Università degli Studi di Salerno
Corso di Ingegneria del Software

CoinVerter
Test Execution Report
Versione 1.0

Data: 01/02/2025

Coordinatore del progetto:

Nome	Matricola

Partecipanti:

Nome	Matricola
Pastore Alfredo	0512108925
Perna Alessandro	0512118963

Scritto da:	Perna Alessandro

Revision History

Data	Versione	Descrizione	Autore
01/022025	1.0	Struttura del progetto	Perna Alessandro

Indice

Sommario

1. Test di Unità

1.1. ModifyProductServletTest

1.1.1. testDoPost

- Descrizione: verifica il Redirect al form di modifica
- Input: Stringa id
- Output atteso: verifica se il prodotto inserito è nel carrello e se la quantità è giusta
- Output effettivo: il prodotto inserito è nel carrello e la quantità è giusta

1.1.2. testDoGet

- Descrizione: verifica il DoPost
- Input: request, response
- Output atteso: verifica se il DoGet chiama il DoPost e inoltra i valori di request e response
- Output effettivo: il metodo viene invocato e i valori vengono inoltrati con successo

1.2. CarrelloTest

1.2.1. testAddProductCorretto

- Descrizione: verifica l'aggiunta prodotto al carrello
- Input: intero i random compreso tra 1 e 1000
  Mock del ProductBean
- Output atteso: verifica se il prodotto inserito è nel carrello e se la quantità è giusta
- Output effettivo: il prodotto inserito è nel carrello e la quantità è giusta

1.2.2. testAddProductQuantitàErrata

- Descrizione: verifica l'aggiunta della quantità errata del prodotto al carrello
- Input: intero i = 0
  Mock del ProductBean
- Output atteso: verifica se il prodotto non è stato inserito al carrello perché la quantità è 0
- Output effettivo: il prodotto non è inserito è nel carrello

1.2.3. testAddProductProdottoErrato

- Descrizione: verifica l'aggiunta di un prodotto nullo al carrello
- Input: intero "i" random compreso tra 1 e 1000
  ProductBean = null
- Output atteso: verifica che il prodotto non è stato inserito al carrello perché il prodotto è nullo
- Output effettivo: il prodotto non è inserito è nel carrello

1.2.4. testDeleteProductCorretto

- Descrizione: verifica la rimozione di un prodotto dal carrello
- Input: intero i random compreso tra 1 e 1000
  Mock del ProductBean
- Output atteso: verifica se il prodotto inserito è stato rimosso dal carrello e se la quantità è giusta
- Output effettivo: il prodotto inserito è stato rimosso dal carrello e la quantità è giusta

1.2.5. testDeleteProductProdottoErrato

- Descrizione: verifica la rimozione di un prodotto che non è presente nel carrello
- Input: intero i random compreso tra 1 e 1000
  Mock del ProductBean
- Output atteso: il prodotto non viene rimosso dal carrello poiché non presente in quest'ultimo
- Output effettivo: il carrello non viene modificato

1.3. CartServletTest

1.3.1. testDoPostADD

- Descrizione: servlet che aggiunge i prodotti al carrello
- Input: carrello dalla sezione , id, action e quantity
- Output atteso: verifica se il carrello viene riempito correttamente dai parametri inseriti
- Output effettivo: il carrello viene riempito correttamente di parametri inseriti

1.3.2. testDoPostREMOVE

- Descrizione: servlet che rimuove i prodotti al carrello
- Input: carrello dalla sezione , id, action e quantity
- Output atteso: verifica se il prodotto viene rimosso correttamente dal carrello con i rispettivi parametri inseriti
- Output effettivo: il prodotto viene rimosso correttamente dal carrello

1.3.3. testDoGet

- Descrizione: verifica il DoPost
- Input: request, response
- Output atteso: verifica se il DoGet chiama il DoPost e inoltra i valori di request e response
- Output effettivo: il metodo viene invocato e i valori vengono inoltrati con successo

1.4. CheckoutTest

1.4.1. testDoGet

- Descrizione: verifica che il carrello venga svuotato, l'aggiunta al portafoglio dei prodotti acquistati, la ridireziona dell'utente alla pagina principale e verifica la chiamata della funzione che finalizza l'acquisto
- Input: user e cart
- Output atteso: verifica se il carrello sia svuoto, se il portafoglio è stato modificato, se l'utente è stato ridirezionato e se la funzione che finalizza l'acquisto è stata chiamata
- Output effettivo: l'output rispecchia a pieno quello atteso

1.4.2. testDoPost

- Descrizione: verifica il DoGet
- Input: request, response
- Output atteso: verifica se il DoPost chiama il DoGet e inoltra i valori di request e response
- Output effettivo: il metodo viene invocato e i valori vengono inoltrati con successo

1.5. OrderDaoDataSouceTest

1.5.1. testDoRetrieveAll

- Descrizione: verifica che il filtro vuoto restituisca tutti gli ordini dal DB
- Input: nessuno
- Output atteso: verifica se il filtro vuoto restituisca tutti gli ordini dal DB
- Output effettivo: il filtro vuoto restituisce tutti gli ordini dal DB

1.5.2. testDoRetrieveByDateFilter

- Descrizione: verifica che il filtro restituisca tutti gli ordini dal DB tra 2 date inserite
- Input: data 1, data 2
- Output atteso: verifica se il filtro restituisce tutti gli ordini dal DB tra 2 date inserite
- Output effettivo: il filtro restituisce tutti gli ordini dal DB tra le 2 date inserite

1.5.3. testDoRetrieveByNameFilter

- Descrizione: verifica che il filtro restituisca tutti gli ordini dal DB inserendo un nome
- Input: nome
- Output atteso: verifica se il filtro restituisce tutti gli ordini dal DB del nome inserito
- Output effettivo: il filtro restituisce tutti gli ordini dal DB del nome inserito

1.6. OriniServletTest

1.6.1. testDoGetAllOrders

- Descrizione: verifica che il filtro vuoto restituisca tutti gli ordini dal DB
- Input: user = null, start = null e end = null
- Output atteso: verifica se il filtro vuoto restituisca tutti gli ordini dal DB
- Output effettivo: il filtro vuoto restituisce tutti gli ordini dal DB

1.6.2. testDoGetDateFilter

- Descrizione: verifica che il filtro restituisca tutti gli ordini dal DB tra 2 date inserite
- Input: user = null, start = data1 e end = data2
- Output atteso: verifica se il filtro restituisce tutti gli ordini dal DB tra 2 date inserite
- Output effettivo: il filtro restituisce tutti gli ordini dal DB tra le 2 date inserite

1.6.3. testDoGetNameFilter

- Descrizione: verifica che il filtro restituisca tutti gli ordini dal DB inserendo un nome
- Input: user = nome
- Output atteso: verifica se il filtro restituisce tutti gli ordini dal DB del nome inserito
- Output effettivo: il filtro restituisce tutti gli ordini dal DB del nome inserito

1.6.4. testDoPost

- Descrizione: verifica il DoGet
- Input: request, response
- Output atteso: verifica se il DoPost chiama il DoGet e inoltra i valori di request e response
- Output effettivo: il metodo viene invocato e i valori vengono inoltrati con successo

1.7.ManageProductServletTest

1.7.1. testDoPostNoAdmin

- Descrizione: verifica che l'utente non è un admin
- Input: user
- Output atteso: verifica se l'utente non è un admin
- Output effettivo: l'utente non è un admin

1.7.2. testDoPostNullUser

- Descrizione: verifica gli accessi non autorizzati
- Input: user = null
- Output atteso: se l'utente non è valido si viene reindirizzati alla home page
- Output effettivo: l'utente non è valido ed è stato reindirizzato alla home page

1.7.3. testDoPostActivityNull

- Descrizione: verifica l'attività se è null ridireziona l'utente al ProdottoForm
- Input: admin user, activity = null, id
- Output atteso: verifica l'attività se è null ridireziona l'utente al ProdottoForm
- Output effettivo: l'attività è null e l'utente è stato ridirezionato al ProdottoForm

1.7.4. testDoPostActivityModifyEmpty

- Descrizione: effettua la modifica di un prodotto senza passare alcun valore
- Input: admin user, activity = modify, id=1, valore/prezzo/tipo/nome = “”
- Output atteso: la modifica non viene effettuata
- Output effettivo: la modifica non viene effettuata

1.7.5. testDoPostActivityModify

- Descrizione: effettua la modifica di un prodotto
- Input: admin user, activity = modify, id = 1, valore = 1, prezzo = 1, tipo = moneta, nome = nome, foto = mock path
- Output atteso: la modifica viene effettuata
- Output effettivo: la modifica è stata effettuata

1.7.6. testDoPostActivityAdd

- Descrizione: effettua l'aggiunta di un prodotto
- Input: admin user, activity = add, valore = 1, prezzo = 1, tipo = moneta, nome = nome, foto = mock path
- Output atteso: il prodotto viene aggiunto
- Output effettivo: il prodotto è stato aggiunto

1.7.7. testDoPostActivityRemove

- Descrizione: rimuove la disponibilità di un prodotto
- Input: admin user, activity = remove, id =1
- Output atteso: il prodotto viene reso non disponibile
- Output effettivo: il prodotto è stato reso non disponibile

1.7.8. testDoPostActivityDelete

- Descrizione: rimuove un prodotto permanentemente
- Input: admin user, activity = delete, id =1
- Output atteso: il prodotto viene rimosso permanentemente
- Output effettivo: il prodotto è stato rimosso permanentemente

1.7.9. testDoGet

- Descrizione: verifica il DoPost
- Input: request, response
- Output atteso: verifica se il DoGet chiama il DoPost e inoltra i valori di request e response
- Output effettivo: il metodo viene invocato e i valori vengono inoltrati con successo

1.8. ProductAdminServletTest

1.8.1. testDoPost

- Descrizione: verifica la restituzione di tutti i prodotti dal DB e se l'admin viene reindirizzato alla pagina gestione prodotti
- Input: user = null, start = null e end = null
- Output atteso: restituisce tutti i prodotti dal DB e l'admin viene reindirizzato alla pagina gestione prodotti
- Output effettivo: restituisce tutti i prodotti dal DB e l'admin è stato reindirizzato alla pagina gestione prodotti

1.8.2. testDoGet

- Descrizione: verifica il DoPost
- Input: request, response
- Output atteso: verifica se il DoGet chiama il DoPost e inoltra i valori di request e response
- Output effettivo: il metodo viene invocato e i valori vengono inoltrati con successo

1.9.ShopServletTest

1.9.1. testDoPostRicercaEmpty

- Descrizione: verifica il caricamento di tutti i prodotti dello shop se la ricerca è vuota
- Input: nessuno
- Output atteso: verifica il caricamento di tutti i prodotti dello shop se la ricerca è vuota
- Output effettivo: carica tutti i prodotti dello shop quando la ricerca è vuota

1.9.2. testDoPostRicercaFilter

- Descrizione: verifica il caricamento di tutti i prodotti dello shop che rispettano il filtro
- Input: filter = filter
- Output atteso: verifica il caricamento di tutti i prodotti dello shop che rispettano il filtro
- Output effettivo: carica tutti i prodotti dello shop che hanno rispettato il filtro

1.9.3. testDoPostCategoria

- Descrizione: verifica il caricamento di tutti i prodotti dello shop che rispettano la categoria
- Input: filter = filter, action = categoria
- Output atteso: verifica il caricamento di tutti i prodotti dello shop che rispettano la categoria
- Output effettivo: carica tutti i prodotti dello shop che hanno rispettato la categoria

1.9.4. testDoGet

- Descrizione: verifica il DoPost
- Input: request, response
- Output atteso: verifica se il DoGet chiama il DoPost e inoltra i valori di request e response
- Output effettivo: il metodo viene invocato e i valori vengono inoltrati con successo

1.10. AccountAdminServletTest

1.10.1. testDoPost

- Descrizione: Recupera dal database tutti gli utenti, li carica in sessione e reindirizza l'utente nella sezione di gestione account
- Input: u = mock della classe User
- Output Atteso: tutti gli utenti vengono caricati in sessione
- Output Effettivo: gli utenti vengono recuperati con successo

1.10.2. testDoGet

- Descrizione: verifica il DoPost
- Input: request, response
- Output atteso: verifica se il DoGet chiama il DoPost e inoltra i valori di request e response
- Output effettivo: il metodo viene invocato e i valori vengono inoltrati con successo

1.11. LoginTest

1.11.1. testDoPostCorrect

- Descrizione: Effettua l'autenticazione basandosi sui dati inseriti
- Input: email =”test”, pwd = “test”
- Output Atteso: L'utente viene autenticato e reidirizzato
- Output Effettivo: L'autenticazione avviene con successo

1.11.2. testDoPostEmptyEmail

- Descrizione: prova ad effettuare l'autenticazione con una mail vuota
- Input: email =””, pwd = “test”
- Output Atteso: L'utente non riesce ad autenticarsi e rimane sulla pagina di login
- Output Effettivo: L'autenticazione fallisce

1.11.3. testDoPostEmptyPwd

- Descrizione: prova ad effettuare l'autenticazione con una mail vuota
- Input: email =”test”, pwd = “”
- Output Atteso: L'utente non riesce ad autenticarsi e rimane sulla pagina di login
- Output Effettivo: L'autenticazione fallisce

1.11.4. testDoGet

- Descrizione: verifica il DoPost
- Input: request, response
- Output atteso: verifica se il DoGet chiama il DoPost e inoltra i valori di request e response
- Output effettivo: il metodo viene invocato e i valori vengono inoltrati con successo

1.12. LogoutTest

1.12.1. testDoGet

- Descrizione: effettua il logout invalidando la sessione
- Input: Session
- Output Atteso: L'utente viene scollegato e riportato alla pagina di index del sito
- Output Effettivo: La disconnessione avviene con successo

1.12.2. testDoGet

- Descrizione: verifica il DoPost
- Input: request, response
- Output atteso: verifica se il DoGet chiama il DoPost e inoltra i valori di request e response
- Output effettivo: il metodo viene invocato e i valori vengono inoltrati con successo

1.13. ManageAccountServletTest

1.13.1. doPostModifyAdmin

- Descrizione: Rimuove i privilegi di admin da un account specificato da email
- Input: Activity= "modify", Email= "test@test.com
- Output Atteso: i privilegi vengono rimossi dall'account
- Output Effettivo: i privilegi vengono rimossi con successo
- Requisiti: l'utente deve essere un admin prima dell'avvio del test

1.13.2. doPostModifyNoAdmin

- Descrizione: Aggiunge i privilegi di admin da un account specificato da email
- Input: Activity= "modify", Email= "test@test.com
- Output Atteso: i privilegi vengono aggiunti dall'account
- Output Effettivo: i privilegi vengono aggiunti con successo
- Requisiti: l'utente non deve essere un admin prima dell'avvio del test

1.13.3. doPostRemove

- Descrizione: Rimuove l'account specificato da email dal sistema
- Input: Activity= "remove", Email= "test@test.com
- Output Atteso: l'account viene rimosso
- Output Effettivo: l'account viene rimosso con successo
- Requisiti: l'utente deve essere presente nel sistema prima dell'avvio del test

1.13.4. testDoGet

- Descrizione: verifica il DoPost
- Input: request, response
- Output atteso: verifica se il DoGet chiama il DoPost e inoltra i valori di request e response
- Output effettivo: il metodo viene invocato e i valori vengono inoltrati con successo

1.14. RegisterServletTest

1.14.1. doPostCorrect

- Descrizione: registra un nuovo account nel sistema basandosi sugli input
- Input name= “name”, surname= “surname”, email= “email”, pwd= “pwd”
- Output Atteso: l’account viene creato
- Output Effettivo:la creazione avviene con successo
- Requisiti: l’utente non deve essere presente nel sistema prima dell’avvio del test

1.14.2. testDoGet

- Descrizione: verifica il DoPost
- Input: request, response
- Output atteso: verifica se il DoGet chiama il DoPost e inoltra i valori di request e response
- Output effettivo: il metodo viene invocato e i valori vengono inoltrati con successo

1.15. UserTest

1.15.1. testRicaricaPortafoglioVuoto

- Descrizione: Aggiunge una nuova valuta in un portafoglio vuoto
- Input: valuta= “moneta”, valore= 1.23
- Output Atteso: la valuta viene aggiunta al portafoglio
- Output Effettivo: l’aggiunta avviene con successo

1.15.2. testRicaricaPortafoglioNuovaValuta

- Descrizione: Aggiunge una nuova valuta in un portafoglio non vuoto
- Input: valuta= “moneta”, valore= 1.23
- Output Atteso: la valuta viene aggiunta al portafoglio
- Output Effettivo: l’aggiunta avviene con successo

2. Test di Sistema

2.1.register

- Descrizione: L’utente si registra sul sito
- Input: Nome= “Pinco”, Cognome= “pallino”, Mail= pincopallino@email.com, Password= “testpas$”
- Output Atteso: L’utente si registra con successo e viene reindirizzato alla pagina index del sito
- Output Effettivo: La registrazione va a buon fine
- Requisiti: l’utente pincopallino@email.com non deve essere presente nel sistema prima del test

2.2.incorrectLogin

- Descrizione: L'utente si autentica sul sito con credenziali errate
- Input: Mail= pincopallino@email.com, Password= "testpas"
- Output Atteso: L'utente non riesce ad autenticarsi
- Output Effettivo: L'autenticazione fallisce
- Requisiti: l'utente pincopallino@email.com deve essere presente nel sistema prima del test

2.3.CorrectLogin

- Descrizione: L'utente si autentica sul sito con credenziali corrette
- Input: Mail= pincopallino@email.com, Password= "testpas$"
- Output Atteso: L'utente riesce ad autenticarsi e viene reindirizzato
- Output Effettivo: L'autenticazione avviene con successo
- Requisiti: l'utente pincopallino@email.com deve essere presente nel sistema prima del test

2.4.Logout

- Descrizione: L'utente effettua il logout
- Output Atteso: L'utente viene scollegato con successo
- Output Effettivo: L'utente riesce a scollegarsi
- Requisiti: l'utente pincopallino@email.com deve essere autenticato prima del test

2.5.Acquisto

- Descrizione: L'utente si autentica sul sito con credenziali corrette
- Input: l'utente effettua il login, Si sposta sulla pagina dello shop, Seleziona il primo prodotto, Seleziona 50 unità e le aggiunge al carrello, Si sposta nel carrello ed effettua il checkout, Inserisce i dati della carta ed effettua il pagamento
- Output Atteso: l'acquisto viene effettuato con successo, L'utente viene reindirizzato alla pagina principale
- Output Effettivo: L'utente riesce ad effettuare l'acquisto

2.6.removeCart

- Descrizione: L'utente rimuove oggetti dal suo carrello
- Input: l'utente effettua il login, Si sposta sulla pagina dello shop, Seleziona il primo prodotto, Seleziona 5unità e le aggiunge al carrello, Si sposta nel carrello e rimuove gli oggetti
- Output Atteso: il carrello risulta vuoto
- Output Effettivo: Il carrello viene svuotato con successo

2.7.ModifyProduct

- Descrizione: L'admin modifica le proprietà di un prodotto
- Input: L'utente effettua il login con un account admin, Si sposta nella sezione di gestione prodotti, seleziona la funzione di modifica del primo prodotto, Una volta aperto il form modifica il valore "2" e il prezzo "1", Conferma le modifiche
- Output Atteso: Il prodotto viene modificato con successo
- Output Effettivo: L'admin riesce ad effettuare le modifiche

2.8. DeleteProduct

- Descrizione: L'admin rimuove un prodotto
- Input: L'utente effettua il login con un account admin, Si sposta nella sezione di gestione prodotti, seleziona la funzione di rimozione del primo prodotto
- Output Atteso: Il prodotto viene rimosso dal sistema
- Output Effettivo: L'admin riesce ad effettuare l'eliminazione con successo