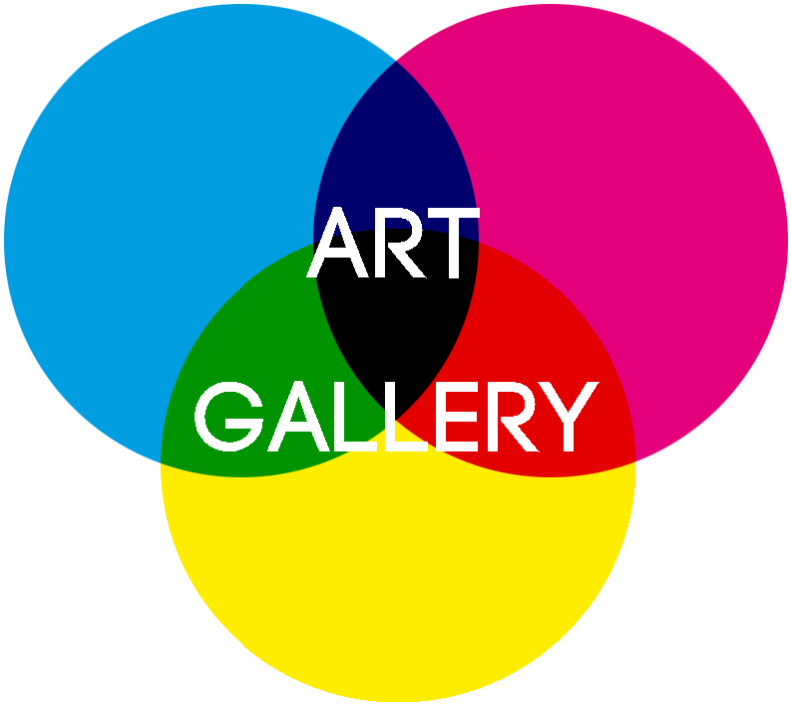
Università degli Studi Di Salerno
Corso di Ingegneria del Software

ArtGallery
Test Plan Document
Versione 1.0



DATA 10/01/2021

Coordinatori del progetto

Nome
De Lucia Andrea
Pecorelli Fabiano

Partecipanti al progetto

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

Scritto da	Emanuele Fittipaldi, Giuseppe Martinelli, Fedele Mauro

REVISION HISTORY

Data	Versione	Descrizione	Autore
07/01/2021	1.0	Features to be tested / not to be tested	Emanuele Fittipaldi
07/01/2021	1.0	Pass/Fail Criteria	Emanuele Fittipaldi
07/01/2021	1.0	Approach	Emanuele Fittipaldi
08/01/2021	1.0	Test Case - Registrazione	Fedele Mauro
08/01/2021	1.0	Test Case - Login	Giuseppe Martinelli
08/01/2021	1.0	Test Case – Pagamento	Giuseppe Martinelli
09/01/2021	1.0	Test Case – Modifica Utente	Emanuele Fittipaldi
09/01/2021	1.0	Test Case – Inserimento Quadro	Fedele Mauro
09/01/2021	1.0	Test Case – Aggiornamento Quadro	Fedele Mauro
10/01/2021	1.0	Introduction	Giusepep Martinelli
10/01/2021	1.0	Relationship to other documents	Giuseppe Martinelli
10/01/2021	1.0	System Overview	Fedele Mauro
10/01/2021	1.0	Suspenstion and resuption	Emanuele Fittipaldi
10/01/2021	1.0	Testing materials (hw/sw requirements)	Fedele Mauro
10/01/2021	1.0	Testing schedule	Fedele Mauro

Test Plan

1. INTRODUCTION

L’obiettivo che ci prefiggiamo è di condurre quanti più test possibili, in modo da massimizzare la branch coverage, e in modo da poter scovare tutte quelle variazioni dal comportamento desiderato che ci possono impedire di realizzare i requisiti funzionali e non funzionali definiti antecedentemente nel RAD. I test si estenderanno a tutte le componenti che contengono della logica o che costituiscono degli intermediari tra componenti. Non c’è bisogno di andare a testare le componenti che rappresentano le entità del nostro sistema. Per ridurre il numero dei casi di test che andremo a condurre adotteremo delle tecniche specifiche, nello specifico, category partition. Il testing sarà condotto in modo incrementale, testando dapprima le singole componenti, isolate le une dalle altre, utilizzando magari dei mockup nel caso di dipendenze da componenti non ancora sviluppate, per poi passare al test di integrazione, verificando che le componenti funzionino bene anche tra di loro ed infine condurremo il test di sistema, e con il docente, il test di accettazione.

2. RELATIONSHIP TO OTHER DOCUMENTS

Questo documento attinge ai requisiti funzionali e non funzionali del RAD, per i quali ci si è accertato del soddisfacimento, attraverso molteplici test case delle componenti coinvolte nella decomposizione del sistema in sottosistemi, proposta nell' SDD e descritta in seguito più in dettaglio nell'ODD.

3. SYSTEM OVERVIEW

Il sistema è strutturato in package, dove ogni package rappresenta un sottosistema capace di svolgere determinate azioni relative ad un determinato aspetto del sistemaper separare logicamente funzioni eterogenee tra di loro, promuovendo la coesione e riducendo l'accoppiamento.

I package sono rispettivamente:

- **Acquisto**: Questo package contiene le componenti necessarie per la gesione del carrello, permettendo azioni quali l'aggiunta di un quadro al carrello, la rimozione di un quadro dal carrello e di avere il totale del carrello.
- **ManagementOrdini**: Questo package contiene le componenti necessarie per inserire un nuovo ordine a seguito della prosecuzione nell'acquisto delle opere attraverso il pulsante di checkout.
- **ManagementQuadri**: Questo package contiene le componenti necessarie per poter effettuare le classiche operazioni CRUD su di un quadro, più quelle componenti che ci permettono di poter caricare determinati quadri, dinamicamente, sulla base del genere selezionato dall'utente nonché delle componenti che permettono di approdare sulla pagina dell'articolo per visionarlo più in dettaglio.
- **ManagementUtenti**: Questo package contiene le componenti necessarie per poter effettuare le classiche operazioni CRUD su di un utente.
- **Autenticazione**: Questo package contiene le componenti necessarie per gestire la registrazione di un nuovo utente, il login e l'area personale. Contiene inoltre le componenti che modellano l'amministratore il quale è una figura diversa dall'utente e per tale ragione, non inserita nel package "ManagementUtenti".

4. FEATURES TO BE TESTED/NOT TO BE TESTED

Le componenti che ci impegniamo a testare sono le componenti in grado di svolgere della logica di business e per le quali dunque ha senso andare ad osservarne il comportamento, fornendo una varietà di input diversi. Le componenti di interesse sono quindi tutti gli oggetti appartenenti al nostro dominio del problema, compresi i relativi metodi, più tutti gli oggetti appartenenti al dominio delle soluzioni che sono fondamentali per realizzare i requisiti funzionali esplicitati nel RAD. Non saranno testate eventuali componenti off-the-shelf in quanto si suppone siano state già ampiamente testate (es. gestione del circuito bancario) e per velocizzare la fase di testing, minimizzando i costi.

5. PASS/FAIL CRITERIA

Come criterio per determinare se un test ha avuto successo prendiamo in considerazione il caso in cui siamo riusciti ad individuare una deviazione del comportamento desiderato rispetto al comportamento osservato. Se ci troviamo in queste condizioni allora il test ha avuto successo, in quanto, lo stato di failure raggiunto ci indica la presenza di qualche fault che ha condotto il nostro sistema in uno stato erroneo.
Di contro, il test lo considereremo fallito se non si possono evincere deviazioni tra il comportamento sperato e quello osservato.

6. APPROACH

Affrontermo il testing utilizzando tenciche **Black Box,** più nello specifico category partition.

BLACKBOX - CATEGORY PARTITION:

La prima cosa che abbiamo fatto è stato individuare quali sono gli input per il nostro sistema, e una volta fatto ciò abbiamo osservato quali sono le caratteristiche di questi parametri. Alcune di queste caratteristiche sono state deducibili dalla funzione stessa che ricoprirà il dato. Per es. una input che rappresenti un anno, deve essere di quattro cifre, etc. Altre caratteristiche invece non sono state dedotte a partire dal significato del dato: es. il dato non può essere null.

Una volta individuate queste caratteristiche, abbiamo applicato dei vincoli su questi dati in input, in modo da ridurre il numero delle combinazioni e ridurre il numero dei casi da testare. Fatto ciò abbiamo combinato queste caratteristiche come in un prodotto cartesiano, in modo da rappresentare tutte le possibili combinazioni.

Queste combinazioni di parametri vengono dati in input alla componente che stiamo testando, per poi paragonare l'ouput fornito dalla componente, con l'**oracolo**, rappresentante il comportamento atteso. Se è possibile notare delle incongruenze, il test ha avuto successo nel trovare una failure e si provvederà ad individuare i fault causa di ciò. A seguito della correzione del fault, si provvederà a condurre un **test di regressione** su tutte le componenti, per verificare che le componenti già testate, continuino a funzionare correttamente anche dopo la modifica apportata alla componente malfunzionante, la quale modifica avrebbe potuto introdurre nuovi errori.

FASI DEL TESTING:

L'ordine con la quale andremo a testare le componenti è:

- Le singole classi e metodi attraverso lo **Unit testing.** Questo tipo di testing lo andremo a compiere con **JUnit**, dove saranno testati tutti i nostri Model. Si verificherà il corretto funzionamento, fornendo come input alle componenti, le diverse combinazioni individuate con il category partition. Per gli input che ci aspettiamo essere non validi, ci assicureremo della corretta segnalazione dell' errore, mediante una eccezione.
- A valle dello Unit testing, dopo che eventuali bug di ogni componente vengono rilevati e successivamente riparati, andiamo a testare ogni singolo livello seguendo un approccio di test di integrazione di tipo **bottom up**. Per permettere il testing dei Controller in isolamento, abbiamo avuto la necessità di Test driver, che abbiamo creato utilizzando il tool **Mockito**.

- Come tutti i sottosistemi interagiscono tra di loro attraverso Il **System testing.** Durante questa fase, il nostro obiettivo è quello di verificare che tutti i requisiti funzionali siano stati rispettati, che funzionino in modo corretto e soprattutto come il cliente si aspetti. A questo punto vengono anche presi in considerazione i requisiti non funzionali, dato che disponiamo dell'intero sistema. Questa fase del testing è stata condotta grazie all'utilizzo di Selenium che ci ha permesso di realizzare **testing funzionale**. I test case sono stati progettati e poi creati a partire dai requisiti funzionali e non funzionali espressi nel RAD (Requirement Analysis Document).

7. TESTING MATERIALS (HARDWARE/SOFTWARE REQUIREMENTS)

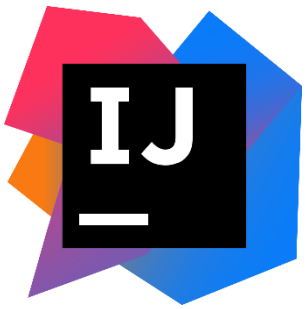
Per la fase di Testing, useremo i seguenti tool:

- **Mockito**: Ci permetterà di creare dei mockup per tutte quelle componenti che non sono state ancora implementate, ma che si rendono necessarie per poter eseguire i test case di una specifica componente.

- **JUnit**: Con JUnit andremo a realizzare materialmente i Test case descritti qui di seguito, e specificati più nel dettaglio nel Test case specification. Ci assicureremo che ogni componente, ogni metodo, ci ritorni il risultato che ci aspettiamo, e nel caso non lo faccia, ci assicuriamo del lancio delle dovute eccezioni, che possiamo gestire direttamente all'interno nel codice, o delegare questi eventi eccezionali ad una eventuale pagina di errore. JUnit lo utilizzeremo a sua volta anche per condurre l'integration testing, quando saranno disponibili le implementazioni di tutte le componenti.

- **Jacoco**: JaCoCo è stato impiegato per calcolare la percentuale di Branch-coverage dopo aver progettato ed implementato i diversi test case. Come criterio di accettazione è stata stabilita una percentuale intorno al 75%.

- **Maven**: con Maven gestiremo tutte le dipendenze per essere in grado di utilizzare i tool sopracitati, velocizzando di molto il setup iniziale che si renderà necessario per sviluppare il progetto. Per inserire le dipendenze, basta semplicemente dichiararle all'interno del file pom.xml e Maven provvederà a risolverle per noi.



- **IntelliJIDEA**: il quale costituisce il nostro ambiente di sviluppo integrato, usato congiuntamente con Maven, JUnit, Mockito e JaCoC

- **Selenium**: Selenium è stato usato per condurre il test di sistema.

8. TEST CASES

8.1. TC_LOGIN

Per questo Test Case, per ragioni di sicurezza, verifichiamo soltanto se l'e-mail è presente all'interno del database o meno, in quanto non vogliamo dare "suggetimenti" di sorta, a chi possa tentare di accedere in modo illecito alla piattaforma attraverso un qualsiasi attacco Brute Force.

Parametro: Username	
Categorie	**Scelte**
Esiste eu	1) Esiste nel database [**propertyEsisteEUok**] 2) Non esiste nel database [**Errore**]

Parametro: Password	
Categorie	**Scelte**
Esiste ep	1) Esiste nel database [**propertyEsisteEPok**] 2) Non esiste nel database [**Errore**]

Codice	Combinazione	Esito
TC_LOGIN_1	Eu2.Ep.1	Errore
TC_LOGIN_2	Eu1.Ep2	Errore
TC_LOGIN_3	Eu2.Eu2	Errore
TC_LOGIN_3	Eu1.Ep1	Success

8.2. TC_REGISTRAZIONE

L'username dell'utente dev'essere una stringa che può contenere lettere dalla a-z e dalla A-Z e anche cifre da 0-9 ed essere lunga massimo 15 caratteri almeno un carattere e non può essere una stringa vuota o contenere spazi o qualsiasi altro tipo di carattere speciale.

Parametro: Username **Formato:** [a-z,A-Z,0-9]{15}	
Categorie	**Scelte**
Lunghezza lu	1) Lunghezza <1 [errore] 2) Lunghezza >15 [errore] 3) Lunghezza >=1 && lunghezza <=15 [property **lunghezzaLUok**]
Formato fu	1) Rispetta il formato [if **lunghezzaLUok**] [property **formatoFUok**] 2) Non rispetta il formato [if **lunghezzaLUok**] [errore]

La password dev'essere una stringa che può contenere lettere dalla a-z e dalla A-Z e anche cifre 0-9 e caratteri speciali *?^=|\!"£$%&/()'+#@°ç<>,;.:-_ deve essere minore di 15 caratteri e maggiore di 8 caratteri,non può essere una stringa vuota o contenere spazi.

Parametro: Password **Formato:** [a-z,A-Z,0-9 *?^=| \!"£$%&/()'+#@°ç<>,;.:-_]{15}	
Categorie	**Scelte**
Lunghezza lp	1) Lunghezza < 8 [errore] 2) lunghezza >15 [errore] 3) lunghezza >8 && <= 15 [property **lunghezzaLPok**]
Formato fp	1) Rispetta il formato [if **lunghezzaLPok**] [property **formatoFPok**] 2) Non rispetta il formato [if **lunghezzaLPok**] [errore]

L'email deve essere una stringa che può contenere lettere dalla a-z e dalla A-Z e può contenere numeri 0-9, non può contenere caratteri speciali eccetto la @ e il . può essere lunga minimo 6 e massimo 30.

Parametro: E-mail **Formato:** [a-z A-Z 0-9 **@** a-z A-Z 0-9 **.it**]{30}	
Categorie	**Scelte**
Lunghezza le	1) Lunghezza < 6 [errore] 2) lunghezza >30 [errore] 3) lunghezza >=6 && <= 30 [property **lunghezzaLEok**]
Formato fe	1) Rispetta il formato [if **lunghezzaLEok**] [property **formatoFEok**] 2) Non rispetta il formato [if **lunghezzaLEok**] [errore]

Il nome deve essere una stringa che può contenere lettere dalla a-z e dalla A-Z, può contenere spazi. Deve essere lunga almeno 2 caratteri e massimo 30 caratteri.

Parametro: Nome **Formato:** [a-z [spazio] A-Z]{30}	
Categorie	**Scelte**
Lunghezza ln	1) Lunghezza < 2 [errore] 2) lunghezza >30 [errore] 3) lunghezza >=2 && <= 30 [property **lunghezzaLNok**]
Formato fn	1) Rispetta il formato [if **lunghezzaLNok**] [property **formatoFNok**] 2) Non rispetta il formato [if **lunghezzaLNok**] [errore]

Il cognome deve essere una stringa che può contenere lettere dalla a-z e dalla A-Z e può contenere spazi. Deve essere lunga almeno 2 caratteri e massimo 30 caratteri.

Parametro: Cognome **Formato:** [a-z [spazio]A-Z]{30}	
Categorie	**Scelte**
Lunghezza lc	4) Lunghezza < 2 [errore] 5) lunghezza >30 [errore] 6) lunghezza >=2 && <= 30 [property **lunghezzaLCok**]
Formato fc	3) Rispetta il formato [if **lunghezzaLCok**] [property **formatoFCok**] 4) Non rispetta il formato [if **lunghezzaLCok**] [errore]

La nazionalità la controlliamo semplicemente se essa è presente tra una lista di tutte le nazionalità Del mondo fornite e mantenute nel sistema.

Parametro: Nazionalità	
Categorie	**Scelte**
Esiste en	1) Esiste [**propertyEsisteENok**] 2) Non esiste [**Errore**]

Codice	Combinazione	Esito
TC_REGISTRAZIONE_1	Lu1	Errore
TC_REGISTRAZIONE_2	Lu2	Errore
TC_REGISTRAZIONE_3	Lu3.Fu2	Errore
TC_REGISTRAZIONE_4	Lu3.Fu1.Lp1	Errore

TC_REGISTRAZIONE_5	Lu3.Fu1.Lp2	Errore
TC_REGISTRAZIONE_6	Lu3.Fu1.Lp3.Fp2	Errore
TC_REGISTRAZIONE_7	Lu3.Fu1.Lp3.Fp1.Le1	Errore
TC_REGISTRAZIONE_8	Lu3.Fu1.Lp3.Fp1.Le2	Errore
TC_REGISTRAZIONE_9	Lu3.Fu1.Lp3.Fp1.Le3.Fe2	Errore
TC_REGISTRAZIONE_10	Lu3.Fu1.Lp3.Fp1.Le3.Fe1.Ln1	Errore
TC_REGISTRAZIONE_11	Lu3.Fu1.Lp3.Fp1.Le3.Fe1.Ln2	Errore
TC_REGISTRAZIONE_12	Lu3.Fu1.Lp3.Fp1.Le3.Fe1.Ln3.Fn2	Errore
TC_REGISTRAZIONE_13	Lu3.Fu1.Lp3.Fp1.Le3.Fe1.Ln3.Fn1.Lc1	Errore
TC_REGISTRAZIONE_14	Lu3.Fu1.Lp3.Fp1.Le3.Fe1.Ln3.Fn1.Lc2	Errore
TC_REGISTRAZIONE_15	Lu3.Fu1.Lp3.Fp1.Le3.Fe1.Ln3.Fn1.Lc3.Fc2	Errore
TC_REGISTRAZIONE_16	Lu3.Fu1.Lp3.Fp1.Le3.Fe1.Ln3.Fn1.Lc3.Fc1.En2	Errore
TC_REGISTRAZIONE_17	Lu3.Fu1.Lp3.Fp1.Le3.Fe1.Ln3.Fn1.Lc3.Fc1.En1	Successo

8.3. TC_IQ (INSERIMENTO QUADRO)

Il titolo deve essere una stringa che può contenere lettere dalla a-z, dalla A-Z e può essere lunga massimo 30 caratteri

Parametro: Titolo **Formato:** [a-z,A-Z, spazio,'',',-]{30}	
Categorie	**Scelte**
Lunghezza lt	**1.** lunghezza > 30 [errore] **2.** lunghezza <1 [errore] **3.** lunghezza >=1 && lunghezza <= 30 [property **lunghezzaLTok**]
Formato ft	**1.** Non rispetta il formato [if **lunghezzaLTok**] [errore] **2.** Rispetta il formato [if **lunghezzaLTok**] [property **formatoFTok**]

L'artista deve essere una stringa che può contenere lettere dalla a-z, dalla A-Z e può essere lunga massimo 20 caratteri

Parametro: Artista **Formato:** [a-z,A-Z,spazio,',-]{20}	
Categorie	**Scelte**
Lunghezza la	**1.** lunghezza > 20 [errore] **2.** lunghezza <1 [errore] **3.** lunghezza >=1 && lunghezza <= 20 [property **lunghezzaLAok**]
Formato fa	**1.** Non rispetta il formato [if **lunghezzaLAok**] [errore] **2.** Rispetta il formato [if **lunghezzaLAok**] [property **formatoFAok**]

L’anno deve essere un intero con 4 cifre

Parametro: Anno **Formato:** [0-9]{4}	
Categorie	**Scelte**
Lunghezza ly	**1.** lunghezza < 4[property **lunghezzaLY**] [errore] **2.** lunghezza > 4 [errore] **3.** lunghezza == 4 [property **lunghezzaLYok**]
Formato fy	**1.** Non rispetta il formato [if **lunghezzaLYok**] [errore] **2.** Rispetta il formato [if **lunghezzaLYok**] [property **formatoFYok**]

Il genere deve essere una stringa che può contenere lettere dalla a-z, dalla A-Z e può essere lunga massimo 20 caratteri

Parametro: Genere **Formato:** [a-z,A-Z, spazio, -]{20}	
Categorie	**Scelte**
Lunghezza lg	**1.** lunghezza > 20 [errore] **2.** lunghezza <1 [errore] **3.** lunghezza>=1 && lunghezza<= 20 [property **lunghezzaLGok**]]
Formato fg	**1.** Non rispetta il formato [if **lunghezzaLGok**] [errore] **2.** Rispetta il formato [if **lunghezzaLGok**] [property **formatoFGok**]

La tecnica deve essere una stringa che può contenere lettere dalla a-z, dalla A-Z e può essere lunga massimo 20 caratteri

Parametro: Tecnica **Formato:** [a-z,A-Z, spazio, -]{20}	
Categorie	**Scelte**
Lunghezza ltec	**1.** lunghezza > 20 [errore] **2.** lunghezza <3 [errore] **3.** lunghezza >= 2 && lunghezza<=30 [property **lunghezzaLTECok**]
Formato ftec	**1.** Non rispetta il formato [if **lunghezzaLTECok**] [errore] **2.** Rispetta il formato [if **lunghezzaLTECok**] [property **formatoFTECok**]

La dimensione deve essere formata da un minimo di 3 caratteri numerici ad un massimo di 7, rappresenta le due dimensioni dei quadri (lunghezza e altezza), sono separate da un *. Se uno dei due valori ha la prima cifra uguale a 0 genera errore poiché non esistono quadri monodimensionali

Parametro: Dimensione **Formato:** [0-9]*[0-9] {7}	
Categorie	**Scelte**
Lunghezza ld	**1.** lunghezza <3 [errore] **2.** lunghezza > 7[errore] **3**. lunghezza >=3 && lunghezza <=7 [property **lunghezzaLTECok**]

Formato fd	**1.** Non rispetta il formato [if **lunghezzaLDok**] [errore] **2.** Rispetta il formato [if **lunghezzaLDok**] [property **formatoFDok**]

Il prezzo deve avere una lunghezza minima di 3 e una massima di 15, è formata da una parte numerica che indica la parte intera e una parte che indica la parte decimale, separate da un "."

Parametro: Prezzo **Formato:** [0-9].[0.9] {15}	
Categorie	**Scelte**
Lunghezza **lp**	**1.** lunghezza <3 [errore] **2.** lunghezza > 15[errore] **3**. lunghezza >=3 && lunghezza <=15 [property **lunghezzaLPok**]
Formato fp	**1.** Non rispetta il formato [if **lunghezzaLPok**] [errore] **2.** Rispetta il formato [if **lunghezzaLPok**] [property **formatoFPok**]

L’immagine deve essere un file con formato jpg o png

Parametro: Immagine **Formato:** .jpg o .png	
Categorie	**Scelte**
Formato fi	**1.** Non rispetta il formato [errore] **2.** Rispetta il formato [property **formatoFlok**]

Codice	Combinazione	Esito
TC_IQ_1	Lt1	Errore
TC_IQ_2	Lt2	Errore
TC_IQ_3	Lt3.Ft1	Errore
TC_IQ_4	Lt3.Ft2.La1	Errore
TC_IQ_5	Lt3.Ft2.La2	Errore
TC_IQ_6	Lt3.Ft2.La3.Fa1	Errore
TC_IQ_7	Lt3.Ft2.La3.Fa2.Ly1	Errore
TC_IQ_8	Lt3.Ft2.La3.Fa2.Ly2	Errore
TC_IQ_9	Lt3.Ft2.La3.Fa2.Ly3.Fy1	Errore
TC_IQ_10	Lt3.Ft2.La3.Fa2.Ly3.Fy2.Lg1	Errore
TC_IQ_11	Lt3.Ft2.La3.Fa2.Ly3.Fy2.Lg2	Errore
TC_IQ_12	Lt3.Ft2.La3.Fa2.Ly3.Fy2.Lg3.Fg1	Errore
TC_IQ_13	Lt3.Ft2.La3.Fa2.Ly3.Fy2.Lg3.Fg2.Ltec1	Errore
TC_IQ_14	Lt3.Ft2.La3.Fa2.Ly3.Fy2.Lg3.Fg2.Ltec2	Errore
TC_IQ_15	Lt3.Ft2.La3.Fa2.Ly3.Fy2.Lg3.Fg2.Ltec3.Ftec1	Errore
TC_IQ_16	Lt3.Ft2.La3.Fa2.Ly3.Fy2.Lg3.Fg2.Ltec3.Ftec2.Ld1	Errore
TC_IQ_17	Lt3.Ft2.La3.Fa2.Ly3.Fy2.Lg3.Fg2.Ltec3.Ftec2.Ld2	Errore
TC_IQ_18	Lt3.Ft2.La3.Fa2.Ly3.Fy2.Lg3.Fg2.Ltec3.Ftec2.Ld3.Fd1	Errore

TC_IQ_19	Lt3.Ft2.La3.Fa2.Ly3.Fy2.Lg3.Fg2.Ltec3. Ftec2.Ld3.Fd2.Lp1	Errore
TC_IQ_20	Lt3.Ft2.La3.Fa2.Ly3.Fy2.Lg3.Fg2.Ltec3. Ftec2. Ld3.Fd2.Lp2	Errore
TC_IQ_21	Lt3.Ft2.La3.Fa2.Ly3.Fy2.Lg3.Fg2.Ltec3. Ftec2.Ld3.Fd2.Lp3.Fp1	Errore
TC_IQ_22	Lt3.Ft2.La3.Fa2.Ly3.Fy2.Lg3.Fg2.Ltec3. Ftec2. Ld3.Fd2.Lp3.Fp2.Fi1	Errore
TC_IQ_23	Lt3.Ft2.La3.Fa2.Ly3.Fy2.Lg3.Fg2.Ltec3. Ftec2. Ld3.Fd2.Lp3.Fp2.Fi2	Quadro inserito con successo

8.4. TC_AQ (AGGIORNAMENTO QUADRO)

Il titolo deve essere una stringa che può contenere lettere dalla a-z, dalla A-Z e può essere lunga massimo 30 caratteri

Parametro: Titolo **Formato:** [a-z,A-Z, spazio,'',',-]{30}	
Categorie	**Scelte**
Lunghezza lt	**1.** lunghezza > 30 [errore] **2.** lunghezza <1 [errore] **3.** lunghezza >=1 && lunghezza <= 30 [property **lunghezzaLTok**]
Formato ft	**1.** Non rispetta il formato [if **lunghezzaLTok**] [errore] **2.** Rispetta il formato [if **lunghezzaLTok**] [property **formatoFTok**]

L’artista deve essere una stringa che può contenere lettere dalla a-z, dalla A-Z e può essere lunga massimo 20 caratteri

Parametro: Artista **Formato:** [a-z,A-Z,spazio,’,-]{20}	
Categorie	**Scelte**
Lunghezzala	**1.** lunghezza > 20 [errore] **2.** lunghezza <1 [errore] **3.** lunghezza >=1 && lunghezza <= 20 [property **lunghezzaLAok**]
Formato fa	**1.** Non rispetta il formato [if **lunghezzaLAok**] [errore] **2.** Rispetta il formato [if **lunghezzaLAok**] [property **formatoFAok**]

L’anno deve essere un intero con 4 cifre

Parametro: Anno **Formato:** [0-9]{4}	
Categorie	**Scelte**
Lunghezzaly	**1.** lunghezza < 4[property **lunghezzaLY**] [errore] **2.** lunghezza > 4 [errore] **3.** lunghezza == 4 [property **lunghezzaLYok**]
Formato fy	**1.** Non rispetta il formato [if **lunghezzaLYok**] [errore] **2.** Rispetta il formato [if **lunghezzaLYok**] [property **formatoFYok**]

Il genere deve essere una stringa che può contenere lettere dalla a-z, dalla A-Z e può essere lunga massimo 20 caratteri

Parametro: Genere **Formato:** [a-z,A-Z, spazio, -]{20}	
Categorie	**Scelte**
Lunghezza lg	**1.** lunghezza > 20 [errore] **2.** lunghezza <8 [errore] **3.** lunghezza>=1 && lunghezza<= 20 [property **lunghezzaLGok**]]
Formato fg	**1.** Non rispetta il formato [if **lunghezzaLGok**] [errore] **2.** Rispetta il formato [if **lunghezzaLGok**] [property **formatoFGok**]

La tecnica deve essere una stringa che può contenere lettere dalla a-z, dalla A-Z e può essere lunga massimo 20 caratteri

Parametro: Tecnica **Formato:** [a-z,A-Z, spazio, -]{20}	
Categorie	**Scelte**
Lunghezza ltec	**1.** lunghezza > 20 [errore] **2.** lunghezza <3 [errore] **3.** lunghezza >= 2 && lunghezza<=30 [property **lunghezzaLTECok**]
Formato ftec	**1.** Non rispetta il formato [if **lunghezzaLTECok**] [errore] **2.** Rispetta il formato [if **lunghezzaLTECok**] [property **formatoFTECok**]

La dimensione deve essere formata da un minimo di 3 caratteri numerici ad un massimo di 7, rappresenta le due dimensioni dei quadri (lunghezza e altezza), sono separate da un *. Se uno dei due valori ha la prima cifra uguale a 0 genera errore poiché non esistono quadri monodimensionali

Parametro: Dimensione **Formato:** [0-9]*[0-9] {7}	
Categorie	**Scelte**
Lunghezza ld	**1.** lunghezza <3 [errore] **2.** lunghezza > 7[errore] **3**. lunghezza >=3 && lunghezza <=7 [property **lunghezzaLTECok**]
Formato fd	**1.** Non rispetta il formato [if **lunghezzaLDok**] [errore] **2.** Rispetta il formato [if **lunghezzaLDok**] [property **formatoFDok**]

Il prezzo deve avere una lunghezza minima di 3 e una massima di 15, è formata da una parte numerica che indica la parte intera e una parte che indica la parte decimale, separate da un “.”

Parametro: Prezzo **Formato:** [0-9].[0.9] {15}	
Categorie	**Scelte**
Lunghezza lp	**1.** lunghezza <3 [errore] **2.** lunghezza > 15[errore] **3**. lunghezza >=3 && lunghezza <=15 [property **lunghezzaLPok**]

Formato fp	**1.** Non rispetta il formato [if **lunghezzaLPok**] [errore] **2.** Rispetta il formato [if **lunghezzaLPok**] [property **formatoFPok**]

L'immagine deve essere un file con formato jpg o png

Parametro: Immagine **Formato:** .jpg o .png	
Categorie	**Scelte**
Formato fi	**1.** Non rispetta il formato [errore] **2.** Rispetta il formato [property **formatoFIok**]

Codice	Combinazione	Esito
TC_AQ_1	Lt1	Errore
TC_AQ_2	Lt2	Errore
TC_AQ_3	Lt3.Ft1	Errore
TC_AQ_4	La1	Errore
TC_AQ_5	La2	Errore
TC_AQ_6	La2.Fa1	Errore
TC_AQ_7	Ly1	Errore
TC_AQ_8	Ly2	Errore
TC_AQ_9	Ly3.Fy1	Errore
TC_AQ_10	Lg1	Errore
TC_AQ_11	Lg2	Errore
TC_AQ_12	Lg3.Fg1	Errore
TC_AQ_13	Ltec1	Errore
TC_AQ_14	Ltec2	Errore
TC_AQ_15	Ltec3.Ftec1	Errore
TC_AQ_16	Ld1	Errore

TC_AQ_17	Ld2	Errore
TC_AQ_18	Ld3.Fd1	Errore
TC_AQ_19	Lp1	Errore
TC_AQ_20	Lp2	Errore
TC_AQ_21	Lp3.Fp1	Errore
TC_AQ_22	Fi1	Errore
TC_AQ_23	Lt3.Ft2	Quadro aggiornato con Successo
TC_AQ_24	La3.Fa2	Quadro aggiornato con Successo
TC_AQ_25	Ly3.Fy2	Quadro aggiornato con Successo
TC_AQ_26	Lg3.Fg2	Quadro aggiornato con Successo
TC_AQ_27	Ltec3.Ftec2	Quadro aggiornato con Successo
TC_AQ_28	Ld3.Fd2	Quadro aggiornato con Successo
TC_AQ_29	Lp3.Fp2	Quadro aggiornato con Successo
TC_AQ_30	Fi2	Quadro aggiornato con Successo
TC_AQ_31	Lt3.Ft2.La3.Fa2.Ly3.Fy2.Lg3.Fg2.Ltec3. Ftec2. Ld3.Fd2.Lp3.Fp2.Fi2	Quadro aggiornato con Successo

*NOTA: in questo test case abbiamo considerato solo le combinazioni in cui si modifica un singolo parametro del quadro. Vanno considerate anche quelle nel caso in cui se ne va a modificare più di uno.

8.5. TC_PAG (PAGAMENTO)

Il tipo carta può essere soltato uno tra quelli esistenti e proposti, quindi facciamo un semplice controllo di questo tipo.

Parametro: Tipologia di carta	
Categorie	**Scelte**
Lunghezza Enc	1. Non esiste 2. Esiste [property **EsusteENCoK**]

Codice	Combinazione	Esito
TC_PAG_1	Enc1	Errore
TC_ PAG_2	Enc2	Successo

8.6. TC_MU (MODIFICA UTENTE)

La nazionalità la controlliamo semplicemente se essa è presente tra una lista di tutte le nazionalità Del mondo fornite e mantenute nel sistema.

Parametro: Nazionalità	
Categorie	**Scelte**
Esiste enz	3) Esiste [**propertyEsisteENZok**] 4) Non esiste [**Errore**]

Il nome deve essere una stringa che può contenere lettere dalla a-z e dalla A-Z, può contenere spazi. Deve essere lunga almeno 2 caratteri e massimo 30 caratteri.

Parametro: Nome **Formato:** [a-z [spazio] A-Z]{1,30}	
Categorie	**Scelte**
Lunghezza ln	1. Lunghezza > 30 [Errore] 2. Lunghezza => 1 && Lunghezza <= 30 [property **LunghezzaLNok**] 3. Lunghezza < 1 [Errore]
Formato fn	1. Rispetta il formato [if **LunghezzaLNok**] [property **formatoFNok**] 2. Non rispetta il formato [if **LunghezzaLNok**] [Errore]

Il cognome deve essere una stringa che può contenere lettere dalla a-z e dalla A-Z, può contenere spazi. Deve essere lunga almeno 2 caratteri e massimo 30 caratteri.

Parametro: Cognome **Formato:** [a-z [spazio] A-Z]{1,30}	
Categorie	**Scelte**
Lunghezza lc	1. Lunghezza > 30 [Errore] 2. Lunghezza => 1 && Lunghezza <= 30 [property **LunghezzaLCok**] 3. Lunghezza < 1 [Errore]
Formato fc	1. Rispetta il formato [if **LunghezzaLCok**] [property **formatoFCok**] 2. Non rispetta il formato [if **LunghezzaLCok**] [Errore]

L'email deve essere una stringa che può contenere lettere dalla a-z e dalla A-Z e può contenere numeri 0-9, non può contenere caratteri speciali eccetto la @ e il . può essere lunga minimo 6 e massimo 30.

Parametro: E-mail **Formato:** [a-z A-Z 0-9 @ a-z A-Z 0-9 .it]{1,30}	
Categorie	**Scelte**
Lunghezza le	1. Lunghezza > 30 [Errore] 2. Lunghezza => 1 && Lunghezza <= 30 [property **LunghezzaLEok**] 3. Lunghezza < 1 [Errore]
Formato fe	1. Rispetta il formato [if **LunghezzaLEok**] [property **formatoFNEok**] 2. Non rispetta il formato [if **LunghezzaLEok**] [Errore]

Codice	Combinazione	Esito
TC_MU_1	Enz1	Errore
TC_ MU _2	Enz2	Successo
TC_ MU _3	Ln1	Errore
TC_ MU _4	Ln3	Errore
TC_ MU _5	Ln2,Fn2	Errore
TC_ MU _6	Ln2,Fn1	Successo
TC_ MU _7	Lc1	Errore
TC_ MU _8	Lc3	Errore
TC_ MU _9	Lc2,Fc2	Errore
TC_ MU _10	Lc2,Fc1	Successo
TC_ MU _11	Le1	Errore
TC_ MU _12	Le2	Errore
TC_MU_13	Le3,Fe2	Errore
TC_MU_14	Le3,Fe1	Successo

9. TESTING SCHEDULE

- Completamento Unit testing 15 gennaio 2021
- Completamento Integration testing 18 gennaio 2021
- Completamente Testing di sistema 20 gennaio 2021

10. GLOSSARY

TC_LOGIN

Tests case inerenti il form di login.

TC_REGISTRAZIONE

Test case inerenti il form per la registrazione.

TC_IQ

Test case inerenti il form per l'inserimento di un nuovo quadro.

TC_AQ

Test case inerenti il form per modificare i campi di un quadro.

TC_PAG

Test case inerenti il form per convalidare l'acquisto di determinati quadri.

TC_MU

Test case inerenti il form per modificare i campi di uno specifico utente.

JaCoCo

Vai ad una descrizione più dettagliata

JUnit

Vai ad una descrizione più dettagliata

Maven

Vai ad una descrizione più dettagliata

Mockito

Vai ad una descrizione più dettagliata

Selenium

Vai ad una descrizione più dettagliata