UNIVERSITÀ DEGLI STUDI DI SALERNO

UNIVERSITÀ DEGLI STUDI DI SALERNO
DIPARTIMENTO DI INFORMATICA

Summer Center

Object Design Document

Top manager

Professore	Tutor
De Lucia Andrea	Pecorelli Fabiano

Partecipanti

Cognome	Nome	Matricola
Dalia	Antonio	0512106118
Donnarumma	Gerardo	0512106064
Esposito	Vincenzo	0512106070

Storico versioni

Data	Versione	Descrizione	Autori
31/01/2021	1.0	Creazione del documento, dell'introduzione e dei package	Esposito
01/02/2021	1.1	Interfaccia delle classi: Utente, Genitore	Esposito
08/02/2021	1.2	Interfaccia delle classi: Bambino, Settimana, Iscrizione	Esposito
09/02/2021	1.4	Interfaccia delle classi: UtenteManage, IscrizioneManage, SettimanaManage	Esposito

Data	Versione	Descrizione	Autori
17/02/2021	1.5	Inserimento design pattern	Donnarumma
18/02/2021	1.6	Revisione generale del documento	Tutti

Sommario

Introduzione

Trade-offs di Object Design

Funzionalità vs Usabilità

Abbiamo deciso di preferire l'usabilità laddove fosse necessario in quanto l'uso della web-application deve essere il più intuitivo possibile per l'utente finale.
Le funzionalità non implementate potranno essere successivamente sviluppate e attivate in maniera graduale.

Sviluppo rapido vs Funzionalità

Abbiamo deciso di non implementare alcune funzionalità a causa delle scadenze fissate e di preferire lo sviluppo e il testing delle funzionalità ad alta priorità per rendere più fruibile il sistema per l'iscrizione ed eventualmente il pagamento.

Manutenibilità vs Performance

Abbiamo deciso di preferire la manutenibilità rispetto alle performance per rendere più semplice eventuali modifiche o aggiunta di funzionalità con il minor impatto possibile sul codice.

Linee guida per la documentazione dell'interfaccia

Affinché il codice sia facilmente leggibile e mantenibile, si è deciso di adottare la notazione CamelCase in quanto è la più familiare per gli sviluppatori coinvolti.
Esempio di nome per una variabile: nomeVariabile
Esempio di nome per una classe: NomeClasse

È consigliato l'uso della seguente forma d'uso delle parentesi graffe per la scrittura di metodi, classi, cicli ed istruzioni di controllo:

```
Nomecampo{
        //Codice
}
```

Il codice dovrà essere opportunamente indentato e l'ambiente di sviluppo Eclipse permette l'indentazione delle istruzioni.

Definizioni, acronimi e abbreviazioni

DENOMINAZIONE	FORMATO
SDD	System Design Document
ODD	Object Design Document.
JSP	JavaServer Pages

Packages

view

DENOMINAZIONE	FORMATO
index.jsp	Home del sistema
Genitore/dashboard.jsp	Dashboard del genitore
Genitore/iscrizione.jsp	Pagina del form per l'iscrizione del bambino
Genitore/dettagli_iscrizione.jsp	Pagina per visualizzare i dettagli di una singola iscrizione.
Genitore/list_iscrizioni.jsp	Pagina che mostra tutte le iscrizioni effettuate (pagate e non)
Login.jsp	Pagina che permette di effettuare il login al sistema

control.gestioneAutenticazione

DENOMINAZIONE	FORMATO
LoginControl	Gestisce il login
LogoutControl	Gestisce il logout

control.gestioneIscrizione

DENOMINAZIONE	FORMATO
IscrizioneControl	Gestisce i campi inseriti per l'iscrizione di un bambino e ne controlla il formato
SettimaneControl	Gestisce le settimane per le iscrizioni
BambinoControl	Gestisce i campi del bambino e ne controlla il formato

model.entity

DENOMINAZIONE	FORMATO
Utente	Rappresenta un utente
Genitore	Rappresenta un genitore
Iscrizione	Rappresenta un’iscrizione
Bambino	Rappresenta un bambino
Settimana	Rappresenta una settimana

model.dao

DENOMINAZIONE	FORMATO
UtenteManage	Modella specifiche interazioni degli utenti con il DB
GenitoreManage	Modella le interazioni del Genitore con il DB
IscrizioneManage	Modelle le interazioni delle Iscrizioni con il DB
BambinoManage	Modella le interazioni del Bambino con il DB
SettimanaManage	Modella le iterazioni del Settimane con il DB

Interfaccia delle classi

Utente

Nome	**Utente**
Descrizione	Rappresenta un utente registrato nel sistema
Attributi	-nome: String -cognome: String -email: String -codiceFiscale: String -dataDiNascita: Date -genere: char -password: String -ruolo: String -numTelefono: String
Signature dei metodi	+getNome() +setNome(String nome) +getCognome() +setCognome(String cognome) +getEmail() +setEmail(String email) +getCF() +setCF(String codiceFiscale) +getDataDiNascita() +setDataDiNascita(Date dataDiNascita) +getRuolo() +setRuolo(char ruolo) +getPassword() +setPassword (String password) +getNumTelefono() +setNumTelfono(String numTelefono)
Pre-condizioni	**context** Utente::setPassword(p) **pre**: p.matches('^(?=.*[0-9])(?=.*[a-zA-Z])([a-zA-Z0-9]+){8,64}$')
	context Utente::setNome(n) **pre**: n.matches('^[A-Za-z]{3,}$')
	context Utente::setCognome(c) **pre**: c.matches('^[A-Za-z]{3,}$')
	context Utente::setEmail(e) **pre**: e.matches('\S+@\S+\.\S+')
	context Utente::setNumTelefono(t) **pre**: t.matches('^[0-9]{8,16}$')

	context Utente::setCodiceFiscale(cf) **pre**: cf.matches('^[a-zA-Z]{6}[0-9]{2}[abcdehlmprstABCDEHLMPRST]{1}[0-9]{2}([a-zA-Z]{1}[0-9]{3})[a-zA-Z]{1}$')
	context Utente::setDataNascita(dataNascita) **pre**: dataNascita.lenght>8 and dataNascita.lenght<16 and dataNascita.matches('^(0?[1-9]|[12][0-9]|3[01])[/](0?[1-9]|1[012])[/]\d{4}$') and dataNascita<now()
Post-condizioni	
Invarianti	

Genitore

Nome	**Genitore**
Descrizione	Rappresenta un genitore registrato nel sistema
Attributi	-luogoNascita: String -indirizzo: String -città: String -provincia: String -cap: Intenger -professione: String -numTelefonoSecondario: String -documentoIdentita: String -consensoInfo: Boolean -consensoImgVideo: Boolean -delega: String -infoFamiliari: String
Signature dei metodi	+getLuognoNascita() +setLuogoNascita(String luogoNascita) +getIndirizzo() +setIndirizzo(String indirizzo) +getCittà() +setCittà(String città) +getProvincia() +setProvincia(String provincia) +getCap() +setCap(int cap) +getProfessione() +setProfessione(String professione) +getNumTelefonoSecondario() +setNumTelefonoSecondario(String numTelefonoSecondario) +getDocumentoIdentita() +setDocumentoIdentita(String documentoIdentita) +getConsensoInfo() +setConsensoInfo(Boolean consensoInfo) +getConsensoImgVideo() +setConsensoImgVideo(Boolean consensoImgVideo) +getDelega() +setDelega(String delega) +getInfoFamiliari()

	+setInfoFamiliari (String infoFamiliari)
Pre-condizioni	**context** Genitore::setLuogoNascita(ln) **pre**: ln.matches('^[a-zA-Z 0-9,]{3,}$')
	context Genitore::setIndirizzo(i) **pre**: i.matches('^[a-zA-Z 0-9,]{3,}$')
	context Genitore::setCitta(c) **pre**: c.matches('^[a-zA-Z]{3,}$')
	context Genitore::setProvincia(p) **pre**: p.matches('^[a-zA-Z]{2,}$')
	context Genitore:: setCap(a) **pre**: a.length = 5 and a.matches('^[0-9]{5}$')
	context Genitore::setProfessione (p) **pre**: p.matches('^[a-zA-Z,]{5,}$')
	context Genitore::setDelega (d) **pre**: d.matches('^[a-zA-Z,]{5,}$')
	context Genitore::setInfoFamiliari(info) **pre**: info.matches('^[a-zA-Z,]{3,}$')
	context Genitore::setNumTelefonoSecondario(t) **pre**: t.matches('^[0-9]{8,16}$')
	context Genitore::setNumTelefonoSecondario(t) **pre**: t.matches('^[0-9]{8,16}$')
Post-condizioni	
Invarianti	

Bambino

Nome	**Bambino**
Descrizione	Rappresenta un bambino iscritto nel sistema
Attributi	-nome: String -cognome: String -dataDiNascita: Date -luogoNascita: String -codiceFiscale: String -genere: String -eta: Integer -esigenzeAlimentari: Boolean -disabilità: Boolean -infoEsigenzeAlimentari: String -ausilioMaterialeGalleggiante: Boolean -farmaci: String -allergie: String -documentoIdentità: String -certificatoMedico: String
Signature dei metodi	+getNome() +setNome(String nome) +getCognome() +setCognome(String cognome) +getDataDiNasciata() +setDataDiNasciata(Date dataDiNascita) +getLuognoNascita() +setLuogoNascita(String luogoNascita) +getCF() +setCF(String codiceFiscale) +getGenere() +setGenere(String genere) +getEsigenzeAlimentari() +setEsigenzeAlimentari(Boolean esigenzeAlimentari) +getDisabilità() +setDisabilità(Boolean disabilità) +getDescrizioneEsigenzeAlimentari() +setDescrizioneEsigenzeAlimentari(String descrizioneEsigenzeAlimentari) +getMaterialeGalleggiante()

	+setMaterialeGalleggiante(Boolean materialeGalleggiante) +getFarmaci() +setFarmaci(String farmaci) +getAllergie() +setAllergie(String allergie) +getDocumentoIdentità() +setDocumentoIdentità(String documentoIdentità) +getCertificatoMedico () +setCertificatoMedico (String certificatoMedico)
Pre-condizioni	**context** Bambino::setNome(n) **pre**: n.matches(‘^[A-Za-z]{3,}$’)
	context Bambino::setCognome(c) **pre**: c.matches(‘^[A-Za-z]{3,}$’)
	context Bambino::setDataNascita(dataNascita) **pre**: dataNascita.lenght>8 and dataNascita.lenght<16 and dataNascita.matches(‘^(0?[1-9]|[12][0-9]|3[01])[/](0?[1-9]|1[012])[/]\d{4}$’) and dataNascita<now()
	context Bambino::setCodiceFiscale(cf) **pre**: cf.matches(‘^[a-zA-Z]{6}[0-9]{2}[abcdehlmprstABCDEHLMPRST]{1}[0-9]{2}([a-zA-Z]{1}[0-9]{3})[a-zA-Z]{1}$’)
	context Bambino::setFarmaci(f) **pre**: f.matches("^[A-Za-z 0-9]{0,}$")
	context Bambino::setAllergie(a) **pre**: a.matches("^[A-Za-z 0-9]{0,}$")
	context Bambino::setInfoEsigenzeAlimentari(info) **pre**: info.matches("^[A-Za-z 0-9]{0,}$")
	context Bambino::setDocumentoIdentità (di) **pre**: di.matches("([a-zA-Z0-9\\s_\\\\.\\-\\(\\):])+(.jpeg|.png|.pdf)$")) && di!=null
	context Bambino::setCertificatoMedico (cm) **pre**: cm.matches("([a-zA-Z0-9\\s_\\\\.\\-\\(\\):])+(.jpeg|.png|.pdf)$")) && cm!=null
	context Bambino::setLuogoNascita(ln) **pre**: ln.matches(‘^[a-zA-Z]{3,}$’)
Post-condizioni	
Invarianti	

Iscrizione

Nome	**Iscrizione**
Descrizione	Rappresenta un'iscrizione nel sistema
Attributi	-idIscrizione: Integer -dataIscrizione: Date -qrCode: String -prezzo: float -richiestaDisdetta: Boolean -rimborsoDisdetta: float -servizioSportivo: Boolean -tipoSoggiorno: String
Signature dei metodi	+getIdIscrizione() +setIdIscrizione(int idIscrizione) +getDataIscrizione() +setDataIscrizione(Date dataIscrizione) +getQrCode() +setQrCode(String qrCode) +getPrezzo() +setPrezzo(float prezzo) +getRichiestaRimborso() +setRichiestaRimborso(Boolean richiestaRimborso) +getRimborsoDisdetta () +setRimborsoDisdetta(float rimborsoDisdetta) +getServizioSportivo() +setServizioSportivo(Boolean servizioSportivo) +getTipoSoggiorno() +setTipoSoggiorno(String tipoSoggiorno)
	context Settimana::setDataIscrizione(dataIscrizione) **pre**: dataIscrizione.lenght>8 and dataIscrizione.lenght<16 and dataIscrizione.matches('^(0?[1-9]|[12][0-9]|3[01])[/](0?[1-9]|1[012])[/]\d{4}$') and dataIscrizione<now()
	context Settimana::setQrCode(qrCode) **pre**: qrCode.matches("^[A-Za-z 0-9]{2,}$")
	context Settimana::setPrezzo(p) **pre**: p.matches('^[0-999,]*$')
	context Settimana::setRimborsoDisdetta(r) **pre**: r.matches('^[0-999,]*$')

	context Settimana::setTipoSoggiorno(tipoSoggiorno) **pre**: tipoSoggiorno.matches(‘^[A-Za-z -]{9,9}$’)
Post-condizioni	
Invarianti	

Settimana

Nome	**Settimana**
Descrizione	Rappresenta una settimana nel sistema
Attributi	-idSettimana: Integer -dataInizio: Date -dataFine: Date -disponibilita: Integer
Signature dei metodi	+getIdSettimana() +setIdSettimana(int idSettimana) +getDataInizio() +setDataInzio(Date dataInizio) +getDataFine() +setDataFine(Date dataFine) +getDisponibilita() +setDisponibilita(int disponibilita)
	context Settimana::setDataInizio(dataInizio) **pre**: dataInizio.lenght>8 and dataInizio.lenght<16 and dataInzio.matches('^(0?[1-9]|[12][0-9]|3[01])[/](0?[1-9]|1[012])[/]\d{4}$')
	context Settimana::setDataFine(dataFine) **pre**: dataFine.lenght>8 and dataFine.lenght<16 and dataFine.matches('^(0?[1-9]|[12][0-9]|3[01])[/](0?[1-9]|1[012])[/]\d{4}$') and dataInizio<dataFine
Post-condizioni	
Invarianti	

UtenteManage

Nome	**UtenteManage**
Descrizione	Modella le interazioni dell'Utente col Database
Attributi	-ds: DataSource -TABLE_NAME: String -em: EntityManager
Signature dei metodi	-save(Utente u) +getUserIfExists(String email, String password)
Pre-condizioni	**context** UtenteManage::save(u) **pre**: u == null && u.getNome().matches("^[A-Za-z]{3,}$") && u.getCognome().matches("^[A-Za-z]{3,}$") && u.getCodiceFiscale().matches("^[a-zA-Z]{6}[0-9]{2}[abcdehlmprstABCDEHLMPRST]{1}[0-9]{2}([a-zA-Z]{1}[0-9]{3})[a-zA-Z]{1}$") && u.getEmail().matches("\\S+@\\S+\\.\\S+") && u.getNumTelefono().matches("^[0-9]{8,16}$") && u.getPassword().matches("^(?=.*[0-9])(?=.*[a-z])(?=.*[A-Z])(?=.*[@#$%^&+=])(?=\\S+$).{8,}$") && u.getEta()>=18
	context UtenteManage::getUserIfExists(String email, String password) **pre**: email.matches("\\S+@\\S+\\.\\S+") && password.matches("^(?=.*[0-9])(?=.*[a-z])(?=.*[A-Z])(?=.*[@#$%^&+=])(?=\\S+$).{8,}$")
Post-condizioni	**context** UtenteManage::save(u) **post**: u!=null
	context UtenteManage::getUserIfExists(u) **post**: u!=null
Invarianti	

BambinoManage

Nome	**BambinoManage**
Descrizione	Modella le interazioni del Bambino col Database
Attributi	-ds: DataSource -TABLE_NAME: String -em: EntityManager
Signature dei metodi	+save(Bambino b); +update(Bambino b); +List<Bambino> getBambini(String cfGenitore); +getBambino(String codiceFiscale);
Pre-condizioni	**context** BambinoManage::save(b) **pre**: b==null && b.getNome().matches("^[A-Za-z]{3,}$") && b.getCognome().matches("^[A-Za-z]{3,}$") && b.getCodiceFiscale().matches("^[a-zA-Z]{6}[0-9]{2}[abcdehlmprstABCDEHLMPRST]{1}[0-9]{2}([a-zA-Z]{1}[0-9]{3})[a-zA-Z]{1}$") && (b.getEta()<=18) && b.getLuogoNascita().matches("^[A-Za-z]{3,}$") && b.getCertificatoMedico().matches("([a-zA-Z0-9\\s_\\\\.\\-\\(\\):])+(.jpeg|.png|.pdf)$")) && b.getDocumentoIdentita().matches("([a-zA-Z0-9\\s_\\\\.\\-\\(\\):])+(.jpeg\|.png\|.pdf)$")) && b.getInfoEsigenzeAlimentari().matches("^[A-Za-z 0-9]{2,}$")) && b.getFarmaci().matches("^[A-Za-z 0-9]{2,}$")) && b.getInfoAllergie().matches("^[A-Za-z 0-9]{2,}$")) && b.getTagliaIndumenti().matches("^[A-Za-z]{1,4}$"))
	context BambinoManage::update(b) **pre**: b==null && b.getNome().matches("^[A-Za-z]{3,}$") && b.getCognome().matches("^[A-Za-z]{3,}$") && b.getCodiceFiscale().matches("^[a-zA-Z]{6}[0-9]{2}[abcdehlmprstABCDEHLMPRST]{1}[0-9]{2}([a-zA-Z]{1}[0-9]{3})[a-zA-Z]{1}$") && (b.getEta()<=18) && b.getLuogoNascita().matches("^[A-Za-z]{3,}$") && b.getCertificatoMedico().matches("([a-zA-Z0-9\\s_\\\\.\\-\\(\\):])+(.jpeg|.png|.pdf)$")) && b.getDocumentoIdentita().matches("([a-zA-Z0-9\\s_\\\\.\\-\\(\\):])+(.jpeg\|.png\|.pdf)$")) && b.getInfoEsigenzeAlimentari().matches("^[A-Za-z 0-9]{2,}$"))

	&& b.getFarmaci().matches("^[A-Za-z 0-9]{2,}$")) && b.getInfoAllergie().matches("^[A-Za-z 0-9]{2,}$")) && b.getTagliaIndumenti().matches("^[A-Za-z]{1,4}$"))
Post-condizioni	**context** BambinoManage::save(b) **post**: b!=null
	context BambinoManage::update(b) **post**: b!=null
	context BambinoManage::getBambino(codiceFiscale) **post**: b!=null
Invarianti	

IscrizioneManage

Nome	**IscrizioneManage**
Descrizione	Modella le interazioni delle Iscrizioni col Database
Attributi	-ds: DataSource -TABLE_NAME: String -em: EntityManager
Signature dei metodi	+save(Iscrizione i) +List<Iscrizione> getIscrizioniByGenitore(String cfGenitore); +List<Iscrizione> getAll();
Pre-condizioni	**context** IscrizioneManage::save(i) **pre**: i==null && i.getQrCode().matches("^[A-Za-z 0-9]{2,}$") && i.getTipoSoggiorno().matches("^[A-Za-z -]{9,9}$") && i.getSettimane()!=null
Post-condizioni	**context** IscrizioneManage::save(i) **post**: i!=null
Invarianti	

SettimanaManage

Nome	**SettimanaManage**
Descrizione	Modella le interazioni della Settimana col Database
Attributi	-ds: DataSource -TABLE_NAME: String -em: EntityManager
Signature dei metodi	+Settimana getSettimana(Settimana s) +List<Settimana> getSettimaneDisponibili()
Pre-condizioni	
Post-condizioni	
Invarianti	

Design Pattern

Il Repository Pattern definisce un'interfaccia detta Repository, la quale specifica tutte le operazioni di lettura e scrittura logica per un'entità specifica. L'interfaccia viene implementata da una o più classi che forniscono le implementazioni dei metodi dell'interfaccia.

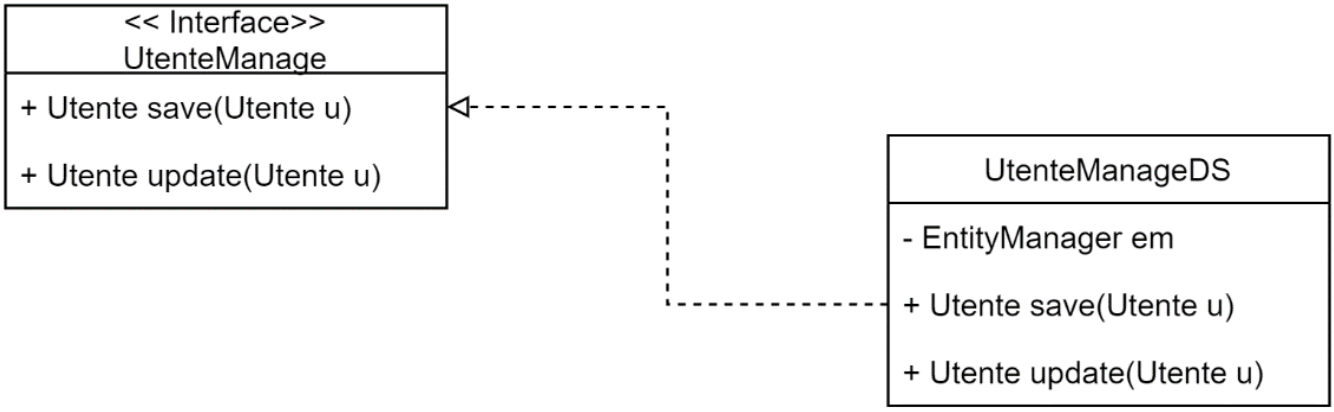
L'obiettivo di questo pattern è quello di separare il codice per la persistenza dal codice di business e a migliorare la riusabilità del codice relativo alla persistenza.

Utilizzando questo pattern abbiamo il vantaggio isolare gli oggetti del dominio dai dettagli del codice di accesso al database e ridurre al minimo la dispersione e la duplicazione del codice di query.

Nel nostro software un Repository viene implementato utilizzando JPA e Hibernate. JPA è specifica Java per mappare un POJO su un database.

Il vantaggio offerto da JPA consiste nell'abilitare un mapping oggetti-relazioni attraverso annotazioni o XML standard, definendo come avviene il mapping tra classi Java e tabelle di un database relazionale grazie all'utilizzo dell'ORM (Object Relational Mapping). JPA definisce inoltre le API di un EntityManager, il cui scopo è la gestione a runtime di queries e transazioni su oggetti resi persistenti.

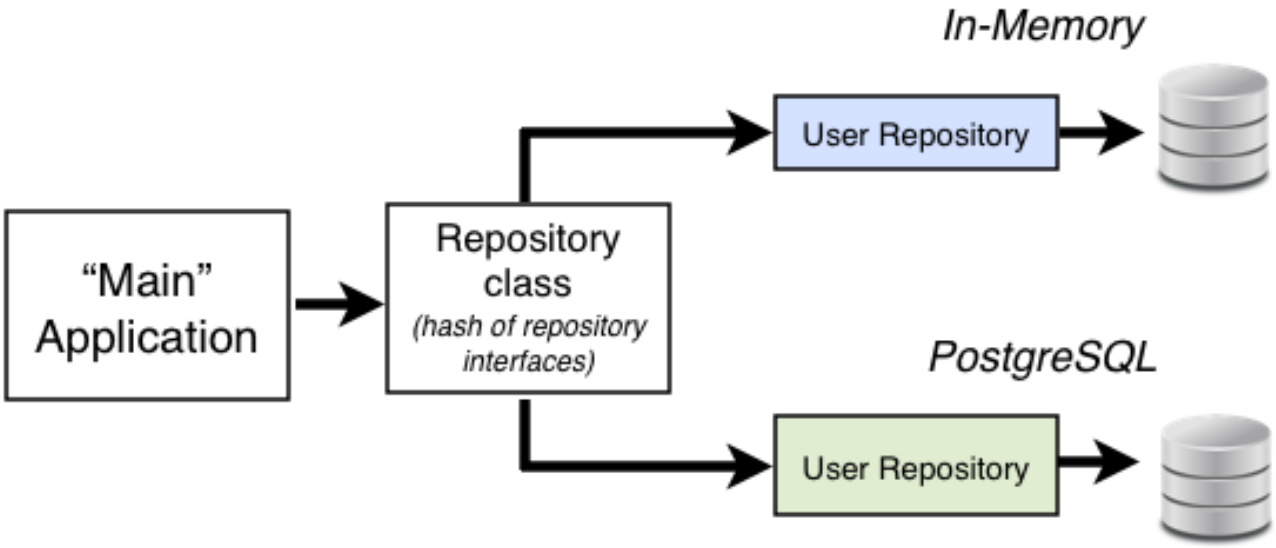
Un esempio del Pattern nella nostra implementazione:



Nel Pattern solitamente, l'interfaccia UtenteManage è chiamata UtenteRepository e l'implementazione UtenteManageDS è chiamata UtenteRepositoryImp, ma è solo una convenzione, il concetto resta tale.

L'UtenteControl invia un messaggio all'UtenteManage per richiedere il salvataggio dei dati di un Utente. L'UtenteManageDS utilizza le API dell'EntityManager per effettuare le operazioni sul database, senza utilizzare il linguaggio di query specifico della tecnologia di persistenza utilizzata. Molto utile nel caso in cui stessimo effettuando operazioni su DBSM differenti che utilizzando una propria traduzione di SQL.

Ecco un esempio esemplificativo che fa capire al meglio il concetto:



La "Main" Application invia un messaggio al Repository per effettuare le operazioni su un database. In questo esempio le operazioni vengono effettuate su un PostgreSQL e in Memoria (per effettuare anche test simulando un database in memoria).