenere nella table Luogo “Nome:Croke park Città: Dublino Indirizzo: Jones' rd drumcondra Stato: Irlanda”. L’utente deve essere nella tab “Luoghi”.	

Test id	14	
Nome test	Elimina Luogo Test	
Descrizione test	Obiettivo di questo test è verificare la funzionalità di eliminazione di un cliente	
Input	**Risultato desiderato**	**Risultato ottenuto**
Selezionare la riga con nome “Arena di Verona” e premere su elimina, successivamente premere “SI” nel popup	Mostra il messaggio “Luogo eliminato correttamente” e l’evento selezionato viene eliminato dal database	
Selezionare la riga con nome “Arena” e premere su elimina, successivamente premere “NO” nel popup	Chiude il popup senza procedere con l’eliminazione	
Selezionare la riga con nome “Croke park” e premere su elimina, successivamente premere “SI” nel popup	Mostra messaggio “ERRORE: Luogo non eliminabile, c’è almeno un biglietto venduto” e non elimina l’evento selezionato dal database	
Note	Il database deve contenere nella table Evento “Nome: Reputation stadium tour	

	Luogo: Croke park Data: 15 giugno 2018 Data Inserimento: 31 gennaio 2018 Prezzo iniziale: 75 Prezzo finale: 600 Tipo: Concerto Max Posti: 60000”, e nella table Luogo “Nome:Croke park Città: Dublino Indirizzo: Jones' rd drumcondra Stato: Irlanda”, “Nome: Arena di Verona Città: Verona Stato: Italia Indirizzo: arena di verona, 97”

Test id	15	
Nome test	Visualizza statistiche luogo Test	
Descrizione test	Obiettivo di questo test è verificare la funzionalità di visualizzazione delle statistiche di un luogo	
Input	**Risultato desiderato**	**Risultato ottenuto**
Inserire “Croke park” nel campo “Nome”, dopodiché selezionare la riga con quel nome e fare click sul tasto “Statistiche”	Mostra una schermata con le statistiche relative a quell’evento	
Note	Il database deve contenere nella table Luogo “Nome:Croke park Città: Dublino Indirizzo: Jones' rd drumcondra Stato: Irlanda”. L’utente deve essere nella tab “Luoghi”.	

Test id	16
Nome test	Visualizza statistiche generali Test

Descrizione test	Obiettivo di questo test è verificare la funzionalità di visualizzazione delle statistiche generali	
Input	**Risultato desiderato**	**Risultato ottenuto**
Clickare sul tasto “Tipo evento per luogo”	Mostra una schermata con le statistiche relative a quell’evento	
Clickare sul tasto “Guadagno per evento”	Mostra una schermata con le statistiche relative a quell’evento	
Clickare sul tasto “Fasce età clienti”	Mostra una schermata con le statistiche relative a quell’evento	
Note	L’utente deve essere nella tab “Statistiche”.	

JUnit Tests

Il testing effettuato è su quattro classi control *(BigliettoDAO, ClienteDAO, EventoDAO, LuogoDAO),* per ogni classe è stata creata una classe di testing per stressare ogni metodo del corrispondente DAO ad eccezione delle eliminazioni.

Perché il testing dei Data Access Object: si è ritenuto fondamentale poter testare in tempi rapidi che le operazioni di crud fossero “corrette” così da poter localizzare eventuali fault all’interno delle altre classi.

Per mantenere l’integrità del database non si è testato il metodo di eliminazione per ogni DAO, tuttavia una possibile soluzione sarebbe potuta essere il creare uno script che utilizzando le api di Aws inserisca un elemento apposito, dopodiché esegua il testing ed al termine proceda con l’eliminazione dell’elemento iniziale.

Il testing dei metodi di ricerca ha presentato un problema strutturale in merito alla ricerca senza attributi che deve riportare l’intera table corrispondente. Poiché si è voluto assicurare che i test fossero generali ed eseguibili un numero arbitrario di volte a prescindere dallo stato del database (table vuota oppure con più elementi di numero non vuoto) si è considerata sufficiente la condizione che nella ricerca senza parametri, che ritorna una List dell’entity corrispondente, vi sia almeno un elemento che è quello usato per il testing nell’inserimento. Ancora una volta, utilizzare uno script apposito avrebbe potuto rendere il testing più efficace inserendo n elementi per poi assicurarsi che la ricerca senza parametri ritorni almeno n elementi.
Nota: al fine di poter testare senza script, si considerano presenti di default nel database gli elementi:
(table) **Biglietto**

```
{
  "CodFiscale": "MCHSVM73R09F284X",
  "DataAcquisto": "1 aprile 2018",
  "Evento": "Test2018",
  "Luogo": "Croke park",
  "NumeroBiglietto": "8710a",
  "Prezzo": 112
}
```

(table) **Cliente**

```
{
  "CodiceFiscale": "MCHSVM73R09F284X",
  "Cognome": "Caparezza",
```

"DataNascita": "9 ottobre 1973",
"Email": "michelecaparezza@gmail.com",
"Nome": "Michele"
}

(table) **Evento**

{
"DataEvento": "4 giugno 2018",
"DataInserimento": "14 febbraio 2018",
"Luogo": "Croke park",
"MassimoSpettatori": 200,
"Nome": "Test2018",
"PrezzoFinale": 20,
"PrezzoIniziale": 10,
"Tipo": "altro"
}

(table) **Luogo**

{
"Città": "Napoli",
"Indirizzo": "via Cinthia",
"Nome": "Test2018",
"Stato": "Italia"
}

BigliettoDAOTest

```
package testing;

import static org.junit.jupiter.api.Assertions.*;

import org.junit.After;
import org.junit.Assert;
import org.junit.Before;
import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;
import com.amazonaws.services.dynamodbv2.document.DynamoDB;

import boundary.FinestraUtente;
import control.BigliettoDAO;

import java.util.*;
```

```
import entity.*;

class BigliettoDAOTest {
    static AmazonDynamoDB dynamoDB;
    private static DynamoDB connessione;
    private BigliettoDAO bigliettoDAO;

       @BeforeEach //crea la connessione con il database hostato su AWS
       void setUp() throws Exception {
              ProfileCredentialsProvider credentialsProvider = new
ProfileCredentialsProvider();
        try {
            credentialsProvider.getCredentials();
        } catch (Exception e) {
        }
        dynamoDB = AmazonDynamoDBClientBuilder.standard()
            .withCredentials(credentialsProvider)
            .withRegion("eu-central-1")
            .build();
             connessione = new DynamoDB(dynamoDB);
           bigliettoDAO=new BigliettoDAO(connessione);
       }


       @Test
       void testCercaPerCodiceFiscale() {
              List<Biglietto> risultato =
bigliettoDAO.cercaPerCodiceFiscale("MCHSVM73R09F284X");
              for(Biglietto curr:risultato) {
                     if(curr.getNumeroBiglietto().equals("8710a")) {
                                          assertEquals("8710a",
curr.getNumeroBiglietto());
                                          assertEquals("MCHSVM73R09F284X",
curr.getCodFiscale());
                                          assertEquals("1 aprile 2018",
curr.getDataAcquisto());
                                          assertEquals("Test2018", curr.getEvento());
                                          assertEquals("Croke park",
curr.getLuogo());
                                          assertEquals(112, curr.getPrezzo());
                     }
              }
       }

       @Test
       void testCercaPerEventoTest() {
              List<Biglietto> risultato = bigliettoDAO.cercaPerEvento("Test");
              for(Biglietto curr:risultato) {
                     if(curr.getNumeroBiglietto().equals("8710a")) {
                                          assertEquals("8710a",
curr.getNumeroBiglietto());
                                          assertEquals("MCHSVM73R09F284X",
curr.getCodFiscale());
```

```
                                        assertEquals("1 aprile 2018",
curr.getDataAcquisto());
                                        assertEquals("Test2018", curr.getEvento());
                                        assertEquals("Croke park",
curr.getLuogo());
                                        assertEquals(112, curr.getPrezzo());
                        }
                }
        }

        @Test
        void testCercaPerLuogo() {
                List<Biglietto> risultato = bigliettoDAO.cercaPerLuogo("Croke park");
                for(Biglietto curr:risultato) {
                        if(curr.getNumeroBiglietto().equals("8710a")) {
                                        assertEquals("8710a",
curr.getNumeroBiglietto());
                                        assertEquals("MCHSVM73R09F284X",
curr.getCodFiscale());
                                        assertEquals("1 aprile 2018",
curr.getDataAcquisto());
                                        assertEquals("Test2018", curr.getEvento());
                                        assertEquals("Croke park",
curr.getLuogo());
                                        assertEquals(112, curr.getPrezzo());
                        }
                }
        }


        @Test
        void testInserisciModifica() {
            bigliettoDAO.inserisciModifica("8710a","MCHSVM73R09F284X","1 MARZO
2018","Test2018","Croke park",112);
            List<Biglietto> risultati =
bigliettoDAO.cercaPerCodiceFiscale("MCHSVM73R09F284X");
            for(Biglietto curr:risultati) {
               if(curr.getNumeroBiglietto().equals("8710a"))  assertEquals("1 MARZO
2018", curr.getDataAcquisto());
            }
            //ripristina il vecchio ticket
            bigliettoDAO.inserisciModifica("8710a","MCHSVM73R09F284X","1 aprile
2018","Test2018","Croke park",112);
          }


}
```

ClienteDAOTest

```
/**
```

```
 *
 */
package testing;

import static org.junit.jupiter.api.Assertions.*;

import org.junit.After;
import org.junit.Assert;
import org.junit.Before;
import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;
import com.amazonaws.services.dynamodbv2.document.DynamoDB;

import boundary.FinestraUtente;
import control.ClienteDAO;

import java.util.*;
import entity.*;
/**
 */
class ClienteDAOTest {
    static AmazonDynamoDB dynamoDB;
    private static DynamoDB connessione;
    private ClienteDAO clienteDAO;
       /**
        * @throws java.lang.Exception
        */
       @BeforeEach //crea la connessione con il database hostato su AWS
       void setUp() throws Exception {
              ProfileCredentialsProvider credentialsProvider = new
ProfileCredentialsProvider();
        try {
            credentialsProvider.getCredentials();
        } catch (Exception e) {
        }
        dynamoDB = AmazonDynamoDBClientBuilder.standard()
            .withCredentials(credentialsProvider)
            .withRegion("eu-central-1")
            .build();
             connessione = new DynamoDB(dynamoDB);
           clienteDAO=new ClienteDAO(connessione);
       }

       /**
        * @throws java.lang.Exception
        */

       @Test
       void testCercaConTuttiParametri() {
```

```
            //Test1: si cerca il cliente inserito nell'operazione di setup
            int count=0;
            List<Cliente> risultati = clienteDAO.cerca("Michele", "Caparezza",
"michelecaparezza@gmail.com", "MCHSVM73R09F284X", "9 ottobre 1973");
            for(Cliente curr:risultati){
                  assertEquals("MCHSVM73R09F284X", curr.getCodiceFiscale());
                  assertEquals("Michele", curr.getNome());
                  assertEquals("Caparezza", curr.getCognome());
                  assertEquals("michelecaparezza@gmail.com", curr.getEmail());
                  assertEquals("9 ottobre 1973", curr.getData());
            }
            //Test2: si cerca se tra tutti i risultati di una ricerca generica vi
è quello inserito nel setup
            risultati = clienteDAO.cerca("", "", "", "", "");
            for(Cliente curr2:risultati){
                  if (curr2.getCodiceFiscale().equals("MCHSVM73R09F284X"))
count++;
            }
            assertEquals(1,count);
      }


      @Test
      void testCercaPerCodiceFiscale() {
            Cliente risultato = clienteDAO.cerca("MCHSVM73R09F284X");
            assertEquals("MCHSVM73R09F284X", risultato.getCodiceFiscale());
            assertEquals("Michele", risultato.getNome());
            assertEquals("Caparezza", risultato.getCognome());
            assertEquals("michelecaparezza@gmail.com", risultato.getEmail());
            assertEquals("9 ottobre 1973", risultato.getData());
      }

      @Test
      void testInserisciModifica() {
          clienteDAO.inserisciModifica("Ani", "Caparezza",
"michelecaparezza@gmail.com", "MCHSVM73R09F284X", "9 ottobre 1973");
          List<Cliente> risultati = clienteDAO.cerca("", "", "",
"MCHSVM73R09F284X", "");
          for(Cliente curr:risultati) assertEquals("Ani", curr.getNome());
          //ripristina il vecchio nome
          clienteDAO.inserisciModifica("Michele", "Caparezza",
"michelecaparezza@gmail.com", "MCHSVM73R09F284X", "9 ottobre 1973");
        }


}
```

EventoDAOTest

```
package testing;
```

```
import static org.junit.jupiter.api.Assertions.*;

import java.util.List;

import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;
import com.amazonaws.services.dynamodbv2.document.DynamoDB;

import control.EventoDAO;
import entity.Evento;

class EventoDAOTest {
    static AmazonDynamoDB dynamoDB;
    private static DynamoDB connessione;
    private EventoDAO eventoDAO;
       @BeforeEach
       void setUp() throws Exception {
              ProfileCredentialsProvider credentialsProvider = new
ProfileCredentialsProvider();
        try {
            credentialsProvider.getCredentials();
        } catch (Exception e) {
        }
        dynamoDB = AmazonDynamoDBClientBuilder.standard()
            .withCredentials(credentialsProvider)
            .withRegion("eu-central-1")
            .build();
             connessione = new DynamoDB(dynamoDB);
           eventoDAO=new EventoDAO(connessione);
       }

       @Test
       void testCercaConTuttiParametri() {
              //Test1: si cerca evento inserito nell'operazione di setup
              int count=0;
              List<Evento> risultati = eventoDAO.cerca("Test2018", "4 giugno 2018",
10.00, 20.00, 200, "altro", "Croke park");
              for(Evento curr:risultati){
                     assertEquals("Test2018", curr.getNome());
                     assertEquals("4 giugno 2018", curr.getData());
                     assertEquals(10, curr.getPrezzoIniziale());
                     assertEquals(20, curr.getPrezzoFinale());
                     assertEquals("altro", curr.getTipo());
              }
              //Test2: si cerca se tra tutti i risultati di una ricerca generica vi
è quello inserito nel setup
              risultati = eventoDAO.cerca("", "", 00.00, 00.00, 0,"","");
              for(Evento curr2:risultati){
```

```
                    if (curr2.getNome().equals("Test2018")) count++;
                }
                assertEquals(1,count);
        }

        void testCercaPerNome() {
                Evento risultato = eventoDAO.cerca("Test2018");
                assertEquals("Test2018", risultato.getNome());
                assertEquals("4 giugno 2018", risultato.getData());
                assertEquals(10, risultato.getPrezzoIniziale());
                assertEquals(20, risultato.getPrezzoFinale());
                assertEquals("altro", risultato.getTipo());
        }

        @Test
        void testInserisciModifica() {
            eventoDAO.inserisciModifica("Test2018", "4 giugno 2018", 10.00, 20.00,
200, "altro", "Croke park", "10 MARZO 2018");
            List<Evento> risultati = eventoDAO.cerca("Test2018", "", 00.00, 00.00,
0, "", "");
            for(Evento curr:risultati) {
               assertEquals("Test2018", curr.getNome());
               assertEquals("10 MARZO 2018", curr.getDataInserimento());
            }
            //ripristina il valore di test2018 per i prossimi test
            eventoDAO.inserisciModifica("Test2018", "4 giugno 2018", 10.00, 20.00,
200, "altro", "Croke park", "14 febbraio 2018");
        }
}
```

LuogoDAOTest

```
/**
 *
 */
package testing;

import static org.junit.jupiter.api.Assertions.*;

import org.junit.After;
import org.junit.Assert;
import org.junit.Before;
import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
```

```
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;
import com.amazonaws.services.dynamodbv2.document.DynamoDB;

import boundary.FinestraUtente;
import control.LuogoDAO;

import java.util.*;
import entity.*;
/**
 *
 */
class LuogoDAOTest {
    static AmazonDynamoDB dynamoDB;
    private static DynamoDB connessione;
    private LuogoDAO luogoDAO;

       @BeforeEach //crea la connessione con il database hostato su AWS
       void setUp() throws Exception {
              ProfileCredentialsProvider credentialsProvider = new
ProfileCredentialsProvider();
        try {
            credentialsProvider.getCredentials();
        } catch (Exception e) {
        }
        dynamoDB = AmazonDynamoDBClientBuilder.standard()
            .withCredentials(credentialsProvider)
            .withRegion("eu-central-1")
            .build();
             connessione = new DynamoDB(dynamoDB);
           luogoDAO=new LuogoDAO(connessione);
       }


       @Test
       void testCercaConTuttiParametri() {
              //Test1: si cerca il luogo inserito nell'operazione di setup
              int count=0;
              List<Luogo> risultati =
luogoDAO.cerca("Test2018","Napoli","Italia","via Cinthia");
              for(Luogo curr:risultati){
                     assertEquals("Test2018", curr.getNome());
                     assertEquals("Napoli", curr.getCittà());
                     assertEquals("Italia", curr.getStato());
                     assertEquals("via Cinthia", curr.getIndirizzo());
              }
              //Test2: si cerca se tra tutti i risultati di una ricerca generica vi
è quello inserito nel setup
              risultati = luogoDAO.cerca("", "", "", "");
              for(Luogo curr2:risultati){
                     if (curr2.getNome().equals("Test2018")) count++;
              }
              assertEquals(1,count);
       }
```

```
        @Test
        void testInserisciModifica() {
            luogoDAO.inserisciModifica("Test2018", "MILANO","Italia","via Cinthia");
            List<Luogo>risultati = luogoDAO.cerca("Test2018", "","","");
            for(Luogo curr:risultati) {
               assertEquals("Test2018", curr.getNome());
               assertEquals("MILANO", curr.getCittà());
            }
            //ripristina il vecchio valore della città
            luogoDAO.inserisciModifica("Test2018", "Napoli","Italia","via Cinthia");
        }


}
```