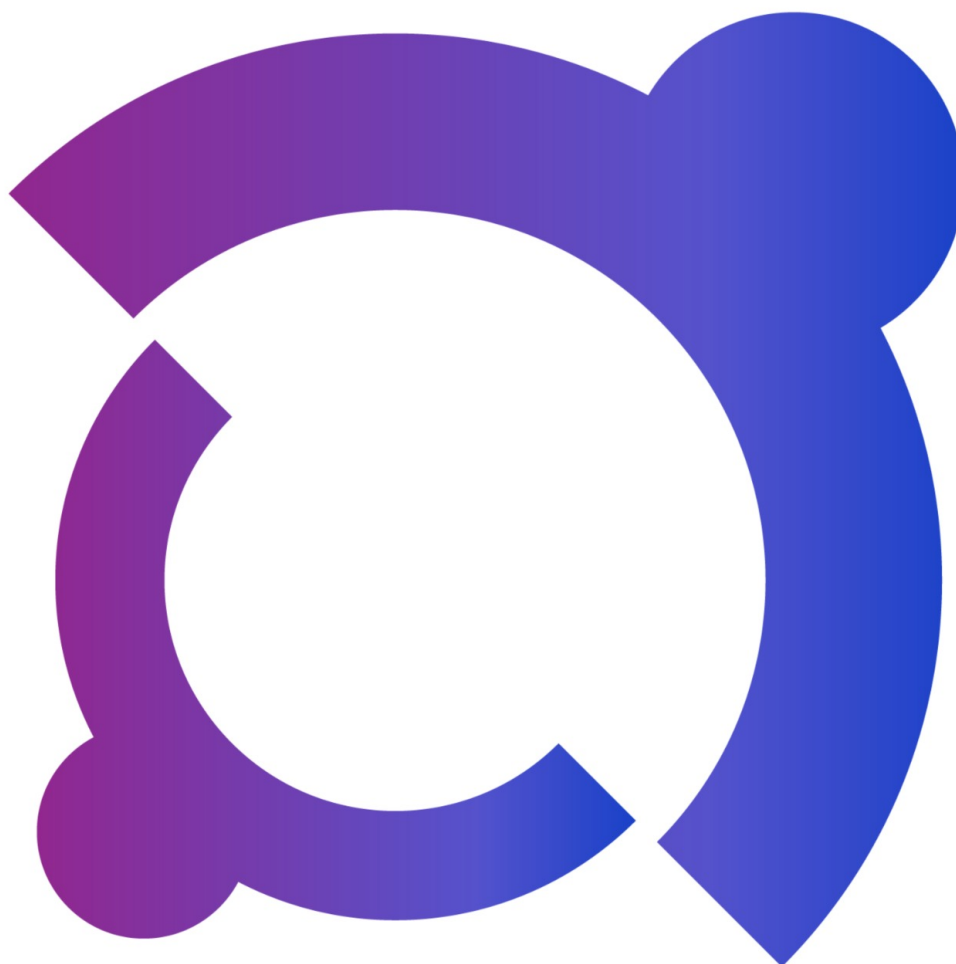




OBJECT DESIGN DOCUMENT - OPENMEET



Object Design Document

All'attenzione di: Prof. Carmine Gravino

Preparato da: Angelo Nazzaro, Yuri Brandi, Roberto Della Rocca, Francesco Granozio

Versione: 0.4



OBJECT DESIGN DOCUMENT - OPENMEET

Sommario

Revision History	3
Team Members	4
1. Introduzione	5
1.1 Object design trade-offs	5
1.2 Linee guida per la documentazione di interfacce	5
1.3 Definizioni, acronimi e abbreviazioni	6
1.4 Riferimenti	6
2. Packages	7
2.1 Shared Package	8
2.2 Webapp Package	17
2.3 Webservice Package	20
2.4 Mobileapp Package	24
3. Class Interfaces	32
3.1 Storage Package	32
3.2 Exceptions Package	41
3.3 Helpers Package	41
3.4 Utils Package	44
4. Design Patterns	51
4.1 Proxy	51
5. Glossario	52



OBJECT DESIGN DOCUMENT - OPENMEET

Revision History

Data	Versione	Descrizione	Autori
22/12/2022	0.1	Prima stesura.	FG, YB, RB
10/02/2023	0.2	Prima stesura delle sezioni: 2. Packages, 4. Design Patterns, 5. Glossario	AN
13/02/2023	0.3	Aggiunta della struttura completa dei packages e del design pattern Proxy.	Membri del Team
14/02/2023	0.4	Stesura della sezione: 3. Class Interfaces	AN, FG
17/02/2023	0.05	Revisione finale.	Membri del Team



OBJECT DESIGN DOCUMENT - OPENMEEET

Team Members

Nome e Cognome	Ruolo	Informazioni di contatto	Matricola	Acronimo
Angelo Nazzaro	Membro del Team	a.nazzaro13@studenti.unisa.it	0512110391	AN
Yuri Brandi	Membro del Team	y.brandi@studenti.unisa.it	0512109740	YB
Roberto Della Rocca	Membro del Team	r.dellarocca5@studenti.unisa.it	0512110802	RD
Francesco Granozio	Membro del Team	f.granozio1@studenti.unisa.it	0512111903	FG

OBJECT DESIGN DOCUMENT - OPENMEEET

1. Introduzione

OpenMeet è un'applicativo mobile per incontri che, rispetto ai concorrenti già presenti sul mercato, offre un'alternativa **gratuita**, **open-source** e **privacy-oriented**.

Nella prima sezione del documento verranno descritti i trade-offs, le linee guida da rispettare nella fase di implementazione riguardanti la terminologia, documentazione e le convenzioni sui vari formati esistenti.

1.1 Object design trade-offs

Trade-Off	Razionale
Riuso vs. Sviluppo	Se esistono componenti del sistema già sviluppate da terzi, è possibile valutarne il riutilizzo attraverso appositi design pattern solo nel caso in cui siano open-source e ben documentati al fine di mantenere l'integrità open-source del sistema.
Spazio di memoria vs. Tempo di risposta	Se dei dati del sistema vengono prelevati e/o calcolati con frequenze elevate, è preferibile conservarli nei client con meccanismi di caching e ricorrere a ridondanze nella base di dati.
Tempo di consegna vs. Sicurezza	Se lo sviluppo è in ritardo, il PM può decidere di sacrificare momentaneamente aspetti minori legati alla sicurezza degli utenti ed integrarli successivamente.

1.2 Linee guida per la documentazione di interfacce

In queste linee guida vengono descritte le regole per la progettazione e l'assegnazione dei nomi delle interfacce. Sono convenzioni che gli sviluppatori del sistema dovrebbero seguire per aumentare la coerenza e la comunicazione.

A seguire, una lista delle linee guida dei linguaggi che verranno usati per lo sviluppo di OpenMeet:

- [Kotlin Coding Conventions](#);
- [Secure Coding Guidelines for Java SE](#);
- [Guidelines for styling HTML code](#);
- [Guidelines for styling CSS code](#);
- [Guidelines for styling JavaScript code](#);
- [XML Syntax Rules](#);

OBJECT DESIGN DOCUMENT - OPENMEEET

1.3 Definizioni, acronimi e abbreviazioni

Di seguito, una lista di definizioni, acronimi e abbreviazioni:

- **Package:** raggruppamento di classi, interfacce e/o file correlati;
- **Design pattern:** template di soluzioni a problemi ricorrenti impiegati per favorire il riuso e la flessibilità;
- **Interfaccia:** insieme di definizioni delle operazioni offerte da una classe;
- **Javadoc:** sistema di documentazione offerto da Java che viene generato sotto-forma di interfaccia in modo da rendere la documentazione accessibile e facilmente leggibile;
- **KDoc:** sistema di documentazione per Kotlin. È equivalente a Javadoc per Java;
- **PM:** Project Manager.;

1.4 Riferimenti

Di seguito sono riportati gli altri documenti relativi al progetto utili per una maggiore comprensione e contestualizzazione:

- [Requirement Analysis Document - RAD](#)
- [System Document Design - SDD](#)
- [Test Case Plan - TCP](#)
- [Test Case Specification - TCS](#)

OBJECT DESIGN DOCUMENT - OPENMEEET

2. Packages

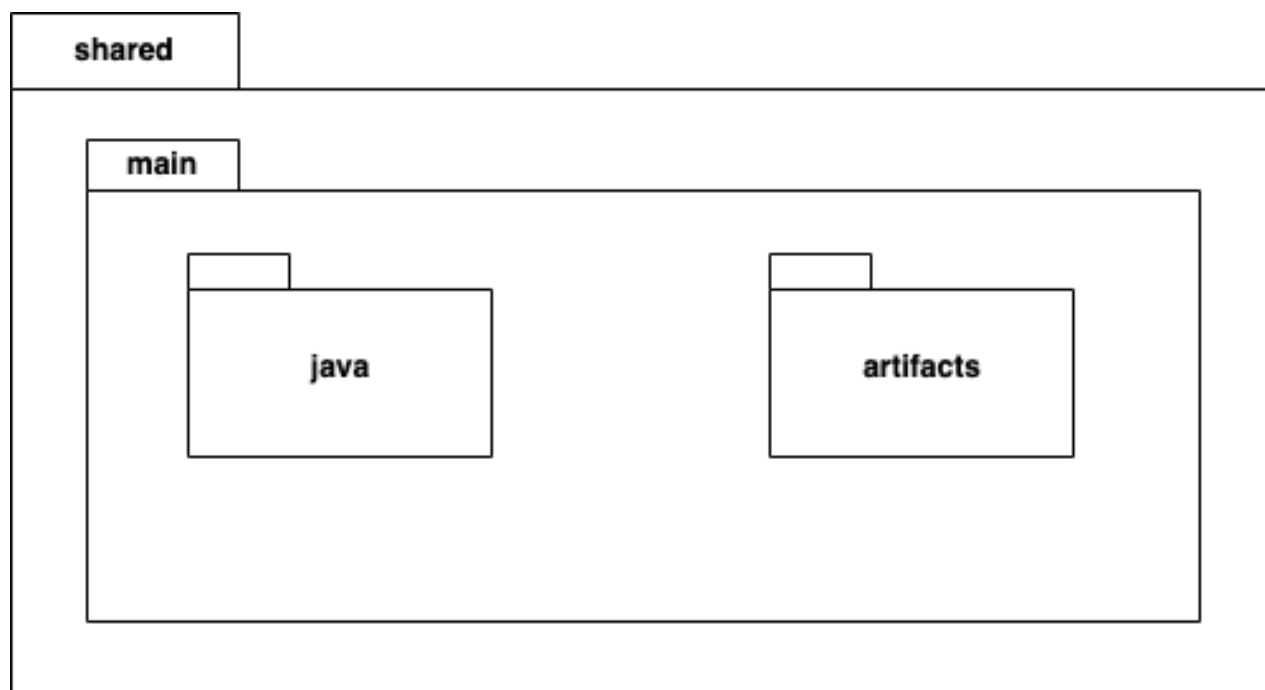
In questa sezione viene mostrata la suddivisione del sistema in packages, in base a quanto definito nel documento di System Design.

- **database:** contiene gli scripts necessari alla creazione e alla popolazione del database.
- **shared**, contiene classi ed interfacce condivise ed usate dalla **webapp**, **webservice** e dalla **mobileapp**.
 - **src**, contiene tutti i files sorgente
 - **main**
 - **java**, contiene le classi Java relative ad entità, Dao ed altre classi di utilità condivise dalla webapp e dai webservice.
 - **artifacts**, contiene il file jar esportato
- **webapp**, contiene i files relativi all'applicativo web usato dai Moderatori.
 - **src**, contiene tutti i file sorgente
 - **main**
 - **java**, contiene le classi al logic e al data tier come Servlets, DAOs e filtri di autenticazione.
 - **webapp**, contiene i file relativi al presentation tier, come componenti views, template e varie risorse.
 - **test**, contiene tutto il necessario per il testing.
 - **java**, contiene le classi java per l'implementazione del testing.
 - **target**, contiene tutti i file prodotti dal sistema di build di Gradle.
- **webservice**, contiene i files riguardanti i web services che vengono usati dall'applicativo mobile per interfacciarsi col resto del livello logico.
 - **src**, contiene tutti i file sorgente
 - **main**
 - **java**, contiene le classi al logic e al data tier come le classi per l'erogazione dei servizi ed implementazioni dei proxies.
 - **test**, contiene tutto il necessario per il testing.
 - **java**, contiene le classi java per l'implementazione del testing.

OBJECT DESIGN DOCUMENT - OPENMEET

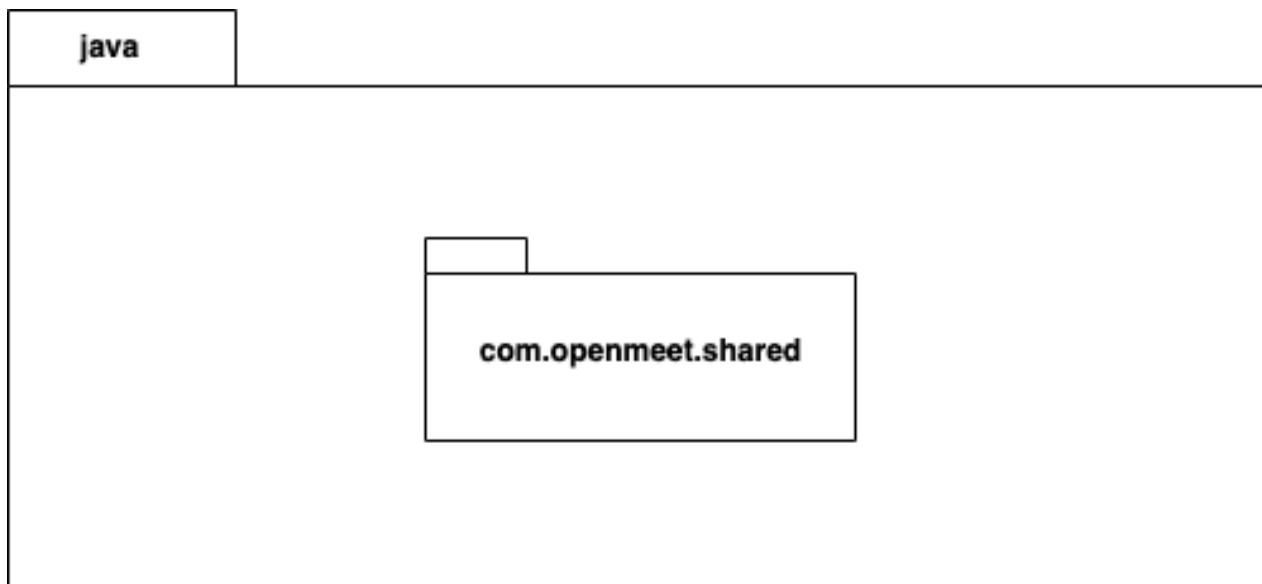
- **target**, contiene tutti i file prodotti dal sistema di build di Gradle.
- **mobileapp**, contiene i files relativi all'applicazione mobile.
 - **app**
 - **src**, contiene tutti i files sorgente.
 - **Build**, contiene tutti i file prodotti dal sistema di build di Gradle.
 - **main**
 - **java**, contiene i files relative al Logic e al Data Layer dell'applicativo mobile.
 - **res**, contiene i files e le risorse relative al Presentation Layer dell'applicativo mobile.

2.1 Shared Package

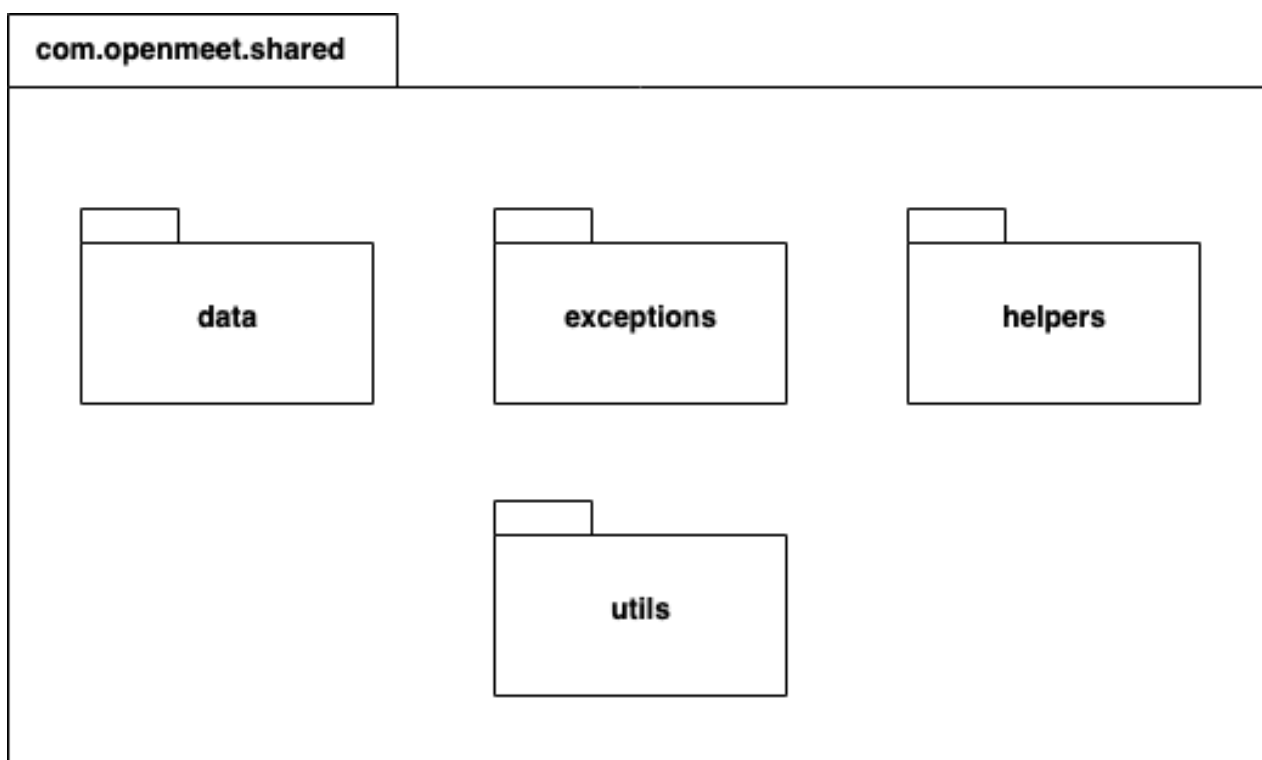


OBJECT DESIGN DOCUMENT - OPENMEEET

2.1.1 Java Package

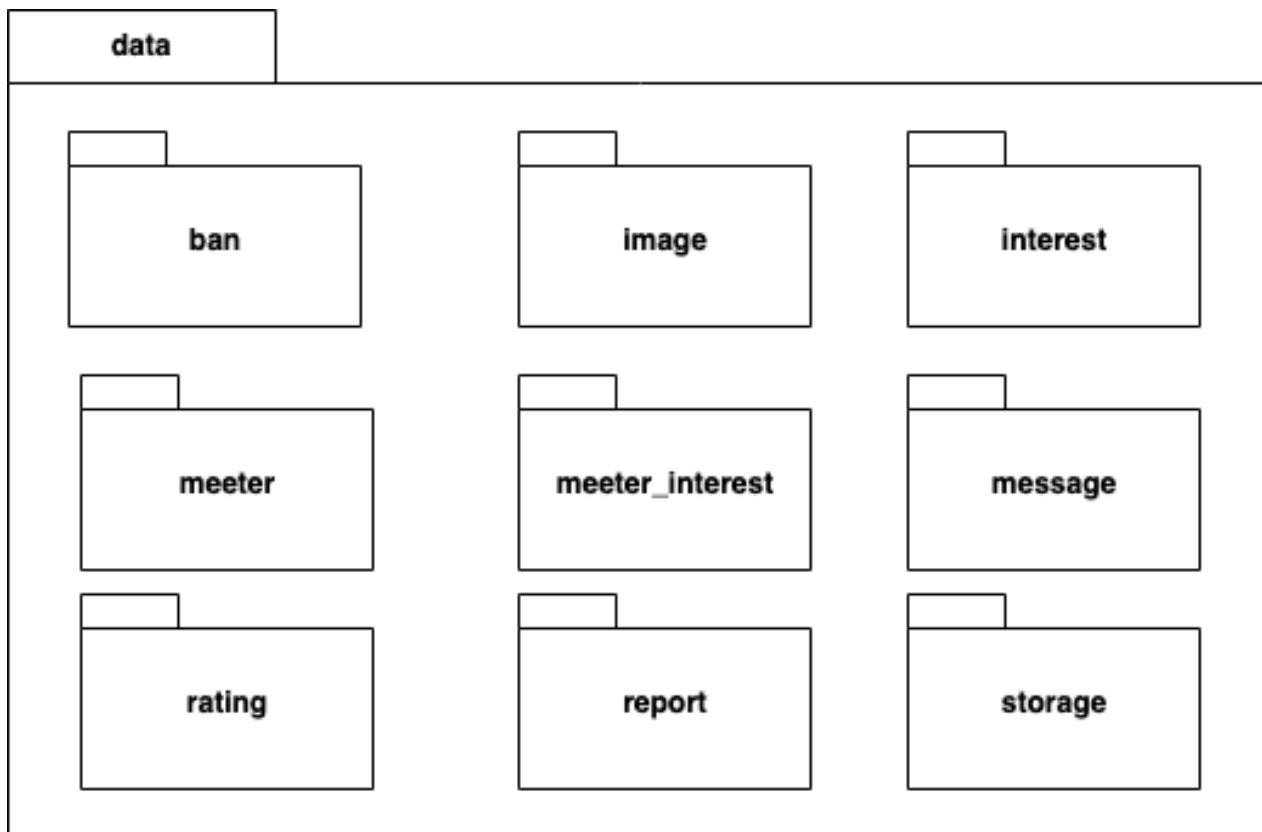


2.1.2 com.openmeet.shared Package

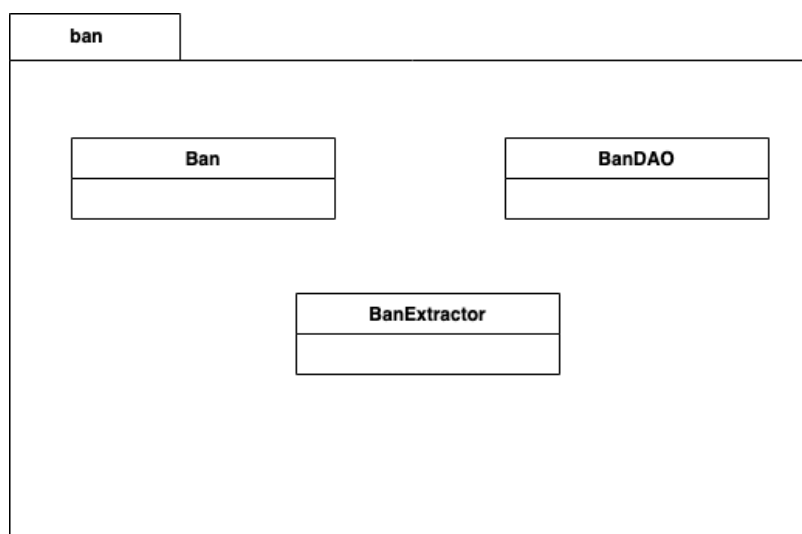


OBJECT DESIGN DOCUMENT - OPENMEEET

2.1.3 Data Package

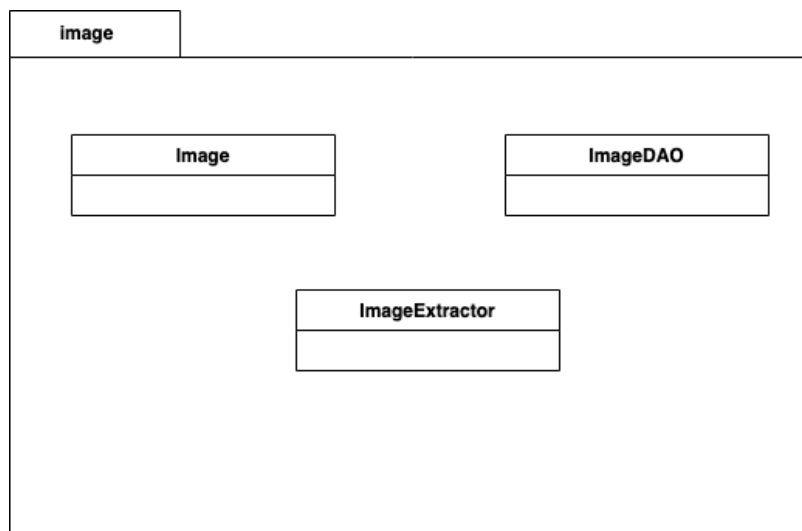


2.1.4 Ban Package

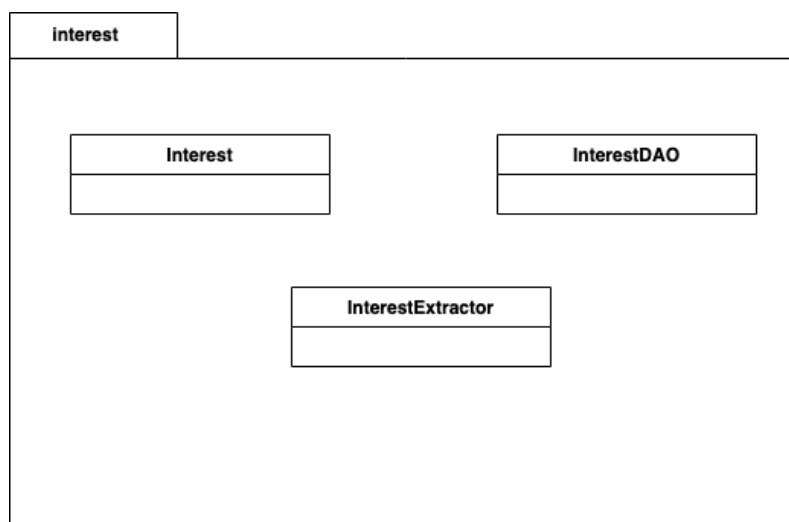


OBJECT DESIGN DOCUMENT - OPENMEEET

2.1.5 Image Package

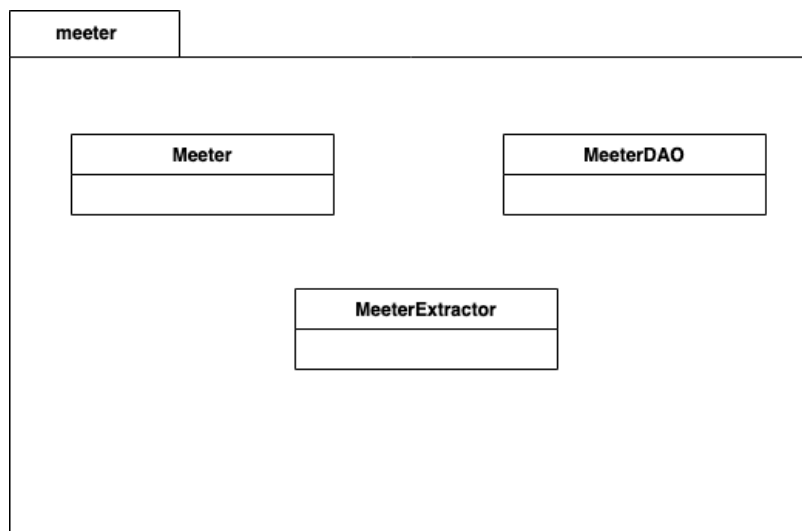


2.1.6 Interest Package

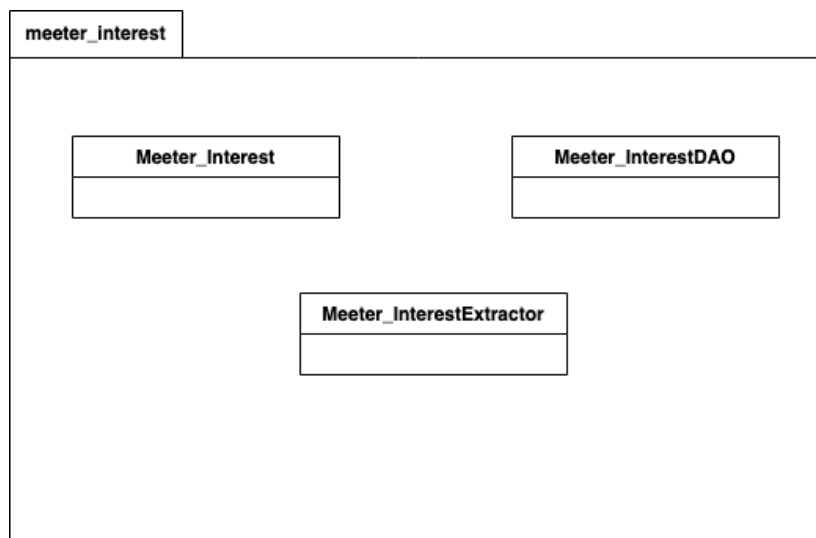


OBJECT DESIGN DOCUMENT - OPENMEEET

2.1.7 Meeter Package

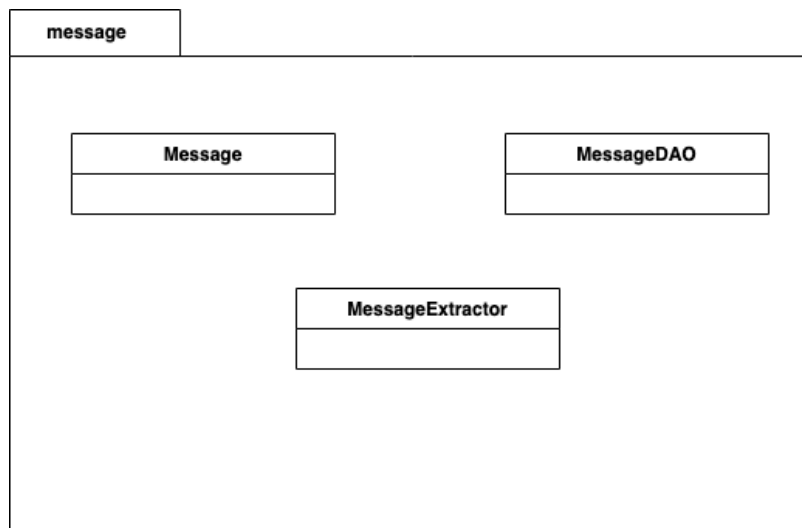


2.1.8 Meeter_Interest Package

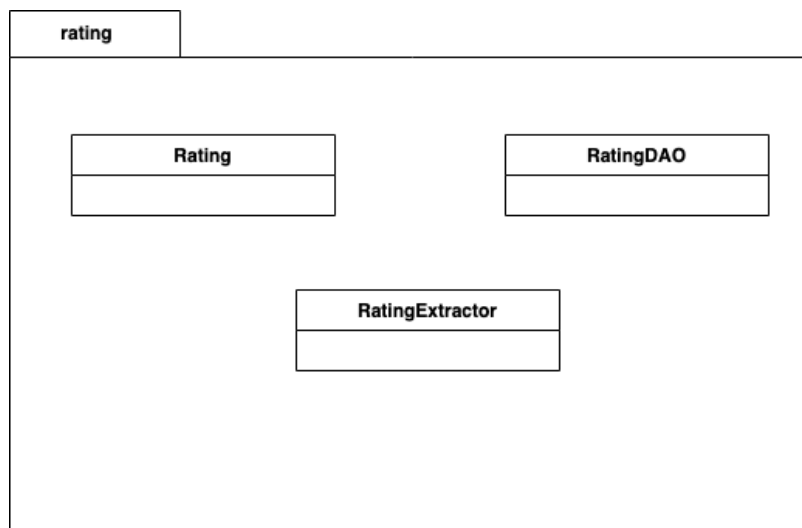


OBJECT DESIGN DOCUMENT - OPENMEEET

2.1.9 Message Package

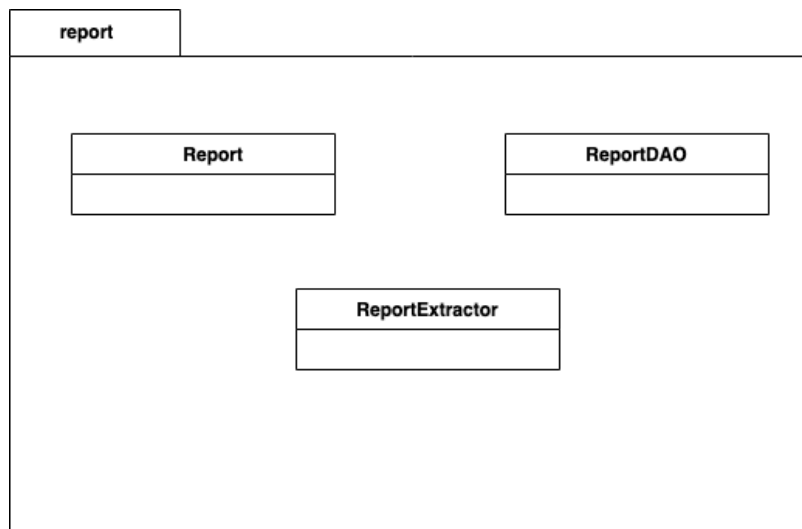


2.1.10 Rating Package

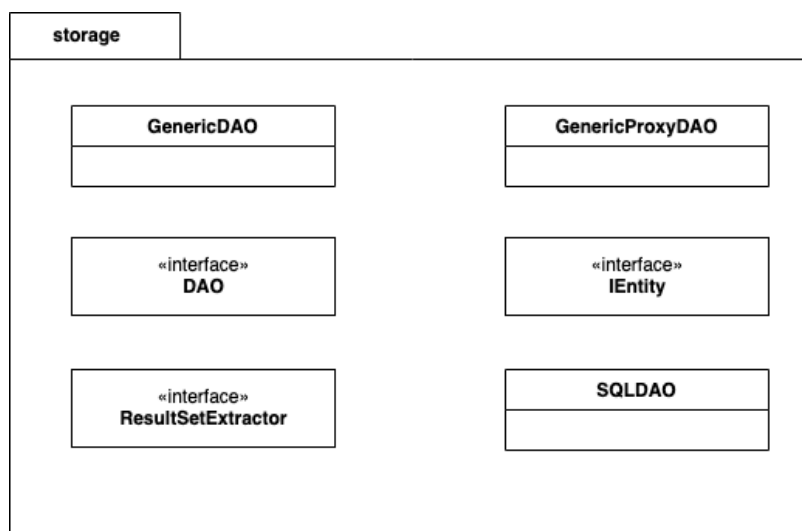


OBJECT DESIGN DOCUMENT - OPENMEEET

2.1.11 Report Package

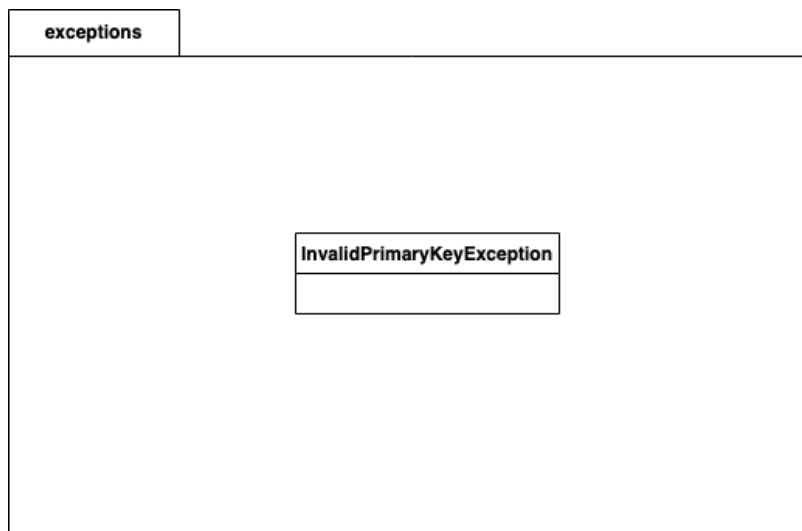


2.1.12 Storage Package

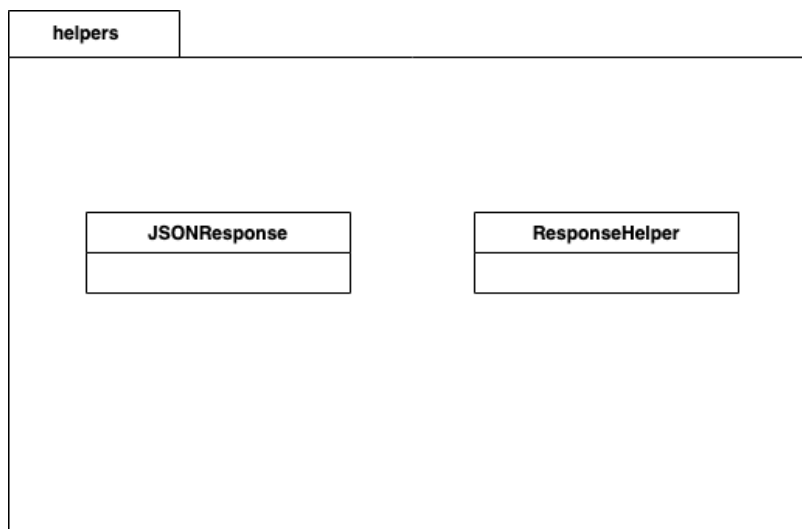


OBJECT DESIGN DOCUMENT - OPENMEEET

2.1.13 Exceptions Package



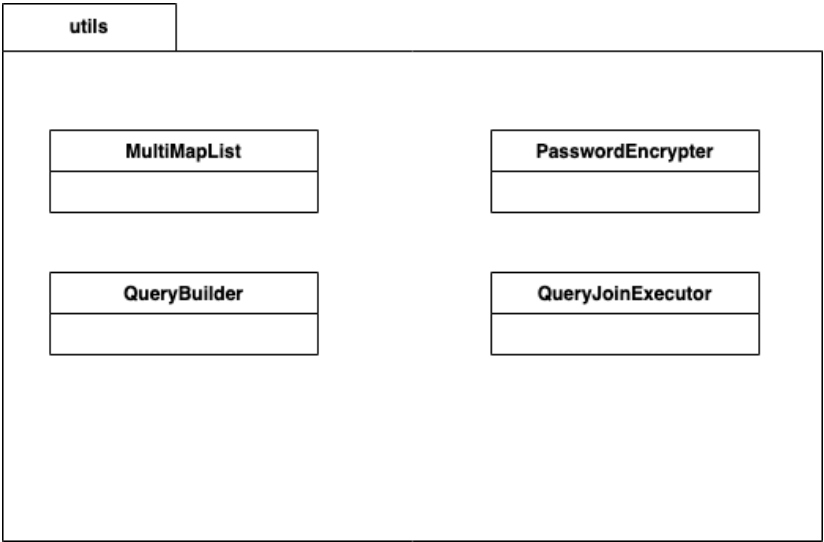
2.1.14 Helpers Package





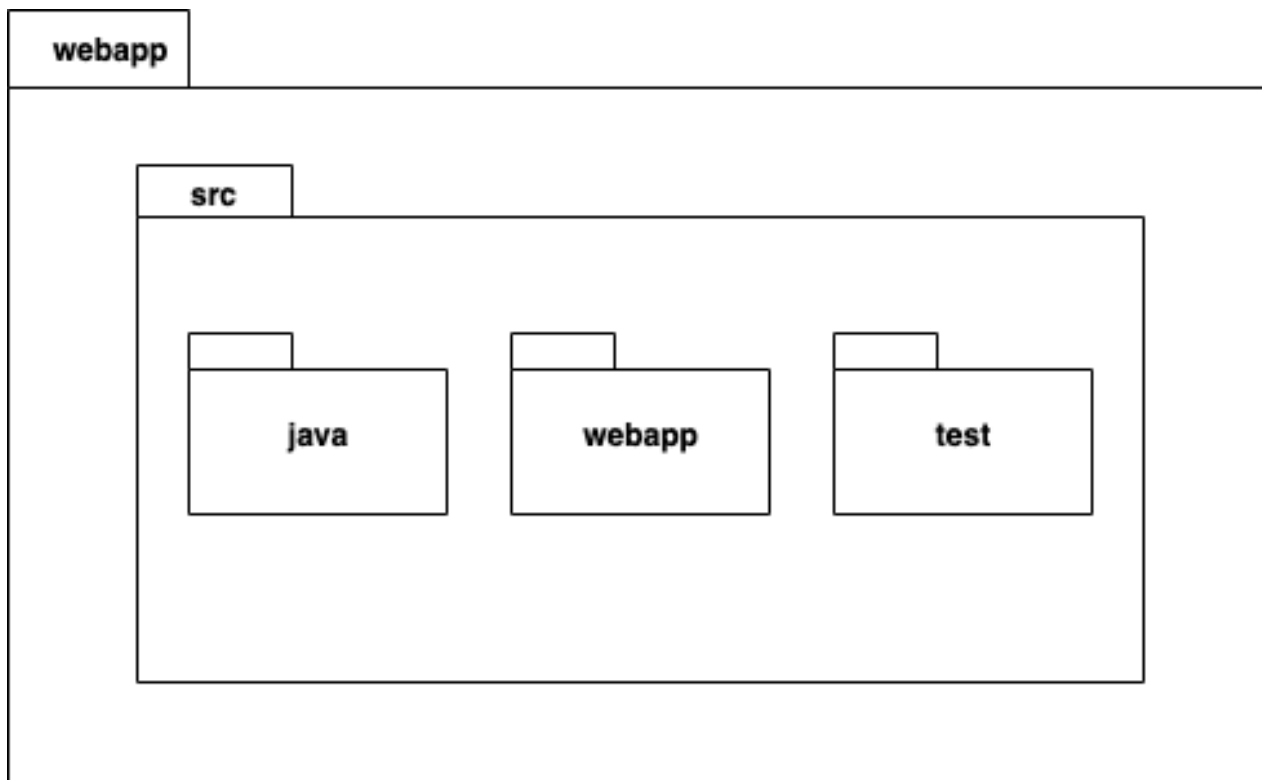
OBJECT DESIGN DOCUMENT - OPENMEEET

2.1.15 Utils Package

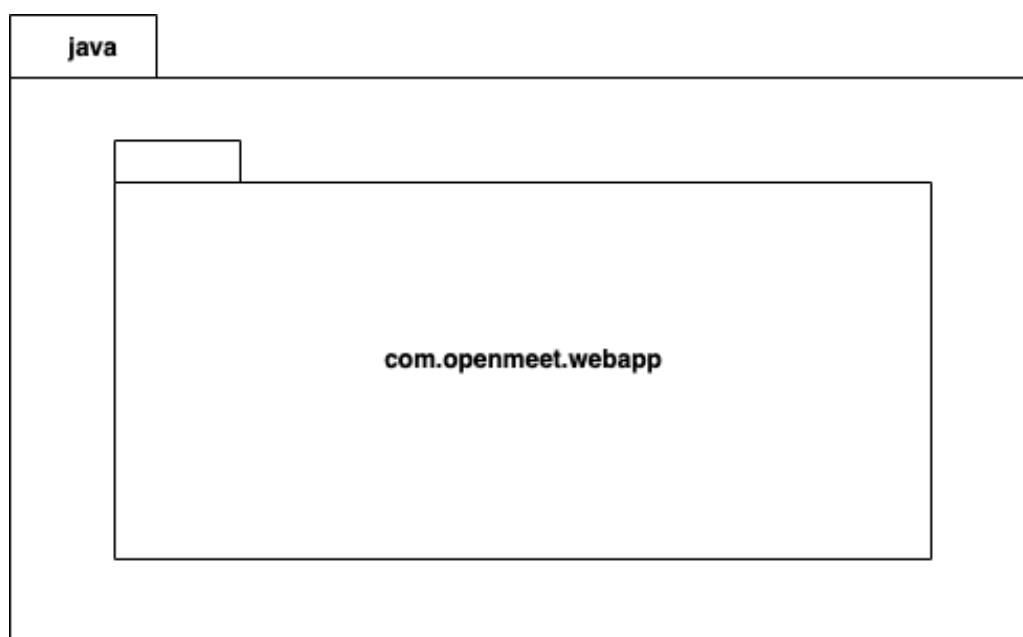


OBJECT DESIGN DOCUMENT - OPENMEEET

2.2 Webapp Package

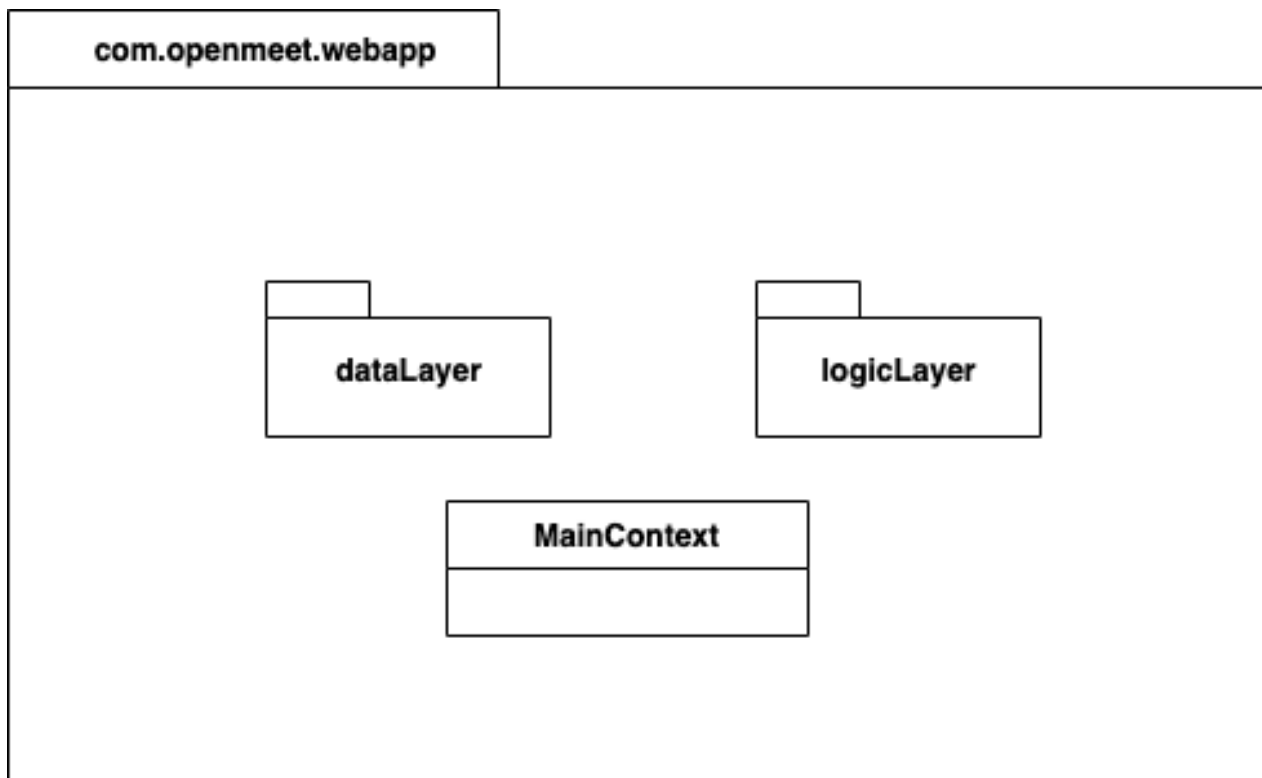


2.2.1 Java Package

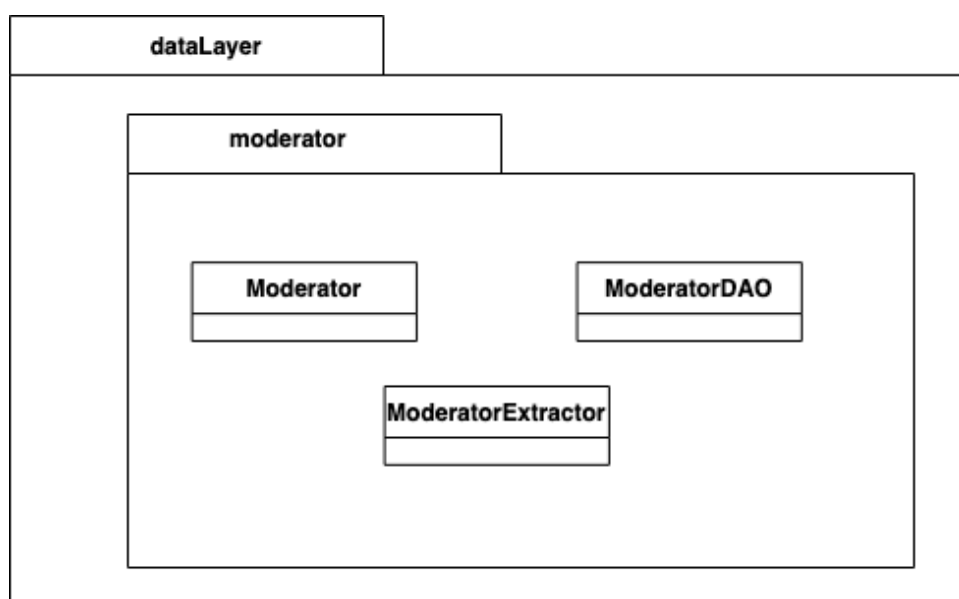


OBJECT DESIGN DOCUMENT - OPENMEEET

2.2.3 com.openmeet.webapp Package

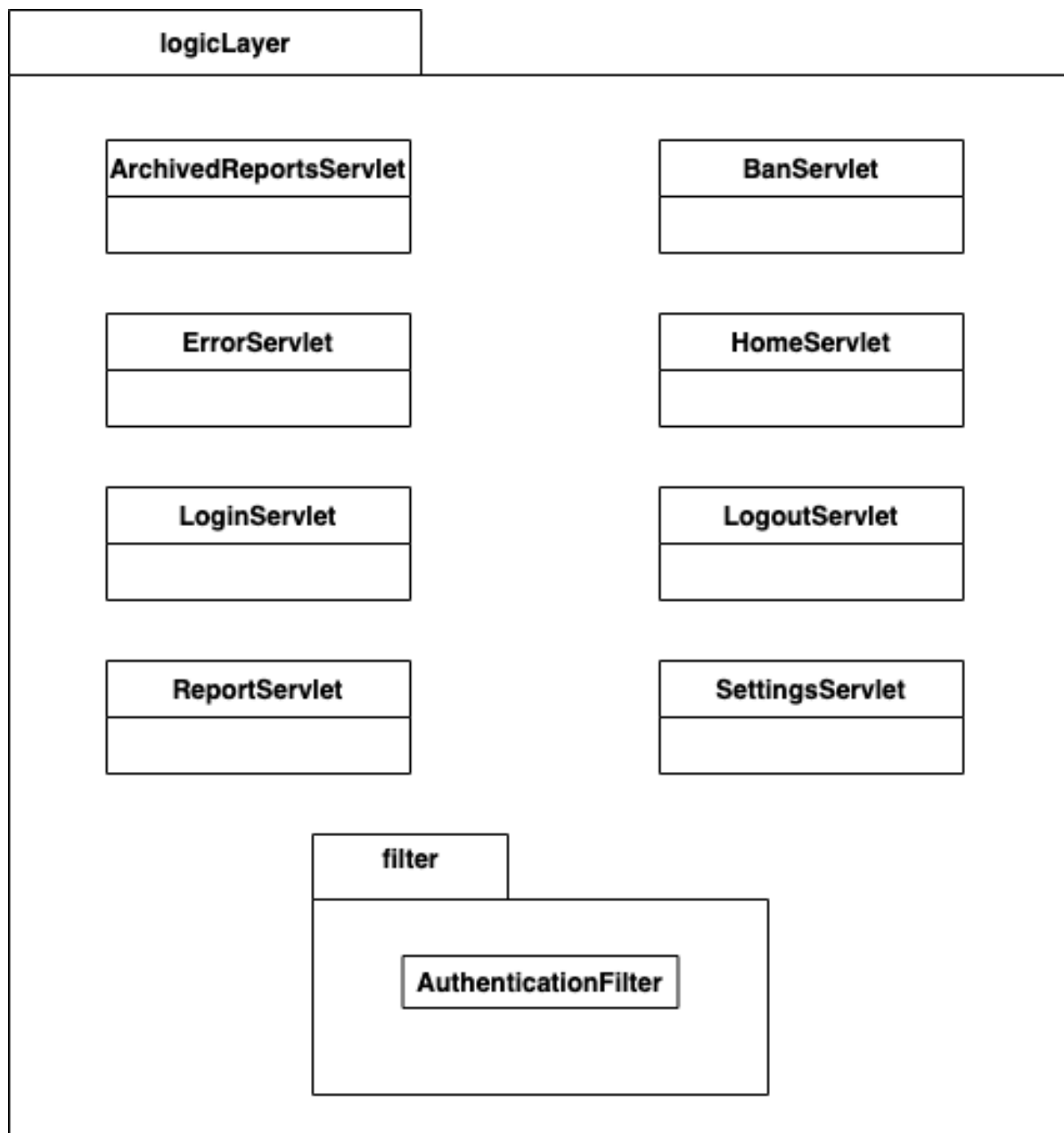


2.2.4 DataLayer Package



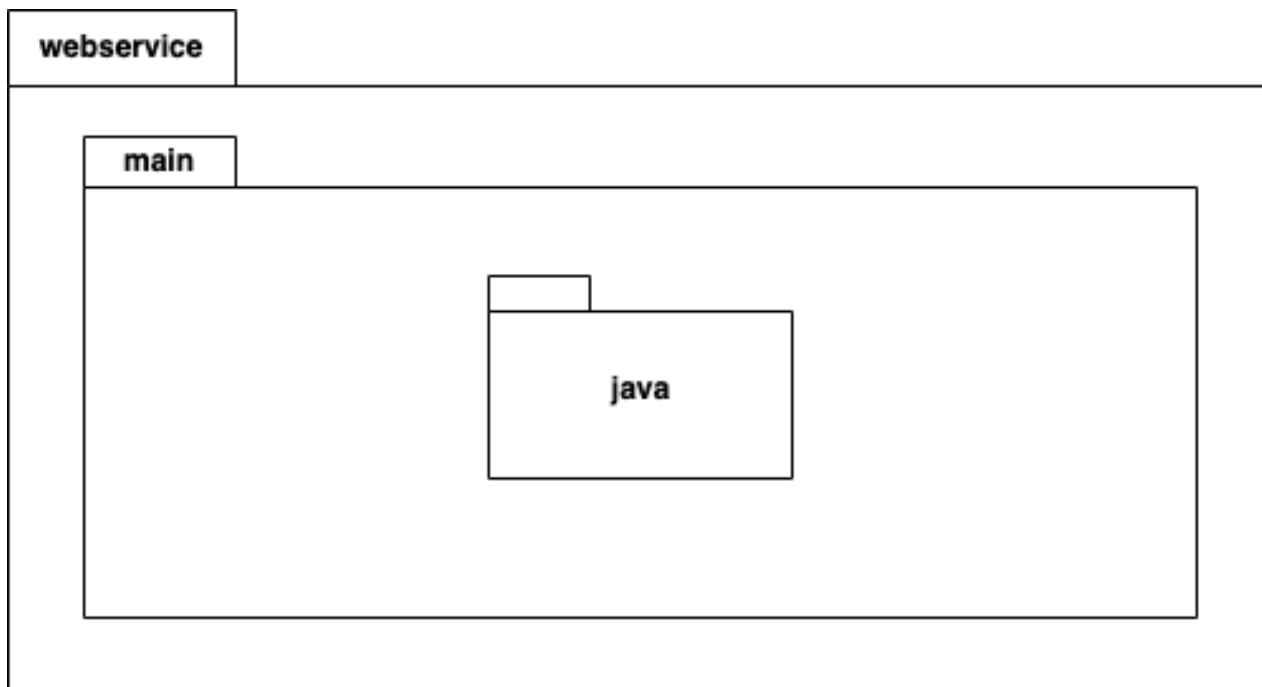
OBJECT DESIGN DOCUMENT - OPENMEEET

2.2.5 LogicLayer Package

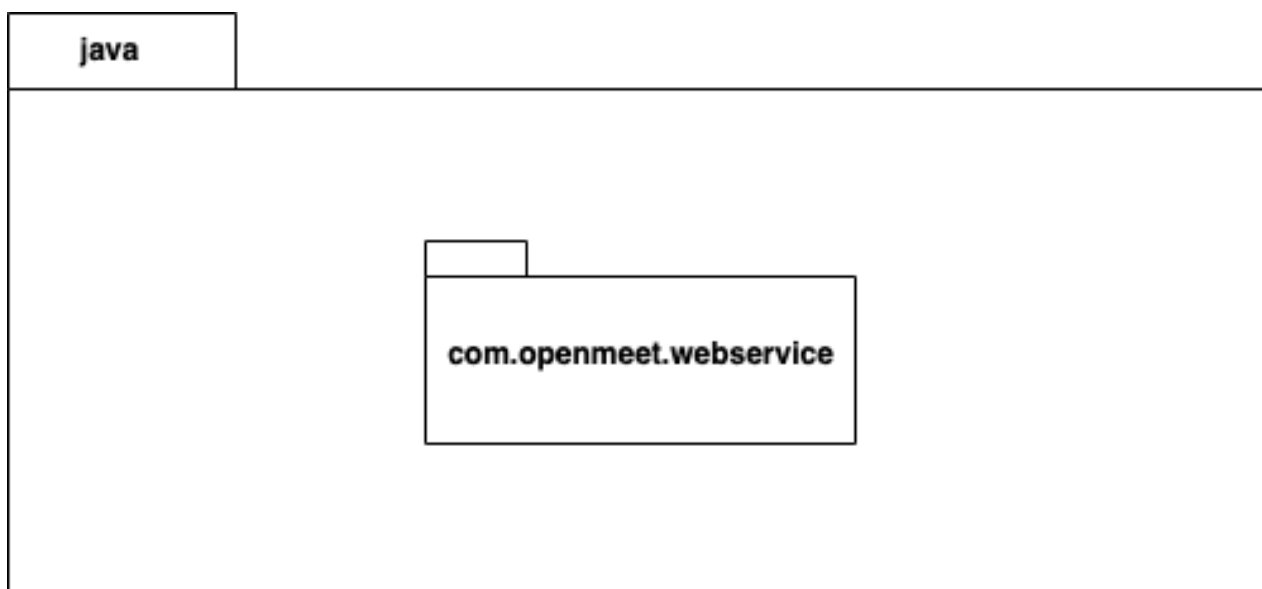


OBJECT DESIGN DOCUMENT - OPENMEEET

2.3 Webservice Package



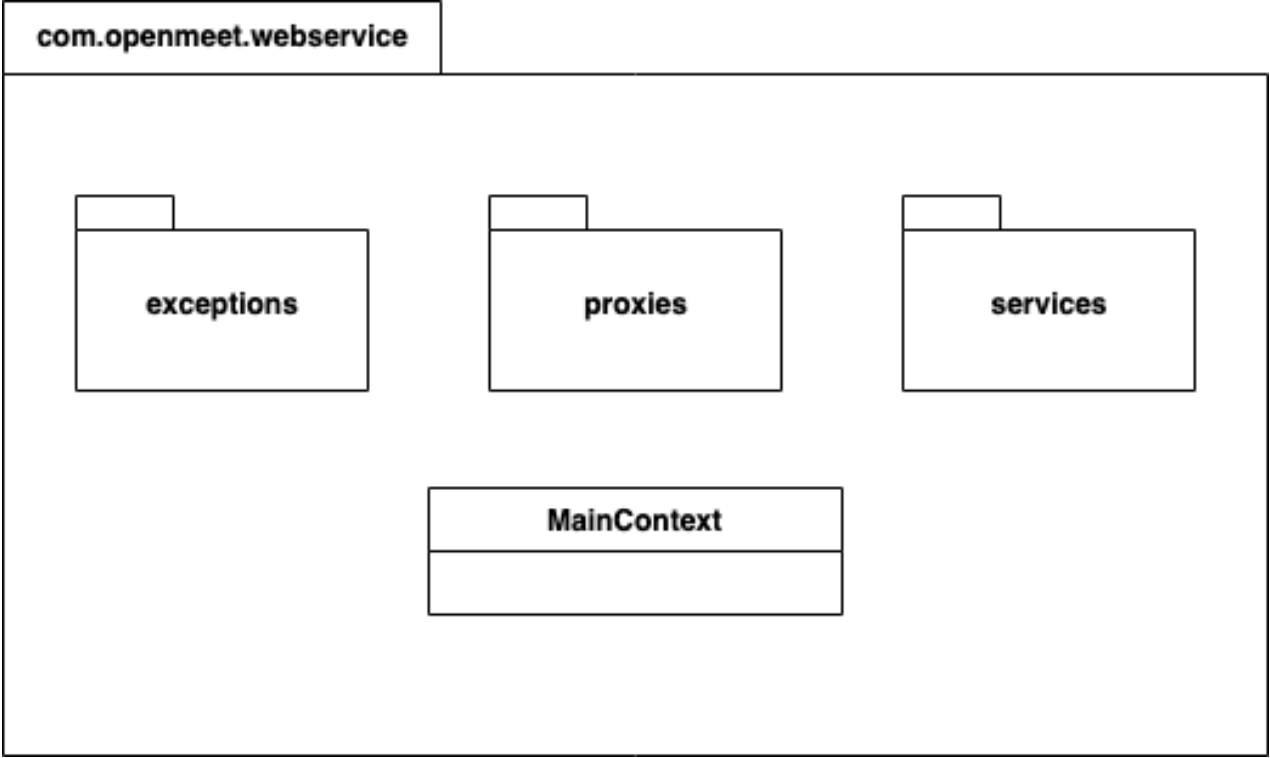
2.3.1 Java Package



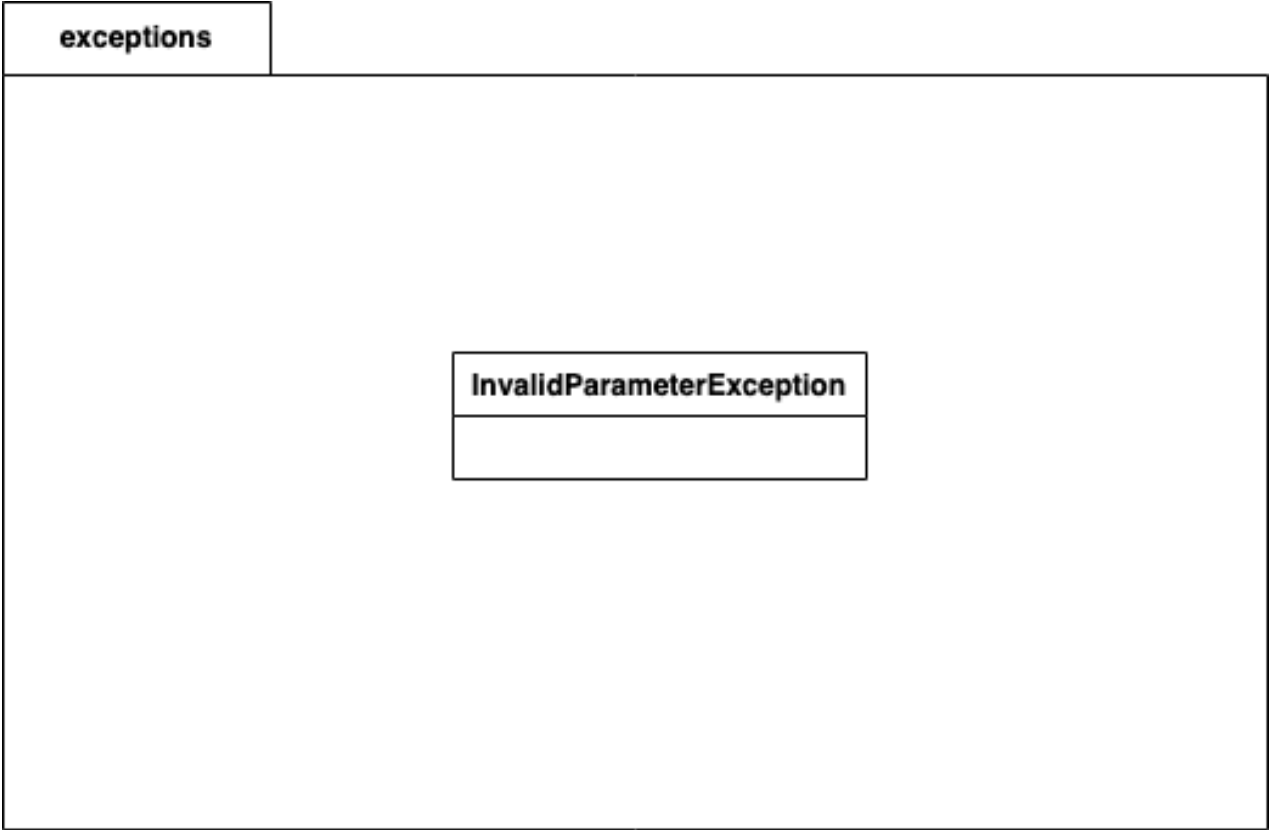


OBJECT DESIGN DOCUMENT - OPENMEEET

2.3.2 com.openmeet.webservice Package

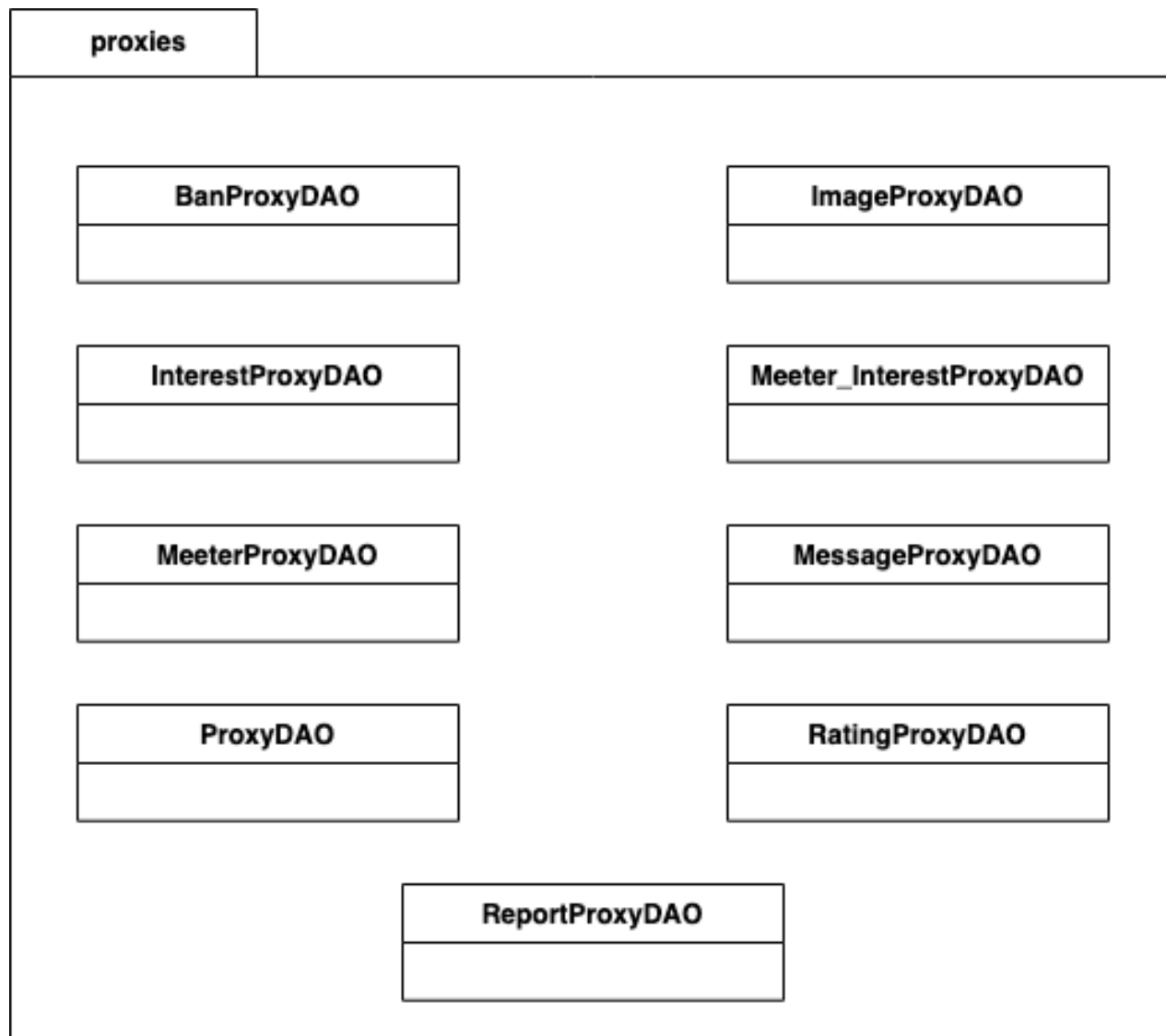


2.3.3 Exceptions Package



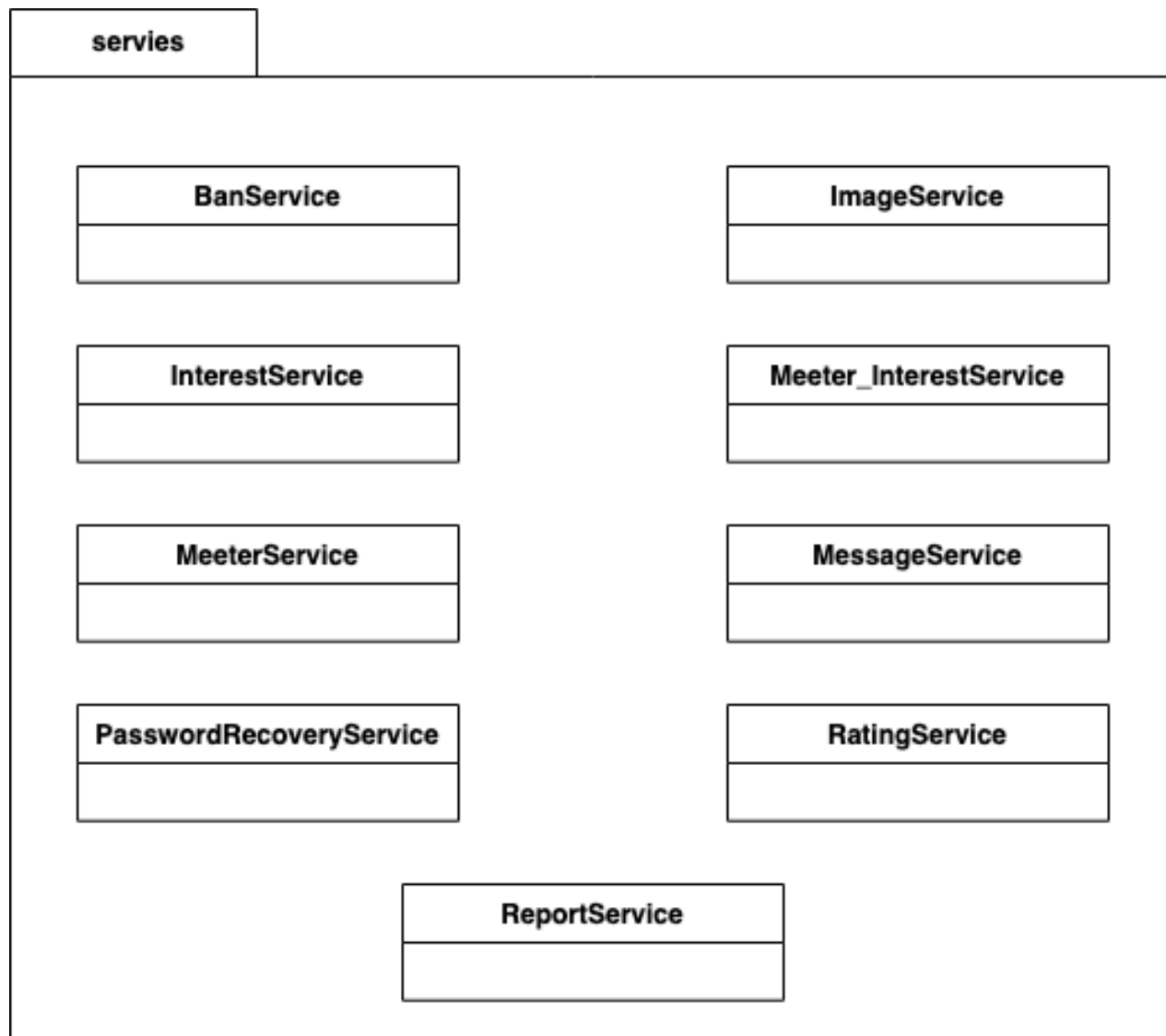
OBJECT DESIGN DOCUMENT - OPENMEEET

2.3.4 Proxies Package



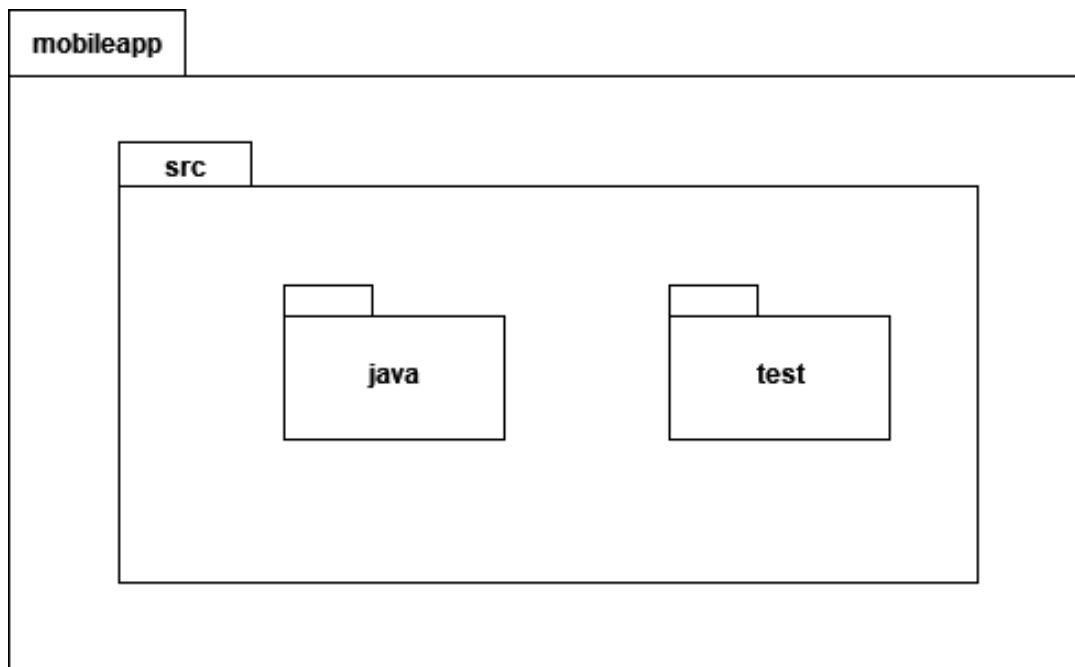
OBJECT DESIGN DOCUMENT - OPENMEEET

2.3.4 Services Package

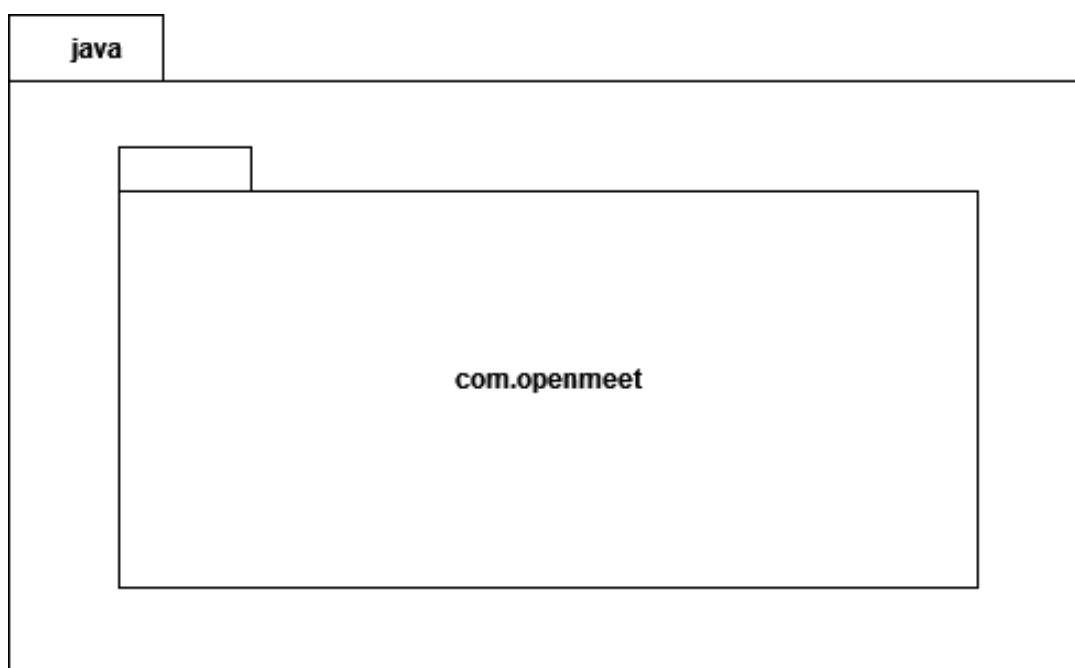


OBJECT DESIGN DOCUMENT - OPENMEEET

2.4 Mobileapp Package

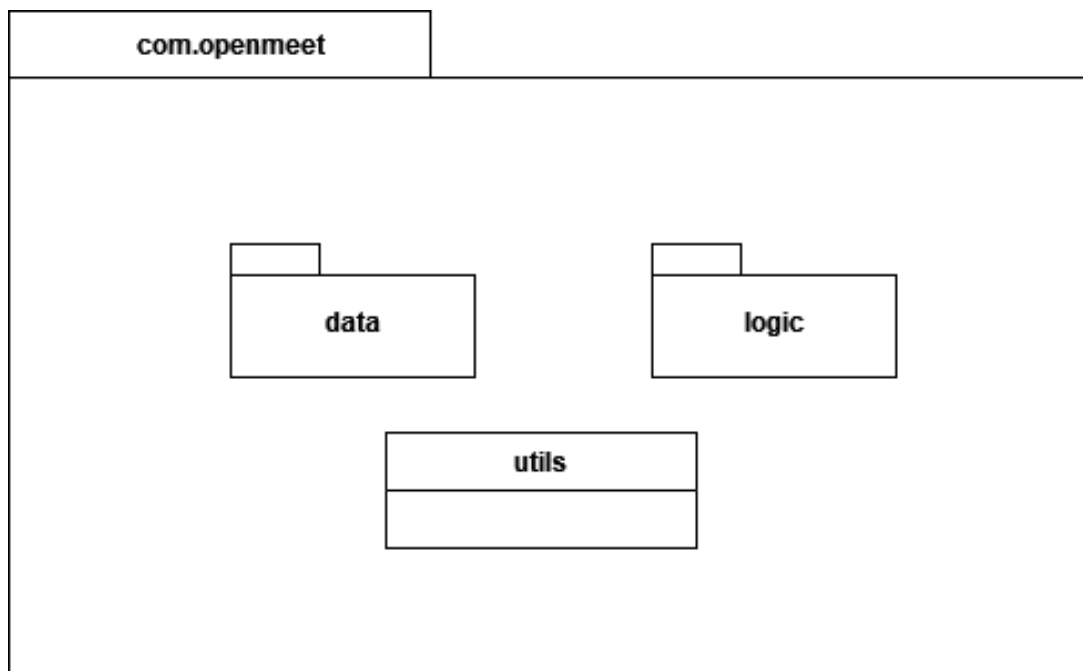


2.4.1 Java Package

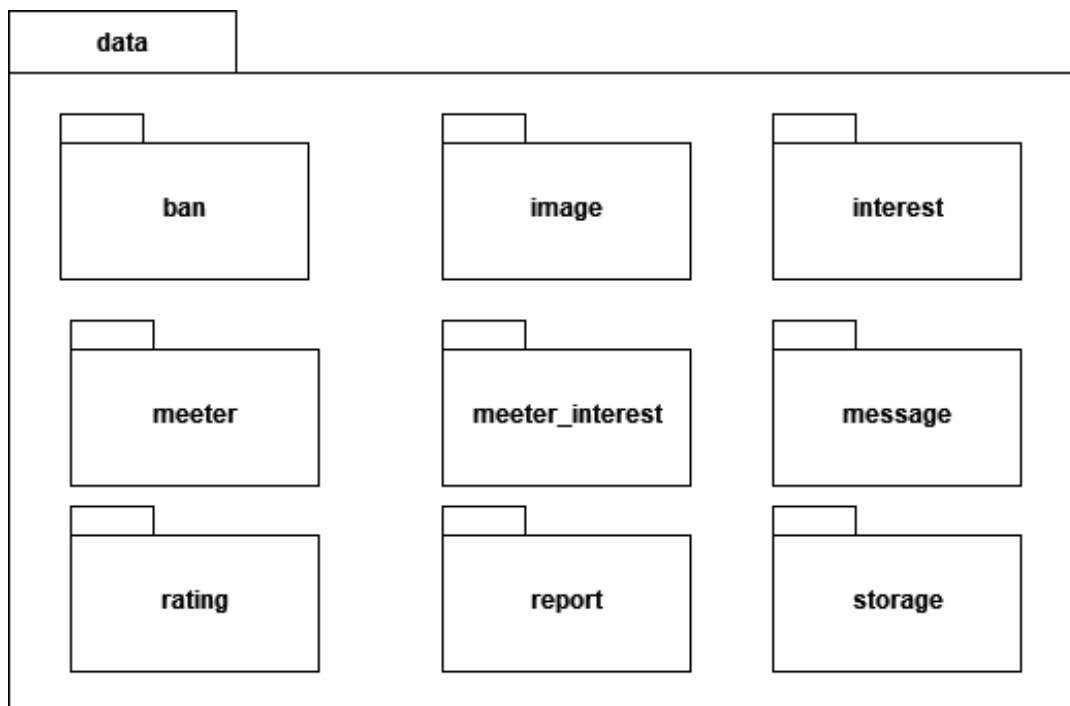


OBJECT DESIGN DOCUMENT - OPENMEEET

2.4.2 com.openmeet Package

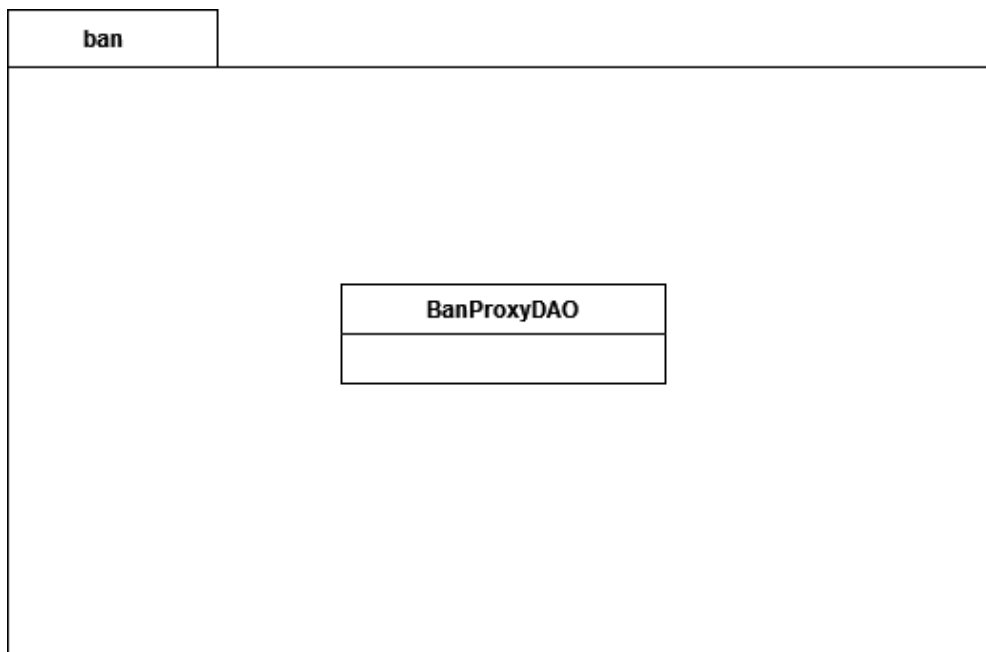


2.4.3 Data Package

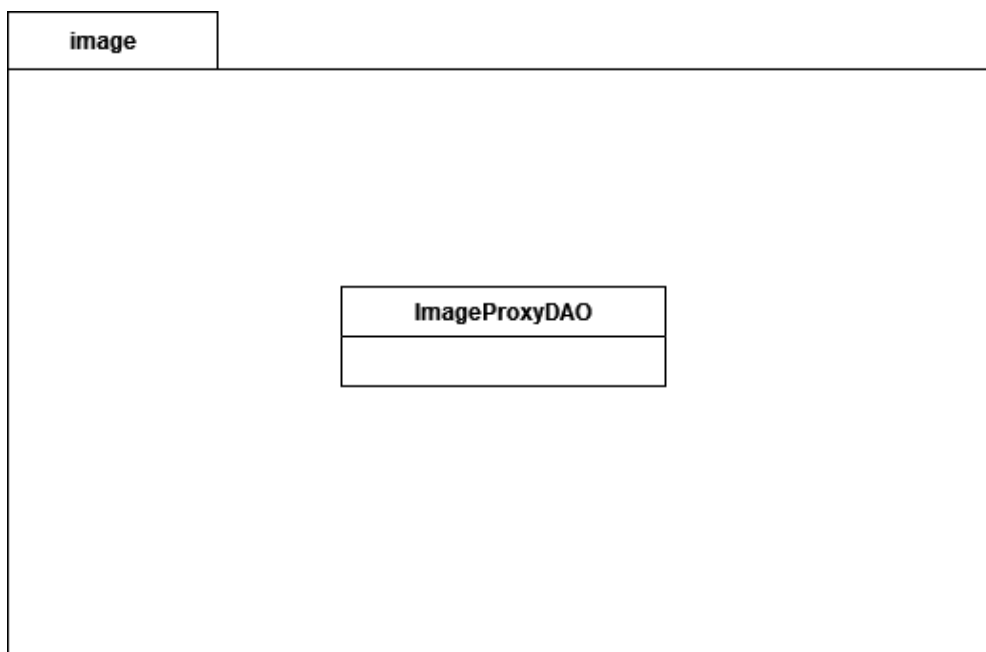


OBJECT DESIGN DOCUMENT - OPENMEEET

2.4.4 Ban Package

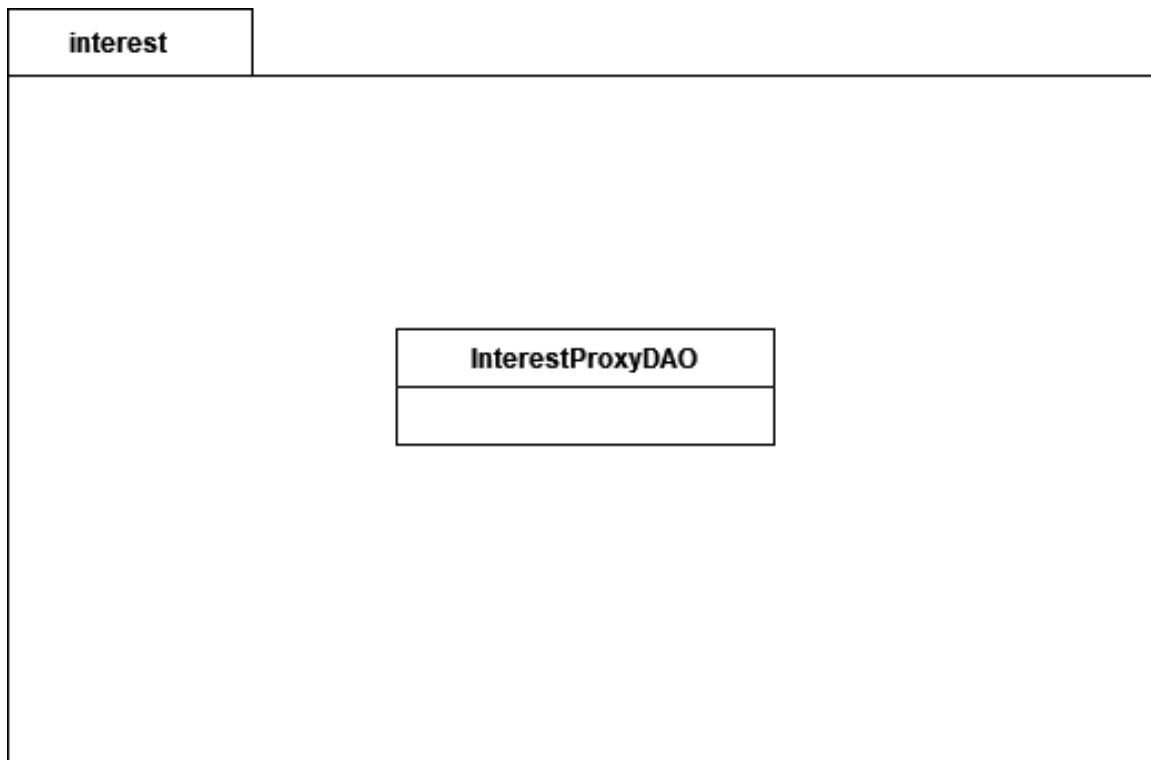


2.4.5 Image Package

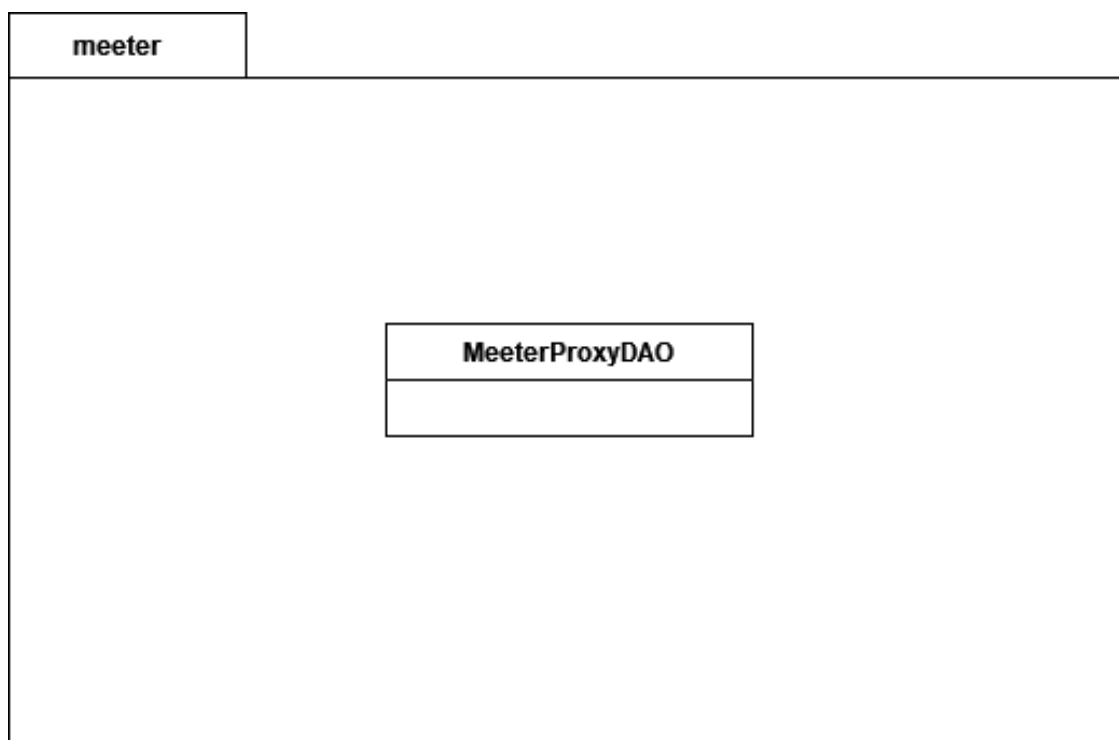


OBJECT DESIGN DOCUMENT - OPENMEEET

2.4.5 Interest Package

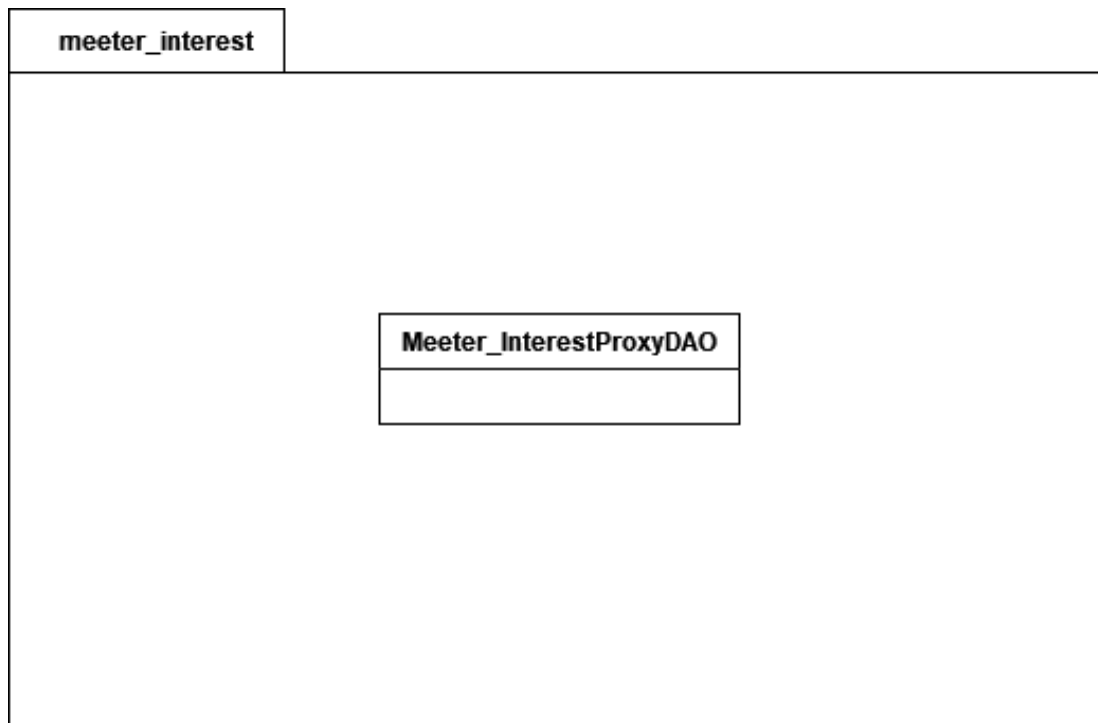


2.4.6 Meeter Package

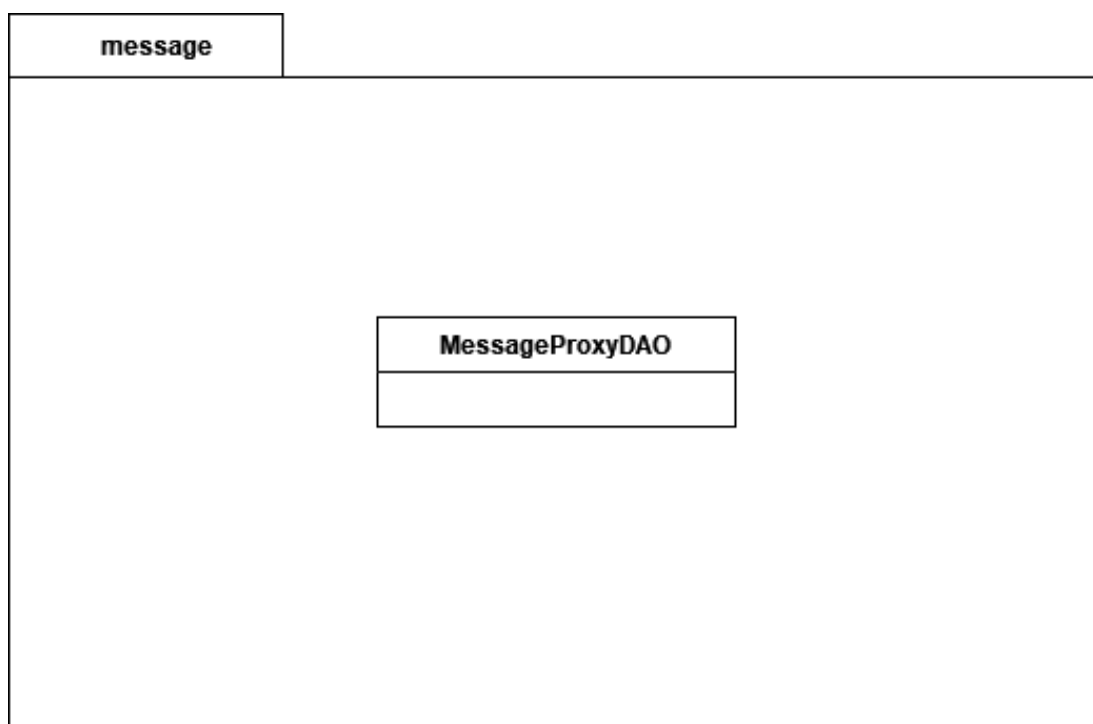


OBJECT DESIGN DOCUMENT - OPENMEEET

2.4.7 Meeter_Interest Package

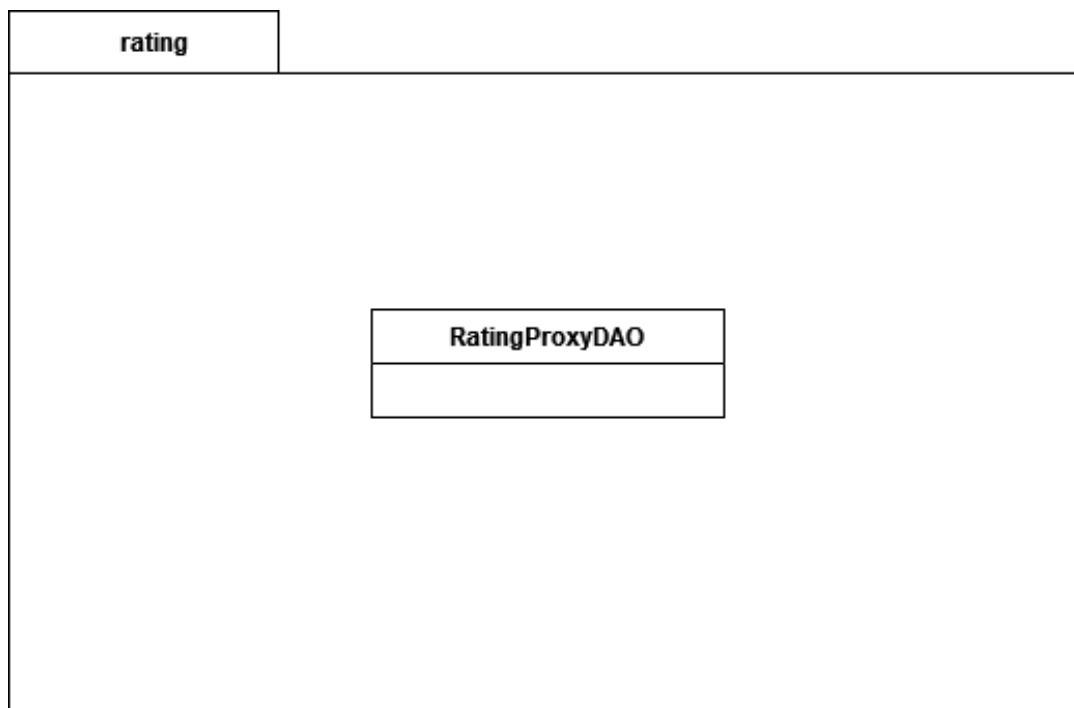


2.4.8 Message Package

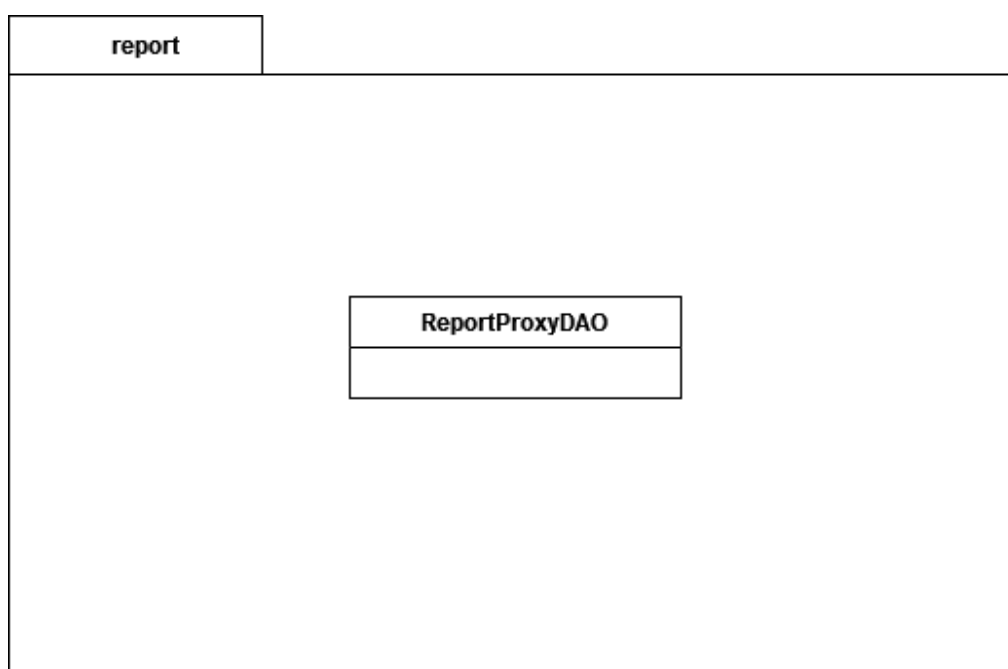


OBJECT DESIGN DOCUMENT - OPENMEEET

2.4.9 Rating Package

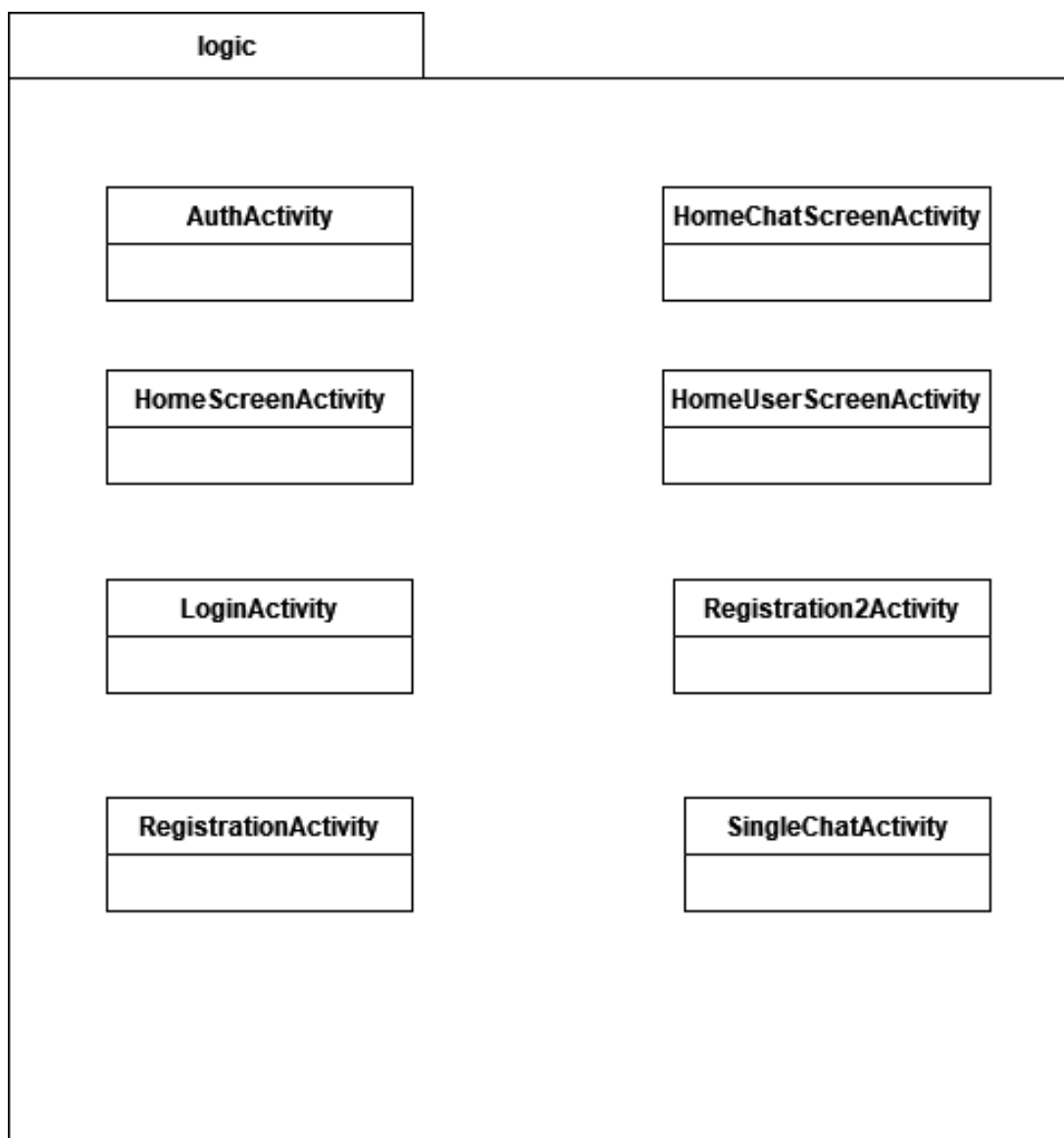


2.4.10 Report Package



OBJECT DESIGN DOCUMENT - OPENMEEET

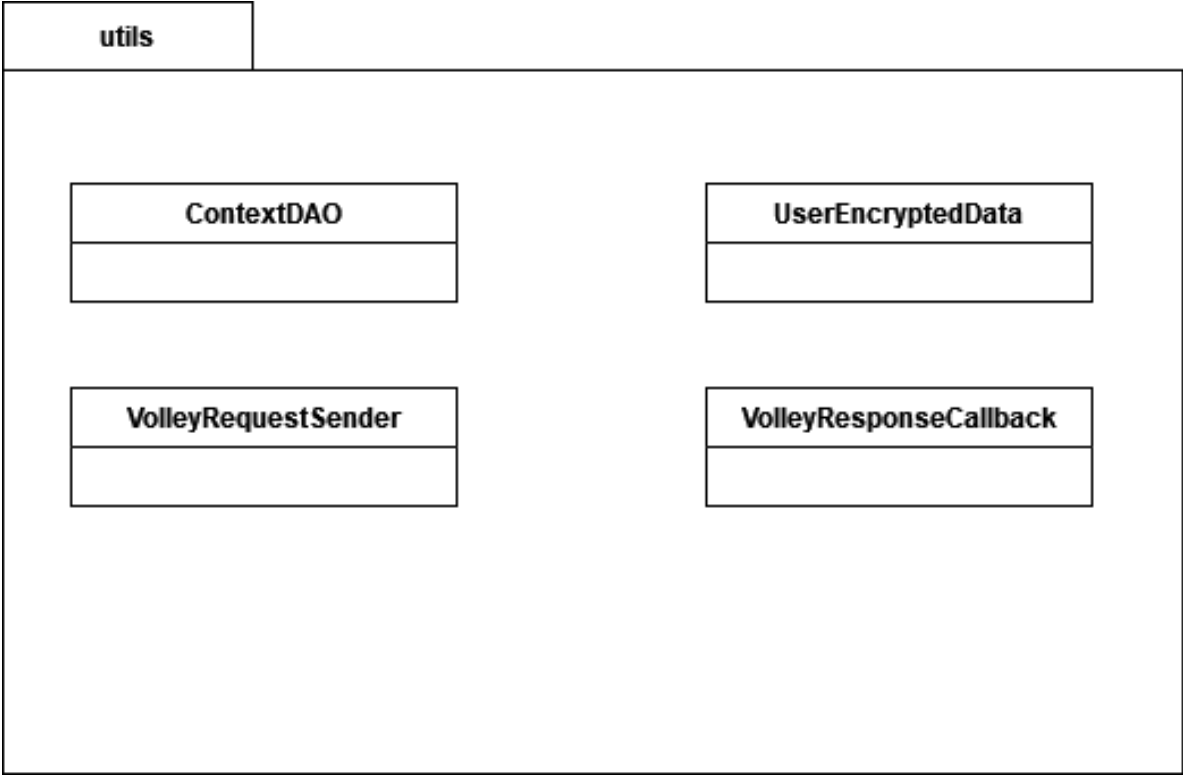
2.4.11 Logic Package





OBJECT DESIGN DOCUMENT - OPENMEEET

2.4.12 Utils Package



OBJECT DESIGN DOCUMENT - OPENMEEET

3. Class Interfaces

Di seguito sono riportate le classi ed interfacce contenute nei packages: **storage**, **exceptions** e **utils** in quanto costituiscono il *core* del sistema. Per quanto riguarda i contenuti dei rimanenti packages si fa riferimento alla **javadoc** e **kdoc** contenute nella [repository del progetto](#).

3.1 Storage Package

Nome classe	DAO
Descrizione	Questa classe permette di gestire le operazioni relative all'inserimento, modifica e cancellazione dei dati nel Database.
Metodi	+ doRetrieveByCondition(String condition): List<T> + doRetrieveByCondition(String condition, int row_count): List<T> + doRetrieveByCondition(String condition, int offset, int row_count): List<T> + doRetrieveByKey(String key): T + doRetrieveAll(): List<T> + doRetrieveAll(int row_count): List<T> + doRetrieveAll(int offset, int row_count): List<T> + doSave(T obj): boolean + doSave(HashMap<String, ?> values): boolean + doUpdate(HashMap<String, ?> values, String condition): boolean + doSaveOrUpdate(T obj): boolean + doDelete(String condition): boolean
Invariante di classe	/

Nome metodo	+ doRetrieveByCondition(String condition): List<T>
Descrizione	Questo metodo restituisce una lista di oggetti dal database che verificano una determinata condizione.
Pre-condizione	context: DAO::doRetrieveByCondition(String condition) pre: condition != null
Post-condizione	/

OBJECT DESIGN DOCUMENT - OPENMEEET

Nome metodo	+ doRetrieveByCondition(String condition, int row_count): List<T>
Descrizione	Questo metodo restituisce una lista al massimo n oggetti (n = row_count) dal database che verificano una determinata condizione.
Pre-condizione	context: DAO::doRetrieveByCondition(String condition, int row_count) pre: condition != null && row_count > 0
Post-condizione	/
Nome metodo	+ doRetrieveByCondition(String condition, int offset, int row_count): List<T>
Descrizione	Questo metodo restituisce una lista al massimo n oggetti saltando i primi m (n = row_count, m = offset) dal database che verificano una determinata condizione.
Pre-condizione	context: DAO::doRetrieveByCondition(String condition, int offset, int row_count) pre: condition != null && row_count > 0 && offset >= 0
Post-condizione	/
Nome metodo	+ doRetrieveByKey(String key): T
Descrizione	Questo metodo restituisce un oggetto data la chiave primaria.
Pre-condizione	context: DAO::doRetrieveByKey(String key) pre: key != null
Post-condizione	/
Nome metodo	+ doRetrieveAll(): List<T>
Descrizione	Questo metodo restituisce una lista di tutti gli oggetti dal Database.
Pre-condizione	/
Post-condizione	/
Nome metodo	+ doRetrieveAll(int row_count): List<T>
Descrizione	Questo metodo restituisce una lista di n (n = row_count) oggetti dal Database.
Pre-condizione	context: DAO::doRetrieveAll(int row_count) pre: row_count > 0
Post-condizione	/



OBJECT DESIGN DOCUMENT - OPENMEEET

Nome metodo	+ doRetrieveAll(int offset, int row_count): List<T>
Descrizione	Questo metodo restituisce una lista di n oggetti saltando i primi m (n = row_count, m = offset) dal Database.
Pre-condizione	context: DAO::doRetrieveAll(int offset, int row_count) pre: row_count > 0 && offset >= 0
Post-condizione	/

Nome metodo	+ doSave(T obj): boolean
Descrizione	Questo metodo salva un oggetto nel Database.
Pre-condizione	context: DAO::doSave(T obj) pre: obj != null
Post-condizione	context: DAO::doSave(T obj) post: DAO.doRetrieveByKey(obj.id) != null

Nome metodo	+ doSave(HashMap<String, ?> values): boolean
Descrizione	Questo metodo salva un oggetto nel Database popolando soltanto i campi contenuti in values.
Pre-condizione	context: DAO::doSave(HashMap<String, ?> values) pre: values != null
Post-condizione	context: DAO::doSave(HashMap<String, ?> values) post: DAO.doRetrieveByKey(values.get(id)) != null

Nome metodo	+ doUpdate(HashMap<String, ?> values, String condition): boolean
Descrizione	Questo metodo aggiorna alcuni o tutti i campi degli oggetti nel Database che verificano una determinata condizione.
Pre-condizione	context: DAO::doUpdate(HashMap<String, ?> values, String condition) pre: values != null && condition != null
Post-condizione	context: DAO::doUpdate(HashMap<String, ?> values, String condition) post: DAO.doRetrieveByKey(values.get(id)) != null



OBJECT DESIGN DOCUMENT - OPENMEEET

Nome metodo	+ doSaveOrUpdate(T obj): boolean
Descrizione	Questo metodo salva un oggetto nel Database se se non esiste, altrimenti lo aggiorna.
Pre-condizione	context: DAO::doSaveOrUpdate(T obj) pre: obj != null
Post-condizione	context: DAO::doSaveOrUpdate(T obj) post: DAO.doRetrieveByKey(values.get(id)) != null

Nome metodo	+ doDelete(String condition): boolean
Descrizione	Questo metodo cancella un oggetto dal database se verifica una determinata condizione.
Pre-condizione	context: DAO::doDelete(String condition) pre: condition != null
Post-condizione	context: DAO::doDelete(String condition) post: DAO.doRetrieveByKey(values.get(id)) = null

Nome classe	GenericDAO
Descrizione	Questa classe permette di gestire le operazioni relative all'inserimento, modifica e cancellazione dei dati nel Database.
Metodi	+ genericDoRetrieveByCondition(String table, String condition, E extractor, DataSource source): List<T> + genericDoSave(String table, HashMap<String, ?> map, DataSource source): boolean + genericDoUpdate(String table, String condition, HashMap<String, ?> values, DataSource source): boolean + genericDoDelete(String table, String condition, DataSource source): boolean
Invariante di classe	/

Nome metodo	+ genericDoRetrieveByCondition(String table, String condition, E extractor, DataSource source): List<T>
Descrizione	Questo metodo restituisce una lista di oggetti dal database che verificano una determinata condizione.
Pre-condizione	context: GenericDAO::genericDoRetrieveByCondition(String table, String condition, E extractor, DataSource source) pre: table != null && condition != null && extractor != null && source != null
Post-condizione	/



OBJECT DESIGN DOCUMENT - OPENMEEET

Nome metodo	+ genericDoSave(String table, HashMap<String, ?> map, DataSource source): boolean
Descrizione	Questo metodo salva un oggetto nel Database.
Pre-condizione	context: GenericDAO::genericDoSave(String table, HashMap<String, ?> map, DataSource source) pre: table != null && map != null && source != null
Post-condizione	context: GenericDAO::genericDoSave(String table, HashMap<String, ?> map, DataSource source) post: DAO.doRetrieveByKey(obj.id) != null

Nome metodo	+ genericDoUpdate(String table, String condition, HashMap<String, ?> values, DataSource source): boolean
Descrizione	Questo metodo aggiorna alcuni o tutti i campi degli oggetti nel Database che verificano una determinata condizione.
Pre-condizione	context: GenericDAO::genericDoUpdate(String table, String condition, HashMap<String, ?> values, DataSource source) pre: table != null && condition != null && values != null
Post-condizione	context: GenericDAO::genericDoUpdate(String table, String condition, HashMap<String, ?> values, DataSource source) post: DAO.doRetrieveByKey(values.get(id)) != null

Nome metodo	+ doDelete(String table, String condition, DataSource source): boolean
Descrizione	Questo metodo cancella un oggetto dal database se verifica una determinata condizione.
Pre-condizione	context: GenericDAO::genericDoDelete(String table, String condition, DataSource source) pre: condition != null
Post-condizione	context: GenericDAO::genericDoDelete(String table, String condition, DataSource source) post: DAO.doRetrieveByKey(values.get(id)) = null



OBJECT DESIGN DOCUMENT - OPENMEEET

Nome classe	GenericProxyDAO
Descrizione	Questa classe permette di gestire le operazioni relative all'inserimento, modifica e cancellazione dei dati nel Database.
Metodi	+ genericProxyDoRetrieveByCondition(String condition, DAO<T> dao, PrintWriter out): List<T> + genericProxyDoRetrieveByCondition(String condition, int offset, int row_count, DAO<T> dao, PrintWriter out): List<T> + genericProxyDoRetrieveByKey(String key, DAO<T> dao, PrintWriter out): T + genericProxyDoSave(T entity, DAO<T> dao, PrintWriter out): boolean + genericProxyDoSave(HashMap<String, ?> values, DAO<T> dao, PrintWriter out): boolean + genericProxyDoUpdate(HashMap<String, ?> values, String condition, DAO<T> dao, PrintWriter out): boolean + genericProxyDoSaveOrUpdate(T entity, DAO<T> dao, PrintWriter out): boolean + genericProxyDoDelete(String condition, DAO<T> dao, PrintWriter out): boolean
Invariante di classe	/

Nome metodo	+ genericProxyDoRetrieveByCondition(String condition, DAO<T> dao, PrintWriter out): List<T>
Descrizione	Questo metodo restituisce una lista di oggetti dal database che verificano una determinata condizione.
Pre-condizione	context: GenericProxyDAO::genericProxyDoRetrieveByCondition(String condition, DAO<T> dao, PrintWriter out) pre: condition != null && dao != null && out != null
Post-condizione	/

Nome metodo	+ genericProxyDoRetrieveByCondition(String condition, int offset, int row_count, DAO<T> dao, PrintWriter out): List<T>
Descrizione	Questo metodo restituisce una lista di oggetti dal database che verificano una determinata condizione.
Pre-condizione	context: GenericProxyDAO::genericProxyDoRetrieveByCondition(String condition, int offset, int row_count, DAO<T> dao, PrintWriter out) pre: condition != null && offset >= 0 && row_count > 0 && dao != null && out != null
Post-condizione	/

OBJECT DESIGN DOCUMENT - OPENMEEET

Nome metodo	+ genericProxyDoRetrieveByKey(String key, DAO<T> dao, PrintWriter out): T
Descrizione	Questo metodo restituisce un oggetto data la chiave primaria.
Pre-condizione	context: GenericProxyDAO::genericProxyDoRetrieveByKey(String key, DAO<T> dao, PrintWriter out) pre: key != null && dao != null && out != null
Post-condizione	/

Nome metodo	+ genericProxyDoSave(T entity, DAO<T> dao, PrintWriter out): boolean
Descrizione	Questo metodo salva un oggetto nel Database.
Pre-condizione	context: GenericProxyDAO::genericProxyDoSave(T entity, DAO<T> dao, PrintWriter out) pre: entity != null && dao != null && out != null
Post-condizione	context: GenericProxyDAO::genericProxyDoSave(T entity, DAO<T> dao, PrintWriter out) post: DAO.doRetrieveByKey(entity.id) != null

Nome metodo	+ genericProxyDoSave(HashMap<String, ?> values, DAO<T> dao, PrintWriter out): boolean
Descrizione	Questo metodo salva un oggetto nel Database.
Pre-condizione	context: GenericProxyDAO::genericProxyDoSave(HashMap<String, ?> values, DAO<T> dao, PrintWriter out) pre: values != null && dao != null && out != null
Post-condizione	context: GenericProxyDAO::genericProxyDoSave(HashMap<String, ?> values, DAO<T> dao, PrintWriter out) post: DAO.doRetrieveByKey(values.get(id)) != null

Nome metodo	+ genericProxyDoUpdate(HashMap<String, ?> values, String condition, DAO<T> dao, PrintWriter out): boolean
Descrizione	Questo metodo aggiorna alcuni o tutti i campi degli oggetti nel Database che verificano una determinata condizione.
Pre-condizione	context: GenericProxyDAO::genericProxyDoUpdate(HashMap<String, ?> values, String condition, DAO<T> dao, PrintWriter out) pre: values != null && condition != null && dao != null && out != null
Post-condizione	context: GenericProxyDAO::genericProxyDoUpdate(HashMap<String, ?> values, String condition, DAO<T> dao, PrintWriter out) post: DAO.doRetrieveByKey(values.get(id)) != null



OBJECT DESIGN DOCUMENT - OPENMEEET

Nome metodo	+ genericProxyDoSaveOrUpdate(T entity, DAO<T> dao, PrintWriter out): boolean
Descrizione	Questo metodo salva un oggetto nel Database se se non esiste, altrimenti lo aggiorna.
Pre-condizione	context: GenericProxyDAO::genericProxyDoSaveOrUpdate(T entity, DAO<T> dao, PrintWriter out) pre: entity != null && dao != null && out != null
Post-condizione	context: GenericProxyDAO::genericProxyDoSaveOrUpdate(T entity, DAO<T> dao, PrintWriter out) post: DAO.doRetrieveByKey(values.get(id)) != null

Nome metodo	+ genericProxyDoDelete(String condition, DAO<T> dao, PrintWriter out): boolean
Descrizione	Questo metodo cancella un oggetto dal database se verifica una determinata condizione.
Pre-condizione	context: GenericProxyDAO::genericProxyDoDelete(String condition, DAO<T> dao, PrintWriter out) pre: condition != null && dao != null && out != null
Post-condizione	context: GenericProxyDAO::genericProxyDoDelete(String condition, DAO<T> dao, PrintWriter out) post: DAO.doRetrieveByKey(values.get(id)) = null

Nome classe	IEntity
Descrizione	Questa classe permette ad ogni oggetto Bean di essere trattato come una HashMap, semplificando l'inserimento e l'aggiornamento degli attributi del Bean nel database.
Metodi	+ toHashMap(): HashMap<String, ?> + toHashMap(String... fields): HashMap<String, ?>
Invariante di classe	/

Nome metodo	+ toHashMap(): HashMap<String, ?>
Descrizione	Questo metodo restituisce una HashMap contenente i campi del Bean che lo implementa.
Pre-condizione	/
Post-condizione	/



OBJECT DESIGN DOCUMENT - OPENMEEET

Nome metodo	+ toHashMap(String... fields): HashMap<String, ?>
Descrizione	Questo metodo restituisce una HashMap contenente esclusivamente i campi passati come parametri del Bean che lo implementa.
Pre-condizione	context: IEntity::toHashMap(String ...fields) pre: fields != null
Post-condizione	/

Nome classe	ResultSetExtractor
Descrizione	Questa classe permette ad ogni oggetto Bean di poter essere estratto da una query.
Metodi	+ extract(ResultSet resultSet): B
Invariante di classe	/

Nome metodo	+ extract(ResultSet resultSet): B
Descrizione	Questo metodo estrae il Bean da un ResultSet restituito da una query.
Pre-condizione	context: ResultSetExtractor::extract(ResultSet resultSet) pre: resultSet != null
Post-condizione	/

Nome classe	SQLDAO
Descrizione	Questa classe permette di condividere le istanza del DataSource.
Metodi	/
Invariante di classe	/

OBJECT DESIGN DOCUMENT - OPENMEET

3.2 Exceptions Package

Nome classe	InvalidPrimaryKeyException
Descrizione	Questa classe permette definire una particolare eccezione in caso di chiave primaria invalida.
Metodi	/
Invariante di classe	/

3.3 Helpers Package

Nome classe	JSONResponse
Descrizione	Questa classe permette di costruire una risposta di tipo JSON.
Metodi	+ getResponse(): HashMap<String, String> + addPair(String key, String value): void + getValue(String key): String + clear(): void + toString(): String
Invariante di classe	/

Nome metodo	+ getResponse(): HashMap<String, String>
Descrizione	Questo metodo restituisce la risposta sotto-forma di HashMap.
Pre-condizione	/
Post-condizione	/

Nome metodo	+ addPair(String key, String value): void
Descrizione	Questo metodo aggiunge una coppia (chiave, valore) alla risposta JSON.
Pre-condizione	context: JSONResponse::addPair(String key, String value) pre: key != null && value != null
Post-condizione	context: JSONResponse::addPair(String key, String value) post: response.size() = @pre.response.size() + 1



OBJECT DESIGN DOCUMENT - OPENMEEET

Nome metodo	+ getValue(String key): String
Descrizione	Questo metodo restituisce il valore associata alla chiave.
Pre-condizione	context: JSONResponse::getValue(String key) pre: key != null
Post-condizione	/

Nome metodo	+ clear(): void
Descrizione	Questo metodo svuota la risposta.
Pre-condizione	/
Post-condizione	context: JSONResponse::clear() post: response.size() == 0

Nome metodo	+ toString(): String
Descrizione	Questo metodo restituisce la risposta come un oggetto JSON.
Pre-condizione	/
Post-condizione	/

Nome classe	ResponseHelper
Descrizione	Questa classe contiene funzioni per facilitare l'invio di risposte HTTP al client.
Metodi	+ checkStringFields(String... parameters): boolean + sendGenericError(PrintWriter out): void + sendCustomError(PrintWriter out, String value): void + sendCustomSuccess(PrintWriter out, String value): void + sendGenericResponse(PrintWriter out, HashMap<String, String> pairs): void - writeResponse(PrintWriter out): void
Invariante di classe	/

OBJECT DESIGN DOCUMENT - OPENMEEET

Nome metodo	+ checkStringFields(String... parameters): boolean
Descrizione	Questo metodo controlla se i parametri della richiesta HTTP sono tutti inizializzati.
Pre-condizione	/
Post-condizione	/

Nome metodo	+ sendGenericError(PrintWriter out): void
Descrizione	Questo metodo invia un messaggio di errore generico come risposta HTTP al client.
Pre-condizione	context: ResponseHelper::sendGenericError(PrintWriter out) pre: out != null
Post-condizione	/

Nome metodo	+ sendCustomError(PrintWriter out, String value): void
Descrizione	Questo metodo invia un messaggio di errore specifico come risposta HTTP al client.
Pre-condizione	context: ResponseHelper::sendCustomError(PrintWriter out, String value) pre: out != null
Post-condizione	/

Nome metodo	+ sendCustomSuccess(PrintWriter out, String value): void
Descrizione	Questo metodo invia un messaggio di successo specifico come risposta HTTP al client.
Pre-condizione	context: ResponseHelper::sendCustomSuccess(PrintWriter out, String value) pre: out != null
Post-condizione	/

Nome metodo	+ sendGenericResponse(PrintWriter out, HashMap<String, String> pairs): void
Descrizione	Questo metodo invia un messaggio specifico come risposta HTTP al client.
Pre-condizione	context: ResponseHelper::sendGenericResponse(PrintWriter out, HashMap<String, String> pairs) pre: out != null && pairs.size() > 0
Post-condizione	/

OBJECT DESIGN DOCUMENT - OPENMEEET

Nome metodo	- writeResponse(PrintWriter out): void
Descrizione	Questo metodo scrive la risposta HTTP al client.
Pre-condizione	context: ResponseHelper::writeResponse(PrintWriter out) pre: out != null
Post-condizione	/

3.4 Utils Package

Nome classe	MultiMapList
Descrizione	Questa classe implementa una MultiMapList, ovvero una collezione di mappe dove ad ogni chiave potrebbero associati molteplici valori.
Metodi	+ toString(): String + getRowsNumber(): int + get(K key, int row): Collection<V> + getRow(int row): Collection<V> + addEntryInCurrentRow(Key k, V value): void + addCurrentRow(): void
Invariante di classe	/

Nome metodo	+ toString(): String
Descrizione	Questo metodo restituisce la MultiMapList sotto-forma di Stringa.
Pre-condizione	/
Post-condizione	/

Nome metodo	+ getRowsNumber(): int
Descrizione	Questo metodo restituisce il numero di righe contenute nella MultiMapList.
Pre-condizione	/
Post-condizione	/



OBJECT DESIGN DOCUMENT - OPENMEEET

Nome metodo	+ get(K key, int row): Collection<V>
Descrizione	Questo metodo restituisce i valori della chiave di una riga specifica.
Pre-condizione	context: MultiMapList::get(K key, int row) pre: row >= 0 && row < data.size()
Post-condizione	/

Nome metodo	+ getRow(int row): Collection<V>
Descrizione	Questo metodo restituisce i valori di una riga specifica.
Pre-condizione	context: MultiMapList::getRow(int row) pre: row >= 0 && row < data.size()
Post-condizione	/

Nome metodo	+ addEntryInCurrentRow(K key, V value): void
Descrizione	Questo metodo inserisce una nuova coppia (chiave, valore) nella riga corrente della MultiMapList.
Pre-condizione	/
Post-condizione	/

Nome metodo	+ addCurrentRow(): void
Descrizione	Questo metodo inserisce una nuova riga alla MultiMapList.
Pre-condizione	context: MultiMapList::addCurrentRow() pre: MultiMapList.addEntryInCurrentRow(key, value) && bufferMap != null
Post-condizione	/

Nome classe	PasswordEncoder
Descrizione	Questa classe viene usata per crittografare le password.
Metodi	+ sha1(String password): String
Invariante di classe	/



OBJECT DESIGN DOCUMENT - OPENMEEET

Nome metodo	+ sha1(): void
Descrizione	Questo metodo restituisce la password crittografata tramite sha1.
Pre-condizione	/
Post-condizione	/

Nome classe	QueryJoinExecutor
Descrizione	Questa classe viene usata per eseguire una query con la JOIN.
Metodi	+ doRetrieveByJoin(String query): MultiMapList<String, String>
Invariante di classe	/

Nome metodo	+ doRetrieveByJoin(String query): MultiMapList<String, String>
Descrizione	Questo metodo restituisce una MultiMapList per gestire i tipi diversi di Bean risultati dalla query JOIN.
Pre-condizione	context: QueryJoinExecutor::doRetrieveByJoin(String query) pre: query != null
Post-condizione	/



OBJECT DESIGN DOCUMENT - OPENMEET

Nome classe	QueryBuilder
Descrizione	Questa classe viene usata per la costruzione programmatica di una query SQL.
Metodi	+ SELECT(String... columns): QueryBuilder + FROM(String... tables): QueryBuilder + JOIN(String table): QueryBuilder + LEFT_JOIN(String table): QueryBuilder + RIGHT_JOIN(String table): QueryBuilder + ON(String predicate): QueryBuilder + WHERE(String predicate): QueryBuilder + ORDER_BY(String... columns): QueryBuilder + LIMIT(int row_count): QueryBuilder + LIMIT(int offset, int row_count): QueryBuilder + UNION(QueryBuilder queryBuilder): QueryBuilder + INSERT_INTO(String table, Map<String, ?> values): QueryBuilder + UPDATE(String table): QueryBuilder + SET(Map<String ?> values): QueryBuilder + DELETE_FROM(String table): QueryBuilder - encode(Object value): String + toString(): String
Invariante di classe	/

Nome metodo	+ SELECT(String... columns): QueryBuilder
Descrizione	Questo metodo restituisce un oggetto QueryBulder con la clausola SELECT aggiunta.
Pre-condizione	context: QueryBuilder::SELECT(String... columns) pre: columns != null
Post-condizione	/

Nome metodo	+ FROM(String... tables): QueryBuilder
Descrizione	Questo metodo restituisce un oggetto QueryBulder con la clausola FROM aggiunta.
Pre-condizione	context: QueryBuilder::FROM(String... tables) pre: QueryBuilder.SELECT(columns) && tables != null
Post-condizione	/



OBJECT DESIGN DOCUMENT - OPENMEEET

Nome metodo	+ JOIN(String table): QueryBuilder
Descrizione	Questo metodo restituisce un oggetto QueryBulder con la clausola JOIN aggiunta.
Pre-condizione	context: QueryBuilder::JOIN(String table) pre: QueryBuilder.FROM(tables) && table != null
Post-condizione	/

Nome metodo	+ LEFT_JOIN(String table): QueryBuilder
Descrizione	Questo metodo restituisce un oggetto QueryBulder con la clausola LEFT JOIN aggiunta.
Pre-condizione	context: QueryBuilder::LEFT_JOIN(String table) pre: QueryBuilder.FROM(tables) && table != null
Post-condizione	/

Nome metodo	+ RIGHT_JOIN(String table): QueryBuilder
Descrizione	Questo metodo restituisce un oggetto QueryBulder con la clausola RIGHT JOIN aggiunta.
Pre-condizione	context: QueryBuilder::RIGHT_JOIN(String table) pre: QueryBuilder.FROM(tables) && table != null
Post-condizione	/

Nome metodo	+ WHERE(String predicate): QueryBuilder
Descrizione	Questo metodo restituisce un oggetto QueryBulder con la clausola WHERE aggiunta.
Pre-condizione	context: QueryBuilder::WHERE(String predicate) pre: QueryBuilder.FROM(tables) && predicate != null
Post-condizione	/

Nome metodo	+ ORDER_BY(String... columns): QueryBuilder
Descrizione	Questo metodo restituisce un oggetto QueryBulder con la clausola ORDER BY aggiunta.
Pre-condizione	context: QueryBuilder::ORDER_BY(String... columns) pre: (QueryBuilder.FROM(tables) QueryBuilder.WHERE(predicate)) && columns != null
Post-condizione	/



OBJECT DESIGN DOCUMENT - OPENMEEET

Nome metodo	+ LIMIT(int row_count): QueryBuilder
Descrizione	Questo metodo restituisce un oggetto QueryBulder con la clausola LIMIT aggiunta.
Pre-condizione	context: QueryBuilder::LIMIT(int row_count) pre: (QueryBuilder.FROM(tables) QueryBuilder.WHERE(predicate) QueryBuilder.ORDER_BY(columns)) && row_count >= 0
Post-condizione	/

Nome metodo	+ LIMIT(int offset, int row_count): QueryBuilder
Descrizione	Questo metodo restituisce un oggetto QueryBulder con la clausola LIMIT aggiunta.
Pre-condizione	context: QueryBuilder::LIMIT(int row_count) pre: (QueryBuilder.FROM(tables) QueryBuilder.WHERE(predicate) QueryBuilder.ORDER_BY(columns)) && offset >= 0 && row_count >= 0
Post-condizione	/

Nome metodo	+ UNION(QueryBuilder queryBuilder): QueryBuilder
Descrizione	Questo metodo restituisce un oggetto QueryBulder con la clausola UNION aggiunta e unisce due oggetti QueryBuilder.
Pre-condizione	context: QueryBuilder::UNION(QueryBuilder queryBuilder) pre: queryBuilder != null
Post-condizione	/

Nome metodo	+ INSERT_INTO(String table, Map<String, ?> values): QueryBuilder
Descrizione	Questo metodo restituisce un oggetto QueryBulder con la clausola INSERT INTO.
Pre-condizione	context: QueryBuilder::INSERT_INTO(String table, Map<String, ?> values) pre: table != null && values != null
Post-condizione	/

Nome metodo	+ UPDATE(String table): QueryBuilder
Descrizione	Questo metodo restituisce un oggetto QueryBulder con la clausola UPDATE.
Pre-condizione	context: QueryBuilder::UPDATE(String table) pre: table != null
Post-condizione	/



OBJECT DESIGN DOCUMENT - OPENMEEET

Nome metodo	+ SET(Map<String, ?> values): QueryBuilder
Descrizione	Questo metodo restituisce un oggetto QueryBulder con la clausola UPDATE.
Pre-condizione	context: QueryBuilder::SET(Map<String, ?> values) pre: QueryBuilder.UPDATE(table) && values != null
Post-condizione	/

Nome metodo	+ DELETE_FROM(String table): QueryBuilder
Descrizione	Questo metodo restituisce un oggetto QueryBulder con la clausola DELETE FROM.
Pre-condizione	context: QueryBuilder::DELETE_FROM(String table) pre: table != null
Post-condizione	/

Nome metodo	- encode(Object value): String
Descrizione	Questo metodo converte un valore in una rappresentazione in stringa.
Pre-condizione	/
Post-condizione	/

Nome metodo	+ toString(): String
Descrizione	Questo metodo fornisce la rappresentazione di un oggetto QueryBuilder in stringa.
Pre-condizione	/
Post-condizione	/

OBJECT DESIGN DOCUMENT - OPENMEEET

4. Design Patterns

Nella presente sezione si andranno a descrivere e dettagliare i design patterns utilizzati nello sviluppo dell'applicativo OpenMeet. Per ogni pattern verranno forniti:

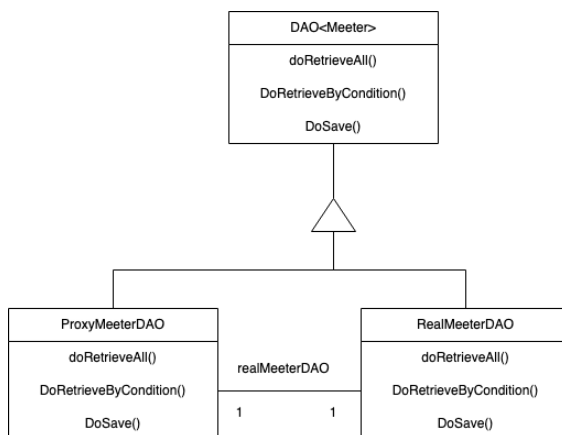
- Una brevissima introduzione teorica;
- Una descrizione del problema che doveva risolvere all'interno di OpenMeet;
- Una spiegazione di come si è risolto il problema in OpenMeet;
- Un grafico della struttura delle classi che implementano il pattern;

4.1 Proxy

Il Proxy è un design pattern che descrive come risolvere i problemi riguardanti l'accesso controllato agli oggetti e all'aggiunta di nuove funzionalità quando si accede ad un oggetto.

La soluzione proposta dal Proxy è quella di definire un oggetto "Proxy" che agisce come un sostituto dell'oggetto originario.

Il Proxy design pattern di OpenMeet getta le fondamenta per l'intera comunicazione tra client Mobile e Server remoto, in particolare alla parte di persistenza. Ciascuna richiesta al Database da parte del client, avviene attraverso una chiamata ad un **DAO** (Data Access Object) **Proxy**, che implementa la stessa identica interfaccia del vero DAO del Server. Per cui se nella logica del client è richiesto un `doRetrieveAll()`, questo chiamerà il metodo come farebbe il server, ottenendo lo stesso risultato. In realtà il *proxyDAO* invia richieste HTTP(S) al server specificando il tipo di DAO da richiamare ed eventuali altri parametri richiesti per le *Query*.



OBJECT DESIGN DOCUMENT - OPENMEET

Infine, come è possibile osservare nell'immagine, l'interfaccia implementata è un DAO tipizzato (*<type>*), ciò è dovuto alle numerosi classi di DAO (e conseguentemente, ProxyDAO) ed è, ovviamente, una scelta orientata al riuso.

5. Glossario

La seguente sezione contiene le definizioni dei termini utilizzati nel documento al fine di prevenire o risolvere qualsiasi ambiguità.

Sigla/Termine	Definizione
Package	Raggruppamento di classi ed interfacce.
DAO	Data Access Object, implementazione dell'omonimo pattern architettura che si occupa di fornire un accesso in modo astratto ai dati persistenti.
Servlet	Un oggetto scritto in linguaggio Java che opera all'interna di un server web permettendo la creazione di applicazioni web.
Service	Una classe che implementa la logica di business nei web services.
Proxy	Un oggetto che permette di gestire, tramite una serie di controlli, l'accesso a determinate risorse del sistema.