Università degli Studi di Salerno

Corso di Ingegneria del Software

Quartum Opus
Object Design
Versione 1.0

Data: 13/10/2022

Partecipanti:

Nome	**Matricola**
Monetti Francesca Maria (FMO)	0512110434
D'Aloia Maria Caterina (MDA)	0512110344

Revision History

Data	**Versione**	**Descrizione**	**Autore**
25/06/2023	1.0	Prima stesura Object Design	FMO, MDA

Indice

1. Introduzione

   Lo scopo di questo documento è quello di rappresentare in termini di oggetti il sistema QuartumOpus che andremo a realizzare. In particolare, andiamo a rappresentare tutte le funzionalità descritte all'interno dei documenti di System Design e di Requirement Analysis.

Obiettivi di design trade-offs

Di seguito riportiamo i trade-offs e in particolare a evidenziare le scelte di design adattate per poter rendere il nostro sistema efficiente:

- Tempo di rilascio vs Qualità: Al fine di poter fornire tutti i servizi del sistema in modo corretto, si è preferito mettere in primo piano la correttezza del loro funzionamento per evitare eventuali bug, sacrificando il tempo di rilascio.
- Portabilità vs Efficienza: viene scelto Java come linguaggio essendo portabile e indipendente dalla macchina, andando a perderne di efficienza a causa delle sue prestazioni inferiori rispetto ad altri linguaggi.

Interface documentation guidelines

Variabili

I nomi delle variabili cominciano con una lettera minuscola, se presente un'altra parola allora si utilizza la struttura camelCase. Il nome, inoltre, deve essere descrittivo del loro comportamento.

Metodi

I nomi dei metodi sono della struttura tipo camelCase e indicano un verbo che rappresenta l'azione da compiere.

Classi

I nomi delle classi cominciano con una lettera maiuscola come anche le altre parole seguenti all'interno del nome. Come per le variabili, il nome deve far riferimento allo scopo delle classi.

Riferimenti

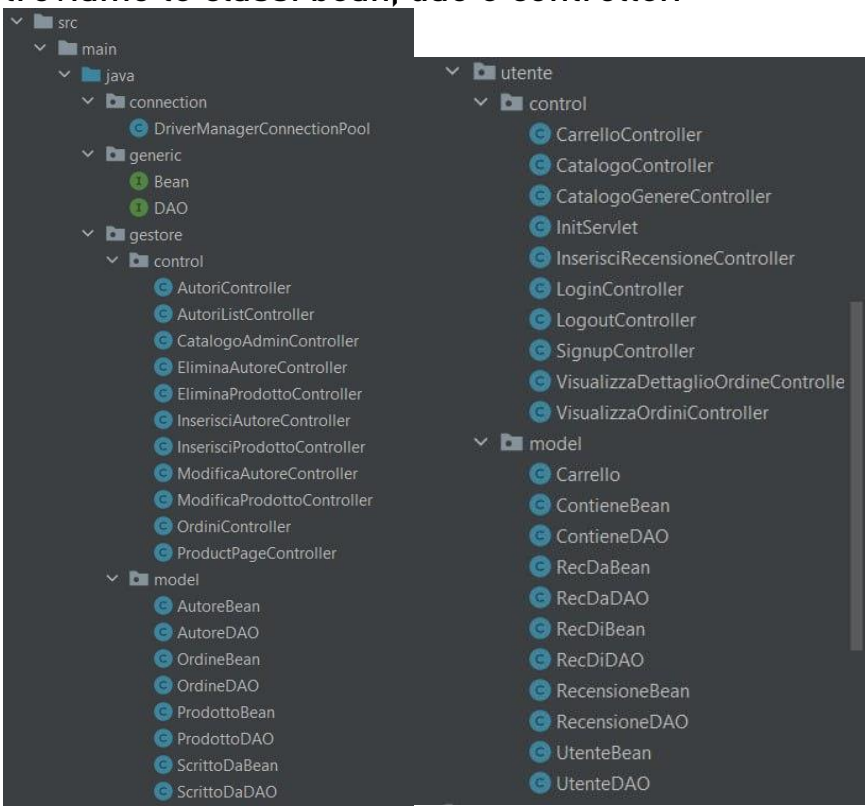
Per la stesura del documento, faremo riferimento ad alcuni punti già espressi nei documenti del RAD e del SDD.

QUARTUM OPUS
E-BOOK STORE

2. Packages

INTERFACE LAYER	L'interfaccia utente con cui l'utente interagisce per poter utilizzare il sistema.
APPLICATION LOGIC LAYER	Livello applicativo che si occupa di elaborare i dati e le richieste http dell'utente.
STORAGE LAYER	Livello che si occupa di rendere persistenti i dati e di gestirli attraverso l'utilizzo di un DBMS.

Durante la scomposizione del sistema, sono stati individuati i packages seguenti:
-utente: il package fornisce le funzionalità riguardo l'utente (registrato e guest), la loro registrazione, autenticazione, gestione della loro procedura di acquisto e visualizzazione dei loro ordini e delle loro prenotazioni. All'interno troviamo le classi bean, dao e controller.
-gestore: il package fornisce le funzionalità riguardo il catalogo, la sua gestione e la visualizzazione dei prodotti.
Il package fornisce le funzionalità riguardante gli ordini, la loro gestione e la loro visualizzazione. All'interno troviamo le classi bean, dao e controller.



3. Class Interfaces

Beans

I vari bean usati dal sistema, basati sull'interfaccia di marcatura Bean presente nel package generic

Nome Classe	AutoreBean
Modulo	gestore
Variabili d'istanza	Nome : String cognome : String codice : String datadinascita : Date
Descrizione	
Firma dei metodi	AutoreBean() getCodice():String setCodice(codice:String):void setCognome(cognome:String): void getCognome():String getNome():String setNome(nome:String):void getDatadiNascita():Date setDatadiNascita(datadinascita:Date):void
Pre e post condizioni	AutoreBean::getCodice():String Pre:true Post: result=codice Context: AutoreBean::setCodice(codice:String) Pre: true Post: this.codice=codice Context: Autorebean::setCognome(cognome:String) Pre: true Post: this.cognome=cognome Context: Autorebean::getCognome():String Pre: true Post: result=cognome Context: Autorebean::getNome():String Pre: true Post: result=nome

	Context: Autorebean::setNome(nome:String) Pre: true Post: this.nome=nome Context: Autorebean::getDatadiNascita():Date Pre: true Post: result=dataDiNascita Context: Autorebean::setDatadiNascita(datadinascita:Date): Pre: true Post: this.datadinascita=datadinascita
inviarianti	

Nome Classe	ProdottoBean
Modulo	gestore
Variabili d'istanza	ISBN : String nome : String genere : String anno : Integer edizione : Integer copertina : String casaEditrice : String prezzo : double acquistabile : Boolean link: String
Descrizione	
Firma dei metodi	ProdottoBean() getISBN():long setISBN(ISBN:long):void setNome(nome:String): void getNome():String getGenere():String setGenere(genere:String):void getAnno():int setAnno(anno:int):void getEdizione():int setEdizione(edizione:int):void getCopertina(): String setCopertina(copertina:String):void

	getCasaEditrice():String setCasaEditrice(casaEditrice:String):void getPrezzo():double setPrezzo(prezzo:double):void getAcquistabile():boolean setAcquistabile(acquistabile:boolean):void setLink(link:String): void getLink():String
Pre e post condizioni	AutoreBean::getISBN():long Pre:true Post: result=ISBN Context: ProdottoBean::setISBN(ISBN:long) Pre: true Post: this.ISBN=ISBN Context: ProdottoBean::setNome(nome:String) Pre: true Post: this.nome=nome Context: ProdottoBean::getNome():String Pre: true Post: result=nome Context: ProdottoBean::getGenere():String Pre: true Post: result=genere Context: ProdottoBean::setGenere(genere:String) Pre: true Post: this.genere=genere Context: ProdottoBean::getAnno():int Pre: true Post: result=anno Context: ProdottoBean::setAnno(anno:int) Pre: true Post: this.anno=anno Context: ProdottoBean::getEdizione() Pre: true

	Post: result=edizione Context: ProdottoBean::setEdizione(edizione:int) Pre: true Post: this.edizione=edizione Context: ProdottoBean::getCopertina() Pre: true Post: result=copertina Context: ProdottoBean::setCopertina(copertina:String) Pre: true Post: this.copertina=copertina Context: ProdottoBean::setCasaEditrice(casaEditrice:String) Pre: true Post: this.casaEditrice=casaEditrice Context: ProdottoBean::getCasaEditrice() Pre: true Post: result=casaEditrice Context: ProdottoBean::getPrezzo() Pre: true Post: result=prezzo Context: ProdottoBean::setPrezzo(prezzo:int) Pre: true Post: this.prezzo=prezzo Context: ProdottoBean::getAcquistabile() Pre: true Post: result=acquistabile Context: ProdottoBean::setAcquistabile(acquistabile:boolean) Pre: true Post: this.acquistabile=acquistabile Context: ProdottoBean::setLink(nome:String) Pre: true Post: this.link=link

	Context: ProdottoBean::getLink():String Pre: true Post: result=link
inviarianti	

Nome Classe	ScrittoDaBean
Modulo	gestore
Variabili d'istanza	prodotto : long autore : String
Descrizione	
Firma dei metodi	ScrittoDaBean() getProdotto():long setProdotto(prodotto:long):void setAutore(autore:String): void getAutore():String
Pre e post condizioni	Context: ScrittoDaBean::getProdotto():long Pre:true Post: result=prodotto Context: ScrittoDaBean::setProdotto(prodotto:long) Pre: true Post: this.prodotto=prodotto Context: ScrittoDaBean::setAutore(autore:String) Pre: true Post: this.autore=autore Context: ScrittoDaBean::getAutore():String Pre:true Post: result=autore
inviarianti	

Nome Classe	UtenteBean
Modulo	utente
Variabili d'istanza	nome : String cognome : String

	password : String mail: String gestore : boolean datadinascita : Date
Descrizione	
Firma dei metodi	UtenteBean() getNome():String setNome(nome : String):void getCognome():String setCognome(cognome:String):void getPassword():String setPassword(password:String):void getMail():String setMail(mail:String):void getGestore():boolean setGestore(gestore:boolean):void getDatadinascita():Date setDatadinascita(datadinascita:Date):void
Pre e post condizioni	Context: UtenteBean::getNome() Pre: true Post: result = nome Context: UtenteBean::setNome(nome:String) Pre: true Post: this.nome = nome Context: UtenteBean::getCognome() Pre: true Post: result = cognome Context: UtenteBean::setCognome(cognome:String) Pre: true Post: this.cognome = cognomen Context: UtenteBean::getPassword() Pre: true Post: result = password Context: UtenteBean::setPassword(password:String) Pre: true Post: this.password = password

	Context: UtenteBean::getMail() Pre: true Post: result = mail Context: UtenteBean::setMail(mail:String) Pre: true Post: this.mail= mail Context: UtenteBean::getGestore() Pre: true Post: result = gestore Context: UtenteBean::setGestore(gestore:boolean) Pre: true Post: this.gestore = gestore Context: UtenteBean::getDatadinascita() Pre: true Post: result = datadinascita Context: UtenteBean::setDatadinascita(datadinascita:String) Pre: true Post: this.datadinascita = datadinascita
inviarianti	

Nome Classe	RecensioneBean
Modulo	utente
Variabili d'istanza	idRecensione : int utente : String
Descrizione	
Firma dei metodi	RecensioneBean() getIdRecensione():int setIdRecensione(idRecensione : int):void setText(text:String):void getText():String
Pre e post condizioni	Context:RecensioneBean::getIdRecensione() Pre: true Post: result = idRecensione Context:

	RecensioneBean::setIdRecensione(idRecensione:String) Pre: true Post: this.idRecensione = idRecensione Context: RecensioneBean::getText() Pre: true Post: result = text Context: RecensioneBean::setText(text:String) Pre: true Post: this.text = text
inviarianti	

Nome Classe	RecDaBean
Modulo	utente
Variabili d'istanza	recensione:int utente:String
Descrizione	
Firma dei metodi	RecDaBean() getRecensione():int setRecensione(idRecensione:int):void getUtente():String setUtente(utente:String):void
Pre e post condizioni	Context:RecDaBean::getRecensione() Pre: true Post: result = recensione Context: RecDaBean::setRecensione(idRecensione:String) Pre: true Post: this.recensione = recensione Context: RecDaBean::getUtente() Pre: true Post: result = utente Context: RecDaBean::setUtente(utente:String) Pre: true Post: this.utente= utente
inviarianti	

Nome Classe	RecDiBean
Modulo	utente
Variabili d'istanza	recensione:int prodotto:int
Descrizione	
Firma dei metodi	RecDiBean() getRecensione():int setRecensione(idRecensione:int):void getProdotto():int setProdotto(int:prodotto):void
Pre e post condizioni	Context:RecDiBean::getRecensione() Pre: true Post: result = recensione Context: RecDiBean::setRecensione(idRecensione:String) Pre: true Post: this.recensione = recensione Context: RecDiBean::getProdotto() Pre: true Post: result = prodotto Context: RecDaBean::setProdotto(prodotto:int) Pre: true Post: this.prodotto = prodotto
inviarianti	

Nome Classe	OrdineBean
Modulo	gestore
Variabili d'istanza	id: int data: Date utente: String
Descrizione	
Firma dei metodi	OrdineBean() getUtente(): String setUtente(utente: String): void

	getData(): Date setData(data: Date): void getId(): int setId(id:int): void
Pre e post condizioni	Context:OrdineBean::getUtente() Pre: true Post: result = utente Context:OrdineBean::setUtente(utente:String) Pre: true Post: this.utente = utente Context:OrdineBean::getData() Pre: true Post: result = data Context:OrdineBean::setData(data: Date) Pre: true Post: this.data = data Context:OrdineBean::getId() Pre: true Post: result = id Context:OrdineBean::setId(id: int) Pre: true Post: this.id=id;
inviarianti	

Nome Classe	ContieneBean
Modulo	utente
Variabili d'istanza	ordine: int prodotto: long
Descrizione	
Firma dei metodi	ContieneBean() getOrdine():int setOrdine(ordine:int):void getProdotto():long setProdotto(long:prodotto):void

Pre e post condizioni	Context:ContieneBean::getOrdine() Pre: true Post: result=ordine; Context:ContieneBean::setOrdine(ordine: int) Pre: true Post: this.ordine=ordine; Context:ContieneBean::getProdotto() Pre: true Post: result=prodotto; Context:ContieneBean::setProdotto(prodotto: long) Pre: true Post: this.prodotto=prodotto;
inviarianti	

Nome Classe	Carrello
Modulo	utente
Variabili d'istanza	items: ArrayList
Descrizione	
Firma dei metodi	Carrello() addItem(): void deleteItem(): void getItems(): ArrayList deleteAll(): void getTotal(): double
Pre e post condizioni	Context:Carrello::addItem(item: ProdottoBean) Pre: not items->exists(a | a.id=item.id) Post: items->exists(a | a.id=item.id) Context:Carrello::deleteItem(item: ProdottoBean) Pre: items->exists(a | a.id=item.id) Post: not items->exists(a | a.id=item.id) Context:Carrello::getItems() Pre: true Post: result = items->select()

	Context:Carrello::deleteAll() Pre: true Post: items->size()=0 Context:Carrello::getTotal() Pre: true Post: result= totale
inviarianti	

DAOs

I vari DAO usati dal sistema basati sull'interfaccia generica DAO nel package generic

Nome Classe	AutoreDAO
Modulo	gestore
Variabili d'istanza	TABLE_NAME : String
Descrizione	
Firma dei metodi	doRetrieveAll(): ArrayList doRetrieveByKey(key:Object) : Bean doDeleteByKey(key:Object) : void doModifyByKey(key:Object) : void doInsert(bean : Bean) : void doModifyCognome(bean:Bean,cognome:String):void doModifyNome(bean:Bean,nome:String):void doModifyCodice(bean:Bean, codice:String):void doModifyDataDiNascita(bean:Bean, datadinascita:Date):void
Pre e post condizioni	AutoreDAO::doRetrieveAll() Pre: true Post: result= autore->select() Context: AutoreDAO::doRetrieveByKey(key:Object) Pre:true Post: autore->select(a.codice | a.codice= key) Context: AutoreDAO::doDeleteByKey(key:Object) Pre: true Post: not autore ->exists(a | a.codice =key) Context: AutoreDAO::doModifyByKey(key:Object):

	Pre: true Post: autore->exists(a | a.codice=key) Context: AutoreDAO::doInsert(bean:Bean) Pre: true Post: autore ->exists(a | a.codice = key) Context: AutoreDAO::doModifyCognome(bean:Bean,cognome:String): Pre: true Post: autore->exists(a | a.codice=codice and a.cognome=cognome) Context: AutoreDAO::doModifyNome(bean:Bean, nome:String) Pre: true Post: autore -> exists(a | a.codice=codice and a.nome=nome) Context: Autorebean::doModifyDatadiNascita():Date Pre: true Post: autore -> exists(a | a.codice=codice and a.datadinascita=datadinascita)
inviarianti	

Nome Classe	OrdineDAO
Modulo	gestore
Variabili d'istanza	TABLE_NAME : String
Descrizione	
Firma dei metodi	doRetrieveAll(): ArrayList doRetrieveByKey(key:Object) : Bean doDeleteByKey(key:Object) : void doModifyByKey(key:Object) : void doInsert(bean : Bean) : void doGetLatestId(): int doRetrieveByUser(): String
Pre e post condizioni	OrdineDAO::doRetrieveAll() Pre: true Post: result= ordini->select()

	Context: OrdineDAO::doRetrieveByKey(key:Object) Pre:true Post: ordini->select(a.codice | a.codice= key) Context: OrdineDAO::doDeleteByKey(key:Object) Pre: true Post: not ordini ->exists(a | a.codice =key) Context: OrdineDAO::doModifyByKey(key:Object): Pre: true Post: ordini->exists(a | a.codice=key) Context: OrdineDAO::doInsert(bean:Bean) Pre: true Post: ordini ->exists(a | a.codice = key) Context: OrdineDAO::doModifyByKey(key:Object): Pre: true Post: ordini->exists(a | a.codice=key) Context: OrdineDAO::doGetLatestId() Pre: true Post: ordini->select(a | a.max(id)) Context: OrdineDAO::doRetrieveByUser(email:String) Pre:true Post: ordini->select(a.codice | a.email=email)
inviarianti	

Nome Classe	ContieneDAO
Modulo	utente
Variabili d'istanza	TABLE_NAME : String
Descrizione	
Firma dei metodi	doRetrieveAll(): ArrayList doRetrieveByKey(key:Object) : Bean doDeleteByKey(key:Object) : void doModifyByKey(key:Object) : void

	doInsert(bean : Bean) : void doRetrieveByOrderId(id: int): ArrayList
Pre e post condizioni	ContieneDAO::doRetrieveAll() Pre: true Post: null Context: ContieneDAO::doRetrieveByKey(key:Object) Pre:true Post: null Context: ContieneDAO::doDeleteByKey(key:Object) Pre: true Post: null Context: ContieneDAO::doModifyByKey(key:Object): Pre: true Post: contiene->exists(a | a.codice=key) Context: ContieneDAO::doInsert(bean:Bean) Pre: true Post: null Context: ContieneDAO::doRetrieveByOrderId(id:int) Pre: true Post: contiene -> exists(a | a.id=id)
inviarianti	

Nome Classe	RecDADAO
Modulo	utente
Variabili d'istanza	TABLE_NAME : String
Descrizione	
Firma dei metodi	doRetrieveAll(): ArrayList doRetrieveByKey(key:Object) : Bean doDeleteByKey(key:Object) : void doModifyByKey(key:Object) : void

	doInsert(bean : Bean) : void
Pre e post condizioni	RecDaDAO::doRetrieveAll() Pre: true Post: result= recda->select() Context: RecDaDAO::doRetrieveByKey(key:Object) Pre:true Post: recda->select(a.codice | a.codice= key) Context: RecDaDAO::doDeleteByKey(key:Object) Pre: true Post: not recda ->exists(a | a.codice =key) Context: RecDaDAO::doModifyByKey(key:Object): Pre: true Post: recda->exists(a | a.codice=key) Context: RecDaDAO::doInsert(bean:Bean) Pre: true Post: recda ->exists(a | a.codice = key)
inviarianti	

Nome Classe	RecDiDAO
Modulo	utente
Variabili d'istanza	TABLE_NAME : String
Descrizione	
Firma dei metodi	doRetrieveAll(): ArrayList doRetrieveByKey(key:Object) : Bean doDeleteByKey(key:Object) : void doModifyByKey(key:Object) : void doInsert(bean : Bean) : void doRetrieveByProduct(key:long) : ArrayList
Pre e post condizioni	RecDiDAO::doRetrieveAll() Pre: true Post: result= recdi->select()

	Context: RecDiDAO::doRetrieveByKey(key:Object) Pre:true Post: recdi->select(a.codice | a.codice= key) Context: RecDiDAO::doDeleteByKey(key:Object) Pre: true Post: not recdi ->exists(a | a.codice =key) Context: RecDiDAO::doModifyByKey(key:Object): Pre: true Post: recdi->exists(a | a.codice=key) Context: RecDiDAO::doInsert(bean:Bean) Pre: true Post: recdi ->exists(a | a.codice = key) Context: RecDiDAO::doRetrieveByProduct(key:long) Pre:true Post: recdi->select(a.codice | a.codice= key)
inviarianti	

Nome Classe	RecensioneDAO
Modulo	utente
Variabili d'istanza	TABLE_NAME : String
Descrizione	
Firma dei metodi	doRetrieveAll(): ArrayList doRetrieveByKey(key:Object) : Bean doDeleteByKey(key:Object) : void doModifyByKey(key:Object) : void doInsert(bean : Bean) : void doGetLatestId() : int
Pre e post condizioni	RecensioneDAO::doRetrieveAll() Pre: true

	Post: result= recensioni->select() Context: RecensioneDAO::doRetrieveByKey(key:Object) Pre:true Post: recensioni->select(a.codice | a.codice= key) Context: RecensioneDAO::doDeleteByKey(key:Object) Pre: true Post: not recensioni ->exists(a | a.codice =key) Context: RecensioneDAO::doModifyByKey(key:Object): Pre: true Post: recensioni->exists(a | a.codice=key) Context: RecensioneDAO::doInsert(bean:Bean) Pre: true Post: recensioni->exists(a | a.codice = key) Context: RecensioneDAO::doGetLatestId() Pre: true Post: recensioni->select(a | a.max(idRecensione))
inviarianti	

Nome Classe	UtenteDAO
Modulo	utente
Variabili d'istanza	TABLE_NAME : String
Descrizione	
Firma dei metodi	doRetrieveAll(): ArrayList doRetrieveByKey(key:Object) : Bean doDeleteByKey(key:Object) : void doModifyByKey(key:Object) : void doInsert(bean : Bean) : void
Pre e post condizioni	UtenteDAO::doRetrieveAll() Pre: true Post: result= utenti->select() Context: UtenteDAO::doRetrieveByKey(key:Object)

	Pre:true Post: utenti->select(a.codice | a.codice= key) Context: UtenteDAO::doDeleteByKey(key:Object) Pre: true Post: not utenti ->exists(a | a.codice =key) Context: UtenteDAO::doModifyByKey(key:Object): Pre: true Post: utenti->exists(a | a.codice=key) Context: UtenteDAO::doInsert(bean:Bean) Pre: true Post: utenti->exists(a | a.codice = key)
inviarianti	

Nome Classe	ProdottoDAO
Modulo	gestore
Variabili d'istanza	TABLE_NAME : String
Descrizione	
Firma dei metodi	doRetrieveAll(): ArrayList doRetrieveByKey(key:Object) : Bean doDeleteByKey(key:Object) : void doModifyByKey(key:Object) : void doInsert(bean : Bean) : void doModifyISBN(prodotto:Bean, isbn:long) : void doModifyNome(prodotto:Bean, nome:String) : void doModifyGenere(prodotto:Bean, genere:String) : void doModifyAnno(prodotto:Bean, anno:int) : void doModifyEdizione(prodotto:Bean, edizione:int) : void doModifyCasaEditrice(prodotto:Bean, casaEditrice:String): void doModifyCopertina(prodotto:Bean, copertina:String) : void doModifyPrezzo(prodotto:Bean, prezzo:double) : void doRetrieveByGenre(genre:String) : ArrayList doRetrieveByNome(nome:String) : Bean

	doModifyAcquistabile(prodotto:Bean) : void
Pre e post condizioni	ProdottoDAO::doRetrieveAll() Pre: true Post: result= prodotti->select() Context: ProdottoDAO::doRetrieveByKey(key:Object) Pre:true Post: prodotti->select(a.codice | a.codice= key) Context: ProdottoDAO::doDeleteByKey(key:Object) Pre: true Post: not prodotti ->exists(a | a.codice =key) Context: ProdottoDAO::doModifyByKey(key:Object): Pre: true Post: prodotti->exists(a | a.codice=key) Context: ProdottoDAO::doInsert(bean:Bean) Pre: true Post: prodotti->exists(a | a.codice = key) Context: ProdottoDAO::doModifyISBN(prodotto:Bean, isbn:long): Pre: true Post: prodotti->exists(a | a.codice=prodotto.codice and a.isbn=isbn) Context: ProdottoDAO::doModifyNome(prodotto:Bean, nome:String): Pre: true Post: prodotti->exists(a | a.codice=prodotto.codice and a.nome=nome) Context: ProdottoDAO::doModifyGenere(prodotto:Bean, genere:String): Pre: true Post: prodotti->exists(a | a.codice=prodotto.codice and a.genere=genere) Context: ProdottoDAO::doModifyAnno(prodotto:Bean, anno:int): Pre: true

	Post: prodotti->exists(a | a.codice=prodotto.codice and a.anno=anno) Context: ProdottoDAO::doModifyEdizione(prodotto:Bean, edizione:int): Pre: true Post: prodotti->exists(a | a.codice=prodotto.codice and a.edizione=edizione) Context: ProdottoDAO::doModifyCasaEditrice(prodotto:Bean, casaEditrice: String): Pre: true Post: prodotti->exists(a | a.codice=prodotto.codice and a.casaEditrice= casaEditrice) Context: ProdottoDAO::doModifyCopertina(prodotto:Bean, copertina:String): Pre: true Post: prodotti->exists(a | a.codice=prodotto.codice and a.copertina=copertina) Context: ProdottoDAO::doModifyPrezzo(prodotto:Bean, prezzo:double): Pre: true Post: prodotti->exists(a | a.codice=prodotto.codice and a.prezzo=prezzo) Context: ProdottoDAO::doModifyAcquistabiele(prodotto:Bean, acquistabile:Boolean): Pre: true Post: prodotti->exists(a | a.codice=prodotto.codice and a.acquistabile=acquistabile) Context: ProdottoDAO::doRetrieveByGenre(genre:String) Pre:true Post: prodotti->select(a | a.genre= genre) Context: ProdottoDAO::doRetrieveByNome(nome:String) Pre:true Post: prodotti->select(a | a.nome=nome)
inviarianti	

Nome Classe	ScrittoDaDAO
Modulo	gestore
Variabili d'istanza	TABLE_NAME : String
Descrizione	
Firma dei metodi	doRetrieveAll(): ArrayList doRetrieveByKey(key:Object) : Bean doDeleteByKey(key:Object) : void doModifyByKey(key:Object) : void doInsert(bean : Bean) : void
Pre e post condizioni	UtenteDAO::doRetrieveAll() Pre: true Post: result= scrittoda->select() Context: ScrittoDaDAO::doRetrieveByKey(key:Object) Pre:true Post: scrittoda->select(a.codice | a.codice= key) Context: ScrittoDaDAO::doDeleteByKey(key:Object) Pre: true Post: not scrittoda ->exists(a | a.codice =key) Context: ScrittoDaDAO::doModifyByKey(key:Object): Pre: true Post: scrittoda->exists(a | a.codice=key) Context: ScrittoDaDAO::doInsert(bean:Bean) Pre: true Post: scrittoda->exists(a | a.codice = key)
inviarianti	

Servlet

Nome Classe	AutoriController
Modulo	gestore
Variabili d'istanza	dao: AutoreDAO coll: ArrayList
Descrizione	La classe gestisce gli autori presenti nel db.
Firma dei metodi	• context AutoriController dao : AutoreDAO = new AutoreDAO(); coll: Collection = new LinkedList<AutoreBean> dispatcher: RequestDispatcher
Pre e post condizioni	• context AutoriController::doGet(…) pre: dao <>null and coll <> null post: coll: Collection = AutoreDAO->doRetrieveAll() post: request->setAttribute("autori", coll) post: request->getRequestDispatcher("/adminPageAuthors.jsp")
inviarianti	

Nome Classe	AutoriListController
Modulo	gestore
Variabili d'istanza	dao: AutoreDAO coll: ArrayList
Descrizione	La classe gestisce gli autori presenti nel db.
Firma dei metodi	• context AutoriListController dao : AutoreDAO = new AutoreDAO(); coll: Collection = new LinkedList<AutoreBean> dispatcher: RequestDispatcher
Pre e post condizioni	• context AutoriListController::doGet(…) pre: dao <>null and coll <> null post: coll: Collection = AutoreDAO->doRetrieveAll() post: request->setAttribute("autori", coll) post: request->getRequestDispatcher("/insertProduct.jsp")
inviarianti	

Nome Classe	CatalogoAdminController
Modulo	gestore
Variabili d'istanza	dao: ProdottoDAO coll: ArrayList
Descrizione	La classe gestisce il catalogo dei prodotti.
Firma dei metodi	• context CatalogoAdminController dao : ProdottoDAO = new ProdottoDAO(); coll: Collection = new LinkedList<ProdottoBean> dispatcher: RequestDispatcher
Pre e post condizioni	• context CatalogoAdminController::doGet(…) pre: dao <>null and coll <> null post: coll: Collection = ProdottoDAO->doRetrieveAll() post: request->setAttribute("libri", coll) post: request->getRequestDispatcher("/adminPageCatalog.jsp")
inviarianti	

Nome Classe	EliminaAutoreController
Modulo	gestore
Variabili d'istanza	dao: AutoreDAO coll: ArrayList codice: String dispatcher: RequestDispatcher
Descrizione	La classe si occupa dell'eliminazione degli autori.
Firma dei metodi	• context EliminaAutoreController dao : AutoreDAO = new AutoreDAO(); coll: Collection = new LinkedList<AutoreBean> codice: String = request->getParameter("id")
Pre e post condizioni	• context EliminaAutoreController::doGet(…) pre: dao <>null and coll <> null and codice <> null post: coll: Collection = AutoreDAO ->doRetrieveAll() post: AutoreDAO ->doDeleteByKey(codice) post: request->setAttribute("autori", coll) post: request->getRequestDispatcher("/adminPageAuthors.jsp")

inviarianti	

Nome Classe	EliminaProdottoController
Modulo	gestore
Variabili d'istanza	dao: ProdottoDAO bean: ProdottoBean coll: ArrayList codice: String isbn : long dispatcher: RequestDispatcher
Descrizione	La classe si occupa dell'eliminazione dei prodotti.
Firma dei metodi	• context EliminaProdottoController dao : ProdottoDAO = new ProdottoDAO() bean : ProdottoBean = new ProdottoBean() coll: Collection = new LinkedList<AutoreBean> codice: String = request->getParameter("id") isbn: long = Long->parseLong(codice)
Pre e post condizioni	• context EliminaProdottoController::doGet(…) pre: dao <>null and coll <> null and bean <>null and isbn<> null post: ProdottoDAO->doModifyAcquistabile(bean) post: coll: Collection = ProdottoDAO->doRetrieveAll() post: request->setAttribute("libri", coll) post: request->getRequestDispatcher("/adminPageCatalog.jsp")
inviarianti	

Nome Classe	InserisciAutoreController
Modulo	gestore
Variabili d'istanza	codice: String nome:String cognome:String dataNascita:String data:LocalDate autore:AutoreBean dao:AutoreDAO

	dispatcher: RequestDispatcher
Descrizione	La classe si occupa dell'inserimento degli autori.
Firma dei metodi	• context InserisciAutoreController codice: String = request.getParameter("codice") nome: String = request.getParameter("nome") cognome: String = request.getParameter("cognome") dataNascita: String = request.getParameter("dataNascita") data: LocalDate = Local.date(dataNascita) autore: AutoreBean = new AutoreBean() dao: AutoreDAO = new AutoreDAO()
Pre e post condizioni	• context InserisciAutoreController::doPost(…) pre: dao <>null and coll <> null and codice <>null and autore <> null and nome<> null and cognome <> null and dataNascita <>null and data <> null post: AutoreBean->setCodice(codice) post: AutoreBean->setNome(nome) post: AutoreBean->setCognome(cognome) post: AutoreBean->setDatadiNascita(data) post: AutoreDAO->doInsert(autore) post: request->getRequestDispatcher("/AutoriController.jsp")
inviarianti	

Nome Classe	InserisciProdottoController
Modulo	gestore
Variabili d'istanza	isbn: long nome: String genere: String anno: int edizione: int copertina : String casaEditrice: String prezzo: double codiceAutore: String bean: ProdottoBean dao: ProdottoDAO sdao: ScrittoDaDAO

	sbean: ScrittoDaBean
Descrizione	La classe si occupa dell'inserimento dei prodotti.
Firma dei metodi	• context InserisciProdottoController isbn: Long = Long->parseLong(request->getParameter("isbn")) nome: String = request->getParameter("nome") genere: String = request->getParameter("genere") anno: int = Integer->parseInt(request->getParameter("anno")) edizione: int = Integer->parseInt(request->getParameter("edizione")) copertina : String = request->getParameter("copertina") casaEditrice : String = request->getParameter("casa") prezzo : double = Double->parseDouble(request->getParameter("Prezzo")) codice: String = request->getParameter("codice") bean: ProdottoBean = new ProdottoBean() dao : ProdottoDAO = new ProdottoDAO() sbean : ScrittoDaBean = new ScrittoDaBean() sdao : ScrittoDaDAO = new ScrittoDaDAO()
Pre e post condizioni	• context InserisciProdottoController::doPost(…) pre: isbn <> null and nome <> null and genere <> null and anno <> null and edizione <> null and copertina <> null and casaEditrice <> null and Prezzo <> null and codice <> null and bean <> null and dao <> null and sbean <> null and sdao<> null post: ProdottoBean->setIsbn(isbn) post: ProdottoBean->setNome(nome) post: ProdottoBean->setGenere(genere) post: ProdottoBean->setAnno(anno) post: ProdottoBean->setEdizione(edizione) post: ProdottoBean->setCopertina(copertina) post: ProdottoBean->setCasaEditrice(casaEditrice) post: ProdottoBean->setPrezzo(prezzo) post: ProdottoBean->setAcquistabile(true) post: ProdottoDAO->doInsert(bean) post: ScrittoDaBean->setProdotto(isbn) post: ScrittoDaBean->setAutore(codice) post: ScrittoDaDAO->doInsert(sbean) post: request->getRequestDispatcher("/CatalogoAdminController.jsp")

inviarianti	

Nome Classe	ModificaAutoreController
Modulo	gestore
Variabili d'istanza	action: String codice: String nome: String cognome: String data: Date dao: AutoreDAO bean: AutoreBean
Descrizione	La classe si occupa della modifica degli autori
Firma dei metodi	• context ModificaAutoreController action:String = request->getParameter("action") codice: String = request->getParameter("codice") dao: AutoreDAO = new AutoreDAO() bean: AutoreBean = new AutoreBean() nome: String = request->getParameter("nome") cognome: String = request->getParameter("cognome") data:Date = LocalDate->parse(request->getParameter("birthdate")
Pre e post condizioni	• context ModificaAutoreController::doPost(…) pre: action<> null and codice <> null and dao <> null and bean <> null and nome <> null and cognome <> null and data <> null post: bean: AutoreBean = AutoreDAO->DoRetrieveByKey (codice) post: if action== "modifyNome" then AutoreDAO->doModifyNome(bean:AutoreBean, nome:String) post: if action == "modifyCognome" then AutoreDAO->doModifyCognome(bean:AutoreBean, cognome:String) post: if action== "modifyData" then AutoreDao->doModifyDataNascita, bean:AutoreBean, data:Date) post: dispatcher: RequestDispatcher =

	request.getRequestDispatcher("/AutoriController");
inviarianti	

Nome Classe	ModificaProdottoController
Modulo	gestore
Variabili d'istanza	action: String codice: String nome: String genere: String anno: int edizione: int casa: String copertina: String prezzo: double dao: ProdottoDAO bean: ProdottoBean
Descrizione	La classe si occupa della modifica dei prodotti

Firma dei metodi	• context ModificaAutoreController
	action:String = request->getParameter("action")
	codice: String = request->getParameter("codice")
	dao: ProdottoDAO = new ProdottoDAO()
	bean: ProdottoBean = new ProdottoBean()
	nome: String = request->getParameter("nome")
	genere: String = request->getParameter("genere")
	anno: int: Integer->parseInt(request->getParameter("anno"));
	edizione: int: Integer->parseInt(request->getParameter("edizione"));
	casaEditrice: String = request->getParameter("casaEditrice")
	copertina: String = request->getParameter("copertina")
	prezzo: double = Double->parseDouble(request->getParameter("Prezzo")

Pre e post condizioni	• context ModificaAutoreController::doPost(…) pre: action<> null and codice <> null and dao <> null and

	bean <> null and nome <> null and genere <> null and anno <> null and edizione <> null and casaEditrice <> null and copertina <> null and prezzo <> null post: bean: ProdottoBean = ProdottoDAO->doRetrieveByKey(codice) post: if action=="modifyNome" then ProdottoDAO->ModifyNome(bean:ProdottoBean, nome:String) post: if action=="modifyAnno" then ProdottoDAO->ModifyAnno(bean:ProdottoBean, anno:int) post: if action=="modifyGenere" then ProdottoDAO->ModifyGenere(bean:ProdottoBean, genere:String) post: if action=="modifyEdizione" then ProdottoDAO->ModifyEdizione(bean:ProdottoBean, edizione:int) post: if action=="modifyCasa" then ProdottoDAO->ModifyCasa(bean:ProdottoBean, casa:String) post: if action=="modifyCopertina" then ProdottoDAO->ModifyCopertina(bean:ProdottoBean, copertina:String) post: if action=="modifyPrezzo" then ProdottoDAO->ModifyPrezzo(bean:ProdottoBean, prezzo:double) post: dispatcher : RequestDispatcher= request.getRequestDispatcher("/CatalogoAdminController");
inviarianti	

Nome Classe	OrdiniController
Modulo	gestore
Variabili d'istanza	dao : OrdineDAO coll: ArrayList
Descrizione	La classe si occupa della gestione degli ordini
Firma dei metodi	• context OrdiniController dao : OrdineDAO = new OrdineDAO() coll : ArrayList = new LinkedList<OrdineBean>
Pre e post condizioni	• context OrdiniController::doGet(…) pre: dao<>null and coll <> null post: coll:ArrayList = OrdineDAO->doRetrieveAll() post: request->setAttribute("ordini", coll) post: dispatcher : RequestDispatcher = request.getRequestDispatcher("/adminPageOrders.jsp");

inviarianti	

Nome Classe	ProductPageController
Modulo	gestore
Variabili d'istanza	isbn : long dao: ProdottoDAO libro: ProdottoBean recdidao : RecDiDAO coll1: ArrayList recensioni: ArrayList recdao : RecensioneDAO recdadao : RecDaDAO utente: UtenteDAO
Descrizione	La classe si occupa della gestione della pagina dei prodotti.
Firma dei metodi	• context OrdiniController isbn: Long->parseLong(request->getParameter("ISBN") dao : ProdottoDAO = new ProdottoDAO() libro : ProdottoBean = new ProdottoBean() recdidao = new RecDiDAO() coll1 : ArrayList= new ArrayList() recensioni: ArrayList = new ArrayList() recdao : RecensioneDAO = new RecensioneDAO() recdadao : new RecDaDAO() utente : UtenteDAO = new UtenteDAO()

Pre e post condizioni	• context OrdiniController::doGet(…)
	pre: isbn<>null and dao<>null and libro<>null and recdidao<>null and coll1<>null and recensioni<>null and recdao<>null and recdadao<>null and utente<>null
	post: libro:ProdottoBean = ProdottoDAO->doRetrieveByKey(isbn)
	post: coll1:ArrayList = RecDiDAO->doRetrieveByKey(isbn)
	post: for(RecdiBean x : coll1) recensioni->size<>0
	post: request->setAttribute("libro", libro);
	post: request->setAttribute("recensioni", recensioni);
	post: dispatcher: Request Dispatcher = request->getRequestDispatcher("/productPage.jsp");

inviarianti	

Nome Classe	CarrelloController
Modulo	utente
Variabili d'istanza	model: ProdottoDAO cart: Carrello action: String isbn: long bean:ProdottoBean user: UtenteBean ordine: OrdineBean odao: OrdineDAO id: int
Descrizione	La classe si occupa della gestione del carrello

Firma dei metodi	• context CarrelloController
	model: ProdottoDAO = new ProdottoDAO
	action: String = request->getParameter("action")
	isbn: long = Long->parseLong(request->getParameter("ISBN"));
	bean: ProdottoBean =new ProdottoBean();
	user: UtenteBean = request->getSession()->getAttribute("user");
	ordine: OrdineBean = new OrdineBean()
	od: OrdineDAO = new OrdineDAO

Pre e post condizioni	• context CarrelloController::doGet(…) pre: action <> null and cart<> null and isbn<> null and user <> null and ordine <> null post: if cart== null then cart= new Carrello() else if cart<>null then Carrello = request->getAttribute("carrello") post: if action=="addCart" and bean <> null then cart->exists(bean) post: if action=="clearCart" then cart->size=0 post: if action =="deleteCart" amd bean<>null then not cart->exists(bean) post: if action=="buy" and ordine<> null then odao-

	>doInsert(ordine) post: request->getSession()->setAttribute("carrello", cart); post: response->sendRedirect(response->encodeURL(request.getContextPath() + "/ordineEffettuato.jsp"))
inviarianti	

Nome Classe	CatalogoController
Modulo	utente
Variabili d'istanza	dao : ProdottoDAO coll: ArrayList
Descrizione	La classe si occupa della gestione del catalogo.
Firma dei metodi	• context CatalogoController dao : ProdottoDAO= new ProdottoDAO() coll: ArrayList = new ArrayList()

Pre e post condizioni	• context CatalogoController::doGet(…)
	pre: dao<> null and coll <> null
	post: coll: ArrayList = ProdottoDAO->doRetrieveAll()
	post: request->setAttribute("libri", coll);
	post: dispatcher: RequestDispatcher = request->getRequestDispatcher("/productList.jsp");

inviarianti	

Nome Classe	CatalogoGenereController
Modulo	utente
Variabili d'istanza	genere: String dao: ProdottoDAO coll:ArrayList
Descrizione	La classe si occupa della gestione dei generi dei libri,
Firma dei metodi	• context CatalogoGenereController genere: String = request.GetParameter("genere") dao: ProdottoDAO = new ProdottoDAO

	coll : ArrayList = new ArrayList()

Pre e post condizioni	• context CatalogoGenereController::doGet(…)
	pre:genere <> null and dao <>null and coll <>null
	post: coll: ProdottoDAO-> doRetrieveByGenre(genere)
	post: request->setAttribute("libri", coll);
	post: dispatcher: RequestDispatcher = request->getRequestDispatcher("/productList.jsp");

inviarianti	

Nome Classe	InitServlet
Modulo	utente
Variabili d'istanza	dao: ProdottoDAO caoll: ArrayList
Descrizione	La classe si occupa dell'inizializzazione del sito.
Firma dei metodi	• context InitServlet dao:ProdottoDAO = new ProdottoDAO() coll: ArrayList = new ArrayList()
Pre e post condizioni	• context InitServlet::init(…) pre: coll<>null and dao<>null post: coll: ProdottoDAO->doRetrieveAll() post: ServletContext=-> setAttribute("listaLibri", coll); post: super.init()
inviarianti	

Nome Classe	InserisciRecensioneController
Modulo	utente
Variabili d'istanza	recensione : String mail: String isbn: long recensione: RecensioneBean url: String
Descrizione	La classe si occupa dell'inserimento di una recensione
Firma dei	• context InserisciRecensioneController

metodi	recensione: String = request->getParameter(“testoRec”) mail: String = request.getParameter(“mail”) isbn: long= Long->parseLong(request->getParameter("ISBN"));
Pre e post condizioni	• context InserisciRecensioneController::doPost(…) pre: recensione<>null and mail <>null and isbn <> null and recensione <> null post: if recensione<>null then RecensioneDAO->doInsert(recensione) and recensioni->exists(recensione) post: request: RequestDispatcher = request->getRequestDispatcher(url);
inviarianti	

Nome Classe	LoginController
Modulo	utente
Variabili d’istanza	email: String password: String
Descrizione	La classe si occupa dell’autenticazione dell’utente
Firma dei metodi	• context LoginController email:String= request->getParameter(“email”) password: String= request->getParameter(“password”)

Pre e post condizioni	• context LoginController::doPost(…)
	pre: email<> null and password <> null
	post: if email<> null and password <> null then response->sendRedirect(response->encodeURL(request.getContextPath() + "/index.jsp"));
	post: if checkEmail(email)==false then request->setAttribute("error", true) and request->getRequestDispatcher("/login.jsp")->forward(request,response);
	post: if checkPassword(password)==false then request->setAttribute("error", true) and request-

	>getRequestDispatcher("/login.jsp")-> forward(request,response);
inviarianti	

Nome Classe	LogoutController
Modulo	utente
Variabili d'istanza	
Descrizione	La classe si occupa del logout dell'utente
Firma dei metodi	

Pre e post condizioni	• context LogoutController::doGet(…)
	pre: request->getSession() <> null
	post: request->getSession()->removeAttribute("user");
	post: response->sendRedirect(response->encodeURL (request.getContextPath() + "/index.jsp"));

inviarianti	

Nome Classe	SignupController
Modulo	utente
Variabili d'istanza	nome: String cognome: String password: String email: String data: Date udao: UtenteDAO utente:UtenteBean
Descrizione	La classe si occupa della registrazione di un nuovo utente.
Firma dei metodi	• context SignupController nome:String = request->getParameter("nome") cognome: String = request->getParameter("cognome") password: String= request->getParamter("password") email: Strihng = request->getParameter("email")

	data : LocalDate->parse(Date= request->getParameter("dataNascita"))
Pre e post condizioni	• context SignupController::doGet(…) pre: nome<>null and cognome<>null and password<>null and email<>null and data<>null post: if UtenteDAO->doRetrieveByKey(email)<>null then request->setAttribute("errorSignup", "Email già inserita!") and request->getRequestDispatcher("/signup.jsp")->forward(request, response); post: UtenteDAO->doInsert(utente)
inviarianti	

Nome Classe	VisualizzaDettaglioOrdineController
Modulo	utente
Variabili d'istanza	idOrdine:int coll: ArrayList
Descrizione	La classe si occupa nella visualizzazione dell'ordine in dettaglio.
Firma dei metodi	• context VisualizzaDettaglioOrdineController idOrdine: int= Integer->parseInt(request->getParameter("id") coll: ArrayList = new ArrayList<>()
Pre e post condizioni	• context VisualizzaDettaglioOrdineController::doGet(…) pre: idOrdine <>null and coll <>null post: coll = OrdineDAO->doRetrieveAll() post: request-> setAttribute("libri", coll) and dispatcher : RequestDispatcher = request->getRequestDispatcher("/visualizzaDettaglioOrdine.jsp");
inviarianti	

Nome Classe	VisualizzaOrdiniController
Modulo	utente
Variabili	email:String

d'istanza	coll: ArrayList
Descrizione	La classe si occupa della visualizzazione ordini.
Firma dei metodi	• context VisualizzaOrdiniController email: String = request->getParameter("email") coll : ArrayList = new ArrayList<>()

Pre e post condizioni	• context VisualizzaOrdiniController::doGet(…)
	pre: email<>null and coll <>null
	post: coll: ArrayList = OrdineDao->doRetrieveByUser(email)
	post: request->setAttribute("ordini", coll);
	post: dispatcher: RequestDispatche = request->getRequestDispatcher("/visualizzaOrdini.jsp");

inviarianti	