Università degli Studi di Salerno
Corso di Ingegneria del Software

Mockbuster
Object Design Document
Versione 0.1

MOCKBUSTER

Data: 16/12/2024

Coordinatore del progetto:

Nome	Matricola

Partecipanti:

Nome	Matricola
Roberto Ambrosino	0512117886

Scritto da:	Roberto Ambrosino

Revision History

Data	Versione	Descrizione	Autore
16/12/2024	0.1	Prima stesura del documento	Roberto Ambrosino

Specifica delle interfacce dei sottosistemi 3

Specifica delle interfacce dei sottosistemi

- Account management
  - Registrazione

    context User::signup(email: String, password: String, name: String, surname: String, billingAddress: String): Boolean
    -- Invariante: L'email deve essere unica nel sistema.
    inv: User.allInstances()->forAll(u | u.email <> self.email)

    -- Precondizioni
    pre: not User.allInstances()->exists(u | u.email = email)
    pre: email.matches('[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,}') -- formato email valido
    pre: password.size() >= 8 and password.matches('.*[A-Z].*') and password.matches('.*[a-z].*') and password.matches('.*\\d.*') and password.matches('.*[!_$+].*') -- formato password valido
    pre: name <> '' and surname <> '' -- nome e cognome obbligatori
    pre: billingAddress <> '' -- indirizzo obbligatorio

    -- Postcondizioni
    post: User.allInstances()->exists(u | u.email = email and u.password = password and u.name = name and u.surname = surname and u.billingAddress = billingAddress)

  - Autenticazione

    context User::login(email: String, password: String): Boolean
    -- Precondizioni
    pre: User.allInstances()->exists(u | u.email = email)

    -- Postcondizioni
    post: result = User.allInstances()->exists(u | u.email = email and u.password = password)

- Catalogue management
  - Aggiunta film in catalogo

    context Catalogue::addMovie(title: String, description: String, price: Real): Boolean
    -- Precondizioni
    pre: title <> '' and description <> '' -- Titolo e descrizione non devono essere vuoti

    -- Postcondizioni
    post: Movie.allInstances()->exists(m | m.title = title and m.description = description and m.price = price)

  - Rimozione film dal catalogo

    context Catalogue::removeMovie(movie: Movie): Boolean
    -- Precondizioni
    pre: Movie.allInstances()->includes(movie) -- Il film deve essere presente nel catalogo

    -- Postcondizioni
    post: not Movie.allInstances()->includes(movie) -- Il film viene rimosso
    post: Catalogue.movies = @pre.Catalogue.movies - movie -- Gli altri film rimangono invariati

- Modifica film in catalogo

  context Catalogue::updateMovie(movie: Movie, newTitle: String, newDescription: String, newPrice: Real): Boolean
  -- Precondizioni
  pre: Movie.allInstances()->includes(movie) -- Il film deve esistere nel catalogo
  pre: newTitle <> '' and newDescription <> '' -- Il nuovo titolo e la nuova descrizione non devono essere vuoti

  -- Postcondizioni
  post: movie.title = newTitle and movie.description = newDescription and movie.price = newPrice -- I dettagli del film vengono aggiornati

- Ricerca in catalogo

  context Catalogue::search(query: String): Set(Movie)
  -- Precondizioni
  pre: query <> '' -- La query non può essere vuota

  -- Postcondizioni
  post: result = Movie.allInstances()->select(m | m.title.contains(query)

- Order management
  - Aggiungi al carrello

    context Cart::addToCart(user: User, movie: Movie): Boolean
    -- Precondizioni
    pre: not user.cart.items->includes(movie)
    pre: Catalogue.allInstances()->exists(c | c.movies->includes(movie))

    -- Postcondizioni
    post: user.cart.items->includes(movie)

    context Cart::removeFromCart(user: User, movie: Movie): Boolean
    -- Precondizioni
    pre: user.cart.items->includes(movie)

    -- Postcondizioni
    post: not user.cart.items->includes(movie)

  - Rimuovi dal carrello

    context Cart::removeFromCart(user: User, movie: Movie): Boolean
    -- Precondizioni
    pre: user.cart.items->includes(movie) -- Il film deve essere nel carrello dell'utente

    -- Postcondizioni
    post: not user.cart.items->includes(movie) -- Il film viene rimosso dal carrello
    post: user.cart.items = @pre.user.cart.items - movie -- Gli altri film nel carrello rimangono invariati

- Effettua ordine

  context Order::placeOrder(user: User): Boolean
  -- Precondizioni
  pre: user.cart.items->notEmpty()

  -- Postcondizioni
  post: Order.allInstances()->exists(o | o.user = user and o.items = user.cart.items)
  post: user.cart.items->isEmpty() -- Il carrello viene svuotato dopo l'ordine

- Visualizza ordini

  context Order::viewOrders(user: User): Set(Order)
  -- Precondizioni
  pre: Order.allInstances()->exists(o | o.user = user) -- Esistono ordini associati all'utente

  -- Postcondizioni
  post: result = Order.allInstances()->select(o | o.user = user) -- Restituisce tutti gli ordini dell'utente