Università degli Studi di Salerno

Corso di Ingegneria del Software

Object Design Document 1.0

A.A. 2025/26

Progetto: EarthLocals	Versione: 1.0
Documento: Object Design Document	Data: 22/12/2025

Coordinatore del progetto:

Nome	Matricola

Partecipanti:

Nome	Matricola
Abbatiello Simone	0512119659
Niemiec Francesco	0512118999
Rega Maristella	0512119032
Squitieri Andrea	0512119008

Scritto da:	Abbatiello Simone, Niemiec Francesco, Rega Maristella, Squitieri Andrea

Revision History

Data	Versione	Descrizione	Autore
25/11/2025	0.1	Stesura iniziale dell'Object Design Document	Abbatiello Simone, Niemiec Francesco, Rega Maristella, Squitieri Andrea
30/11/2025	0.2	Definizioni, acronimi e abbreviazioni	Abbatiello Simone, Rega Maristella
5/12/2025	0.3	Gestione Utente	Abbatiello Simone, Niemiec Francesco, Rega Maristella, Squitieri Andrea
6/12/2025	0.4	Gestione Missione	Abbatiello Simone, Niemiec Francesco, Rega Maristella, Squitieri Andrea
10/12/2025	0.5	Gestione Candidature	Abbatiello Simone, Niemiec Francesco, Rega Maristella, Squitieri Andrea
12/12/2025	0.6	Gestione Recensioni	Niemiec Francesco, Andrea Squitieri
19/12/2025	0.7	Sotto consiglio del tutor e del professore definito Gestione Email	Rega Maristella
22/12/2025	1.0	Correzioni finali e fix estetici	Abbatiello Simone, Niemiec Francesco, Rega Maristella, Squitieri Andrea

Contenuti

1. Introduzione 3
1.1 Definizioni, acronimi e abbreviazioni 3
1.2 Riferimenti 3
2. Pacchetti 3
2.1 GestioneUtente 3
2.2 Gestione Missione 5
2.3 GestioneCandidature 8
2.4 GestioneRecensioni 9
2.5 GestioneEmail 11

1. Introduzione

1.1 Definizioni, acronimi e abbreviazioni

Nel seguente documento sono presenti diversi acronimi, qui riportiamo il significato di ciascuno di essi:

- MVC: Model View Controller.
- DB: Database.
- JSP: Servlet Pages.
- JDBC: Java DataBase Connectivity.
- RAD: Requirements Analysis Document.
- PS: Problem Statement.

1.2 Riferimenti

Per la stesura di questo documento si fa riferimento ai termini utilizzati ed ampiamente descritti nei documenti di Problem Statement, Requirements Analysis e System Design.

2. Pacchetti

2.1 GestioneUtente

Non sono presenti invarianti

Servizio / Operazione	Contratti
+ aggiungiUtente(Utente datiPersonali)	**Abstract**: Chiamato durante la registrazione per salvare il nuovo utente. **Precondizioni**: context GestioneUtente::aggiungiUtente(Utente datiPersonali) pre: !self.utenti->includes(datiPersonali) **Postcondizioni**:

	context GestioneUtente::aggiungiUtente(Utente datiPersonali) post: self.utenti->includes(datiPersonali) and datiPersonali.stato = PENDING
+ aggiungiModeratore(Moderatore datiPersonali)	**Abstract**: Chiamato per creare un nuovo utente moderatore. **Precondizioni**: context GestioneUtente::aggiungiModeratore(Moderatore datiPersonali) pre: !self.utenti->includes(datiPersonali) **Postcondizioni**: context GestioneUtente::aggiungiModeratore(Moderatore datiPersonali) post: self.utenti->includes(datiPersonali) and datiPersonali.stato = CONFIRMED
+ finalizzaUtente(Utente datiFinalizzazione)	**Abstract**: Chiamato per confermare la registrazione dopo il click sul link email. **Precondizioni**: context GestioneUtente::finalizzaUtente(Utente datiFinalizzazione) pre: self.utenti->includes(datiPersonali) and datiPersonali.stato = PENDING **Postcondizioni**: context GestioneUtente::finalizzaUtente(Utente datiFinalizzazione) post: datiPersonali.stato = CONFIRMED
+ verificaAccessoUtente(String email, String password) -> Utente	**Abstract**: Verifica le credenziali al login. Restituisce utenteTrovato o eccezioni. **Postcondizioni**: context GestioneUtente::verificaAccessoUtente(String email, String password) -> Utente post: self.utenti->exists(u| u.email = email and u.password = password and result = u) or result = null
+ modificaProfilo(Utente modificato) -> Utente	**Abstract**: Aggiorna i dati del profilo utente. **Precondizioni**: context GestioneUtente::modificaProfilo(Utente modificato) -> Utente pre: self.utenti->exists(u| u.id = modificato.id) **Postcondizioni**: context GestioneUtente::modificaProfilo(Utente

	modificato) -> Utente post: self.utenti->includes(modificato)
+ modificaPasswordProfilo(Utente user, String pwCorrente, String pwNuova)	**Abstract**: Gestisce il cambio password, verificando la vecchia e la validità della nuova. **Precondizioni**: context GestioneUtente::modificaPasswordProfilo(Utente user, String pwCorrente, String pwNuova) -> Utente pre: self.utenti->includes(user) and user.password = pwCorrente **Postcondizioni**: context GestioneUtente::modificaPasswordProfilo(Utente user, String pwCorrente, String pwNuova) -> Utente post: user.password = pwNuova and self.utenti->includes(user)
+ recuperaPassword(String email) -> String	**Abstract**: Avvia il recupero password verificando l'esistenza dell'email. **Precondizioni**: context GestioneUtente::recuperaPassword(String email) -> String pre: self.utenti->exists(u|u.email = email) **Postcondizioni**: context GestioneUtente::recuperaPassword(String email) -> String post: self.utenti->exists(u|u.email = email and u.pwTemp != null and u.scadenzaPwTemp = (currentTime() + TEMPO_MAX)
+ getProfiloUtente(Integer riferimento) -> Utente	**Abstract**: Recupera i dati pubblici del profilo di un altro utente (es. organizzatore). **Precondizioni**: context GestioneUtente::getProfiloUtente(Integer riferimento) -> Utente pre: self.utenti->exists(u|u.riferimento = riferimento)

2.2 Gestione Missione

Non sono presenti invarianti

Servizio / Operazione	Contratto

+ processaNuovaMissione(Missione datiMissione)	**Abstract**: Metodo del controller che orchestra la creazione. Chiama internamente aggiungiMissione. **Precondizioni**: context GestioneMissione::processaNuovaMissione(Missione datiMissione) pre: !self.missioni->includes (datiMissione) **Postcondizioni**: context GestioneMissione::processaNuovaMissione(Missione datiMissione) post: self.missioni->includes (datiMissione) and datiMissione.stato = PENDING
+ aggiungiMissione(Missione datiMissione)	**Abstract**: Salva la missione nel catalogo (stato iniziale Pending). **Precondizioni**: context GestioneMissione::aggiungiMissione(Missione datiMissione) pre: !self.missioni->includes (datiMissione) **Postcondizioni**: context GestioneMissione::aggiungiMissione(Missione datiMissione) post: self.missioni->includes (datiMissione) and datiMissione.stato = PENDING
+ approvaMissione(Missione missione)	**Abstract**: Il moderatore approva la missione. Cambia stato e invia l'email. **Precondizioni**: context GestioneMissione::approvaMissione(Missione missione) pre: self.missioni->includes (datiMissione) and datiMissione.stato = PENDING **Postcondizioni**: GestioneMissione::approvaMissione(Missione missione) post: self.missioni->includes (datiMissione) and datiMissione.stato = CONFIRMED
+ rifiutaMissione(Missione missione)	**Abstract**: Il moderatore rifiuta la missione.

	Precondizioni: context GestioneMissione::rifiuta Missione(Missione datiMissione) pre: self.missioni->includes (datiMissione) **Postcondizioni**: context GestioneMissione::rifiuta Missione(Missione datiMissione) post: datiMissione.stato = DENIED
+ annullaMissione(Missione missione)	**Abstract**: L'organizzatore annulla la missione. **Precondizioni**: context GestioneMissione::annulla Missione(Missione datiMissione) pre: self.missioni->includes (datiMissione) **Postcondizioni**: context GestioneMissione::annulla Missione(Missione datiMissione) post: !self.missioni->includes (datiMissione)
+getMissioniFiltrate(Collection<Predicate> listaFiltri) -> Collection<Missione>	**Abstract**: Restituisce la lista delle missioni che soddisfano i criteri di ricerca. **Postcondizioni**: context GestioneMissione::getMissioniFiltrate(Collection<Predicate> listaFiltri) -> Collection<Missione> post: self.missioni->forAll (m| listaFiltri.forAll(f| f->test(m)) => result->includes(m))
+ getInfoMissione(Integer id) -> Missione	**Abstract**: Recupera i dettagli di una missione (usato anche prima di una recensione). **Precondizioni**: context GestioneMissione::getInfoMissione(Integer id) -> Missione pre: self.missioni->exists(m|m.id = id) **Postcondizioni**: context GestioneMissione::getInfoMissione(Integer id) -> Missione post: self.missioni->exists(m|m.id = id and result = m)
+ visualizzaMissioni(Organizzatore organizzatore) -> Collection<Missione>	**Abstract**: Recupera le missioni dell'organizzatore per la sua area personale. **Precondizioni**:

	context GestioneMissione::visualizzaMissioni(Organizzatore organizzatore) -> Collection<Missione> pre: self.missioni.organizzatori->includes(organizzatore) **Postcondizioni**: context GestioneMissione::visualizzaMissioni(Organizzatore organizzatore) -> Collection<Missione> post: self.missioni->forAll(m| m.organizzatore = organizzatore => result.includes(m))

2.3 GestioneCandidature

Non sono presenti invarianti

Servizio / Operazione	Contratto
+ aggiungiCandidatura(Candidatura candidatura)	**Abstract**: Crea una nuova istanza di candidatura associando il volontario alla missione scelta. **Precondizioni**: context GestioneCandidature::aggiungiCandidature(Candidatura candidatura) pre: !self.candidature->includes(candidatura) **Postcondzioni**: context GestioneCandidature::aggiungiCandidature(Candidatura candidatura) post: self.candidature->includes(candidatura) and candidature.stato = PENDING
+ rimuoviCandidatura(Candidatura candidatura)	**Abstract**: Elimina (o annulla) una candidatura esistente su richiesta del volontario. **Precondizioni**: context GestioneCandidature::rimuoviCandidatura(Candidatura candidatura) pre: self.candidature ->includes(candidatura) **Postcondizioni**: context GestioneCandidature::rimuoviCandidatura(Candidatura candidatura) post: !self.candidature ->includes(candidatura)

+ accettaCandidatura(Candidatura candidatura)	**Abstract**: Registra l'approvazione del volontario da parte dell'organizzatore. **Precondizioni**: context GestioneCandidature::accettaCandidatura(Candidatura candidatura) pre: self.candidature ->includes(candidatura) and candidature.stato = PENDING **Postcondizioni**: context GestioneCandidature::accettaCandidatura(Candidatura candidatura) post: self.candidature ->includes(candidatura) and candidature.stato = CONFIRMED
+ rifiutaCandidatura(Candidatura candidatura)	**Abstract**: Registra il rifiuto della candidatura da parte dell'organizzatore. **Precondizioni**: context GestioneCandidature::rifiutaCandidature(Candidatura candidatura) pre: self.candidature->includes(candidatura) and candidature.stato = PENDING **Postcondzioni**: context GestioneCandidature::rifiutaCandidature(Candidatura candidatura) post: candidature.stato = DENIED

2.4 GestioneRecensioni

Non sono presenti invarianti

Servizio / Operazione	Contratto
+aggiungiRecensioneOrganizzatore(Recensione recensione)	**Abstract**: Salva una recensione scritta da un volontario per un organizzatore. **Precondizioni**: context GestioneRecensioni::aggiungiRecensioneOrganizzatore(Recensione recensione) pre: !self.recensioni->includes(recensione) **Postcondizioni**:

	context GestioneRecensioni::aggiungiRecensioneOrganizzatore(Recensione recensione) post: self.recensioni->includes(recensione) and recensione.stato = PUBLIC
+aggiungiRecensioneVolontario(Recensione recensione)	**Abstract**: Salva una recensione scritta da un organizzatore per un volontario. **Precondizioni**: context GestioneRecensioni::aggiungiRecensioneVolontario(Recensione recensione) pre: !self.recensioni->includes(recensione) **Postcondizioni**: context GestioneRecensioni::aggiungiRecensioneVolontario(Recensione recensione) post: self.recensioni->includes(recensione) and resensione.stato = PUBLIC
+ rimuoviRecensione(Recensione recensione)	**Abstract**: Elimina una recensione segnalata ritenuta inappropriata dal moderatore. **Precondizioni**: context GestioneRecensioni::rimuoviRecensione(Recensione recensione) pre: self.recensioni->includes(recensione) **Postcondizioni**: context GestioneRecensioni::rimuoviRecensione(Recensione recensione) post: !self.recensioni->includes(recensione)
+ approvaRecensione(Recensione recensione)	**Abstract**: Conferma la validità di una recensione segnalata, mantenendola pubblicata. **Precondizioni**: context GestioneRecensioni::approvaRecensione(Recensione recensione) pre: self.recensioni->includes(recensione) and recensione.stato = REPORTED

	Postcondzioni: context GestioneRecensioni::approvaRecensione(Recensione recensione) post: self.recensioni->includes(recensione) and recensione.stato = PUBLIC
+ segnalaRecensione(Recensione recensione)	**Abstract**: Segnala una recensione. **Precondizioni**: context GestioneRecensioni::segnalaRecensione(Recensione recensione) pre: self.recensioni->includes(recensione) and recensione.stato = PUBLIC **Postcondizioni**: GestioneRecensioni::segnalaRecensione(Recensione recensione) post: self.recensioni->includes(recensione) and recensione.stato = REPORTED

2.5 GestioneEmail

Non sono presenti invarianti

Servizio / Operazione (da Diagrammi)	Contratto
+ inviaConferma(Messaggio messaggioCreazione)	Invia link di attivazione account.
+ inviaEmailRecupero(Messaggio messaggio)	Invia credenziali per reset password.
+ inviaEmailCandidatura(Messaggio messaggio)	Notifica nuova candidatura all'organizzatore.
+ inviaEmailAnnullamento(Messaggio messaggio)	Notifica annullamento candidatura (dal volontario).
+ inviaEmailAccettazione(Messaggio messaggio)	Notifica accettazione candidatura.
+ inviaEmailRifiuto(Messaggio messaggio)	Notifica rifiuto candidatura.
+ invioEmailApprovazione(Messaggio messaggio)	Notifica approvazione missione.
+ invioEmailViolazione(Messaggio messaggio)	Notifica rimozione recensione.
+ inviaEmailCreazione(Messaggio messaggioCreazione)	Invia credenziali al nuovo moderatore creato.