Università degli Studi di Salerno
Corso di Ingegneria del Software

Pop!x
Object Design Document
Versione 1.0

Data: 15/12/2024

Coordinatore del progetto:

Nome	Matricola
Scaparra Daniele Pio	0512116260

Partecipanti:

Nome	Matricola
Scaparra Daniele Pio	0512116260
Bonagura Grazia	0512116167
Nappi Antonio	0512117391
Nardiello Raffaele	0512118666

Scritto da:	Scaparra Daniele Pio, Bonagura Grazia, Nappi Antonio, Nardiello Raffaele

Revision History

Data	Versione	Descrizione	Autore
15/12/2024	1.0	Prima versione dell'Object Design Document	Scaparra Daniele Pio, Bonagura Grazia, Nappi Antonio, Nardiello Raffaele

Indice

Interfacce delle classi

Metodo: createUser

Campo	Descrizione
Precondizioni	Il nome deve essere non vuoto (name <> ""), (email.matches("[^@]+@[^\\.]+\\..+")), (password.size() >= 8)
Postcondizioni	(result = true) implies User.allInstances()->exists(u | u.name = name and u.email = email)
Invarianti	Gli utenti devono avere univocità su email: User.allInstances()->isUnique(u | u.email)

Metodo: modifyUser

Campo	Descrizione
Precondizioni	Deve esistere un utente con l'identificativo fornito: User.allInstances()->exists(u | u.id = userId).
Postcondizioni	Se l'operazione ha successo (result = true), l'utente deve avere il nuovo nome e la nuova email: User.allInstances()->exists(u | u.id = userId and u.name = newName and u.email = newEmail).
Invarianti	Gli ordini devono rimanere associati correttamente agli utenti: Order.allInstances()->forAll(o | o.user.id = u.id).

Metodo: addProduct

Campo	Descrizione
Precondizioni	Il nome del prodotto non deve essere vuoto (name <> ""), il prezzo deve essere maggiore di zero (price > 0) lo stock deve essere maggiore/uguale a zero (stock >= 0).
Postcondizioni	Se l'operazione ha successo (result = true), il prodotto deve essere aggiunto al catalogo con i dettagli forniti: Catalog.allProducts->exists(p | p.name = name and p.price = price and p.stock = stock)
Invarianti	Catalog.allProducts->forAll(p | p.stock >= 0)

Metodo: modifyProduct

Campo	Descrizione
Precondizioni	Deve esistere un prodotto con l'identificativo fornito: Catalog.allProducts->exists(p | p.id = productId)
Postcondizioni	Se l'operazione ha successo (result = true), il prodotto deve avere i nuovi dettagli forniti: Catalog.allProducts->exists(p | p.id = productId and p.name = newName and p.price = newPrice and p.stock = newStock)
Invarianti	Order.allInstances()->forAll(o | o.items->excludes(p) implies p.id <> productId)

Metodo: deleteProduct

Campo	Descrizione
Precondizioni	Deve esistere un prodotto con l'identificativo fornito:

	Catalog.allProducts->exists(p | p.id = productId)
Postcondizioni	Se l'operazione ha successo (result = true), il prodotto deve essere rimosso dal catalogo: not Catalog.allProducts->exists(p | p.id = productId)
Invarianti	Cart.allCarts->forAll(c | c.items->excludes(p) implies p.id <> productId)

Metodo: createOrder

Campo	**Descrizione**
Precondizioni	(User.allInstances()->exists(u | u.id = userId)), ((cartItems->notEmpty())
Postcondizioni	(result = true) implies Order.allInstances()->exists(o | o.user.id = userId and o.items = cartItems)
Invarianti	Order.allInstances()->forAll(o | o.items->notEmpty())

Metodo: viewOrder

Campo	**Descrizione**
Precondizioni	Order.allInstances()->exists(o | o.id = orderId)
Postcondizioni	result = Order.allInstances()->any(o | o.id = orderId)
Invarianti	Order.allInstances()->forAll(o | User.allInstances()->exists(u | u.id = o.user.id))

Metodo: manageOrder

Campo	**Descrizione**
Precondizioni	Order.allInstances()->exists(o | o.id = orderId)
Postcondizioni	Order.allInstances()->exists(o | o.id = orderId and o.status = newStatus)

Invarianti	Order.allInstances()->forAll(o | o.status in Sequence{"Creato", "In lavorazione", "Completato"})

Metodo: addToCart

Campo	**Descrizione**
Precondizioni	(User.allInstances()->exists(u | u.id = userId)), (Catalog.allProducts->exists(p | p.id = productId and p.stock >= quantity))
Postcondizioni	Cart.allCarts->exists(c | c.user.id = userId and c.items->exists(i | i.product.id = productId and i.quantity = quantity))
Invarianti	Cart.allCarts->forAll(c | c.items->forAll(i | i.quantity <= i.product.stock))

Metodo: removeFromCart

Campo	**Descrizione**
Precondizioni	Cart.allCarts->exists(c | c.user.id = userId and c.items->exists(i | i.product.id = productId))
Postcondizioni	not Cart.allCarts->exists(c | c.user.id = userId and c.items->exists(i | i.product.id = productId))
Invarianti	Cart.allCarts->forAll(c | c.items->excludes(i | i.product.id = productId) implies i = null)

Metodo: viewCart

Campo	**Descrizione**
Precondizioni	User.allInstances()->exists(u | u.id = userId)
Postcondizioni	result = Cart.allCarts->any(c | c.user.id = userId).items

Invarianti	Order.allInstances()->forAll(o | o.user.id = c.user.id implies c.items->isEmpty())