Object Design Document

SmartBlog

Progetto di Ingegneria del Software 2020/2021

Felice De Chiara	0512105866
Vincenzo Emanuele Martone	0512105758
Antonio Russo	0512106058
Alfonso Graziano	0512105776

Sommario

1. Introduzione

1.1 Object Design Trade-Offs

Data la necessità di implementare l'applicazione con costi ridotti e tempi minimi abbiamo scelto di cercare delle soluzioni che consentano la prototipazione rapida. Nello specifico utilizziamo dei componenti off-the-shelf che rendano più rapido il processo di sviluppo.

Abbiamo però valutato di limitare l'utilizzo di tali componenti solo al front-end in quanto non vogliamo rinunciare a realizzare un back-end customizzato.

1.1.1 Componenti Off-the-Shelf

Il front-end verrà realizzato con il framework *React*.
Quest'ultimo è una soluzione molto utilizzata negli ultimi anni nello sviluppo di *web app* e *progressive web app*. Grazie a questo strumento possiamo realizzare il front-end in maniera fortemente disaccoppiata dal back-end.
Il front-end comunicherà con il back-end attraverso l'utilizzo di *API REST*, il che ci può consentire in un futuro di realizzare ad esempio un'applicazione nativa o un qualsiasi altro tipo di Client senza modificare il Server.
Insieme a React verrà utilizzata la sua libreria grafica *Ant Design* per utilizzare componenti già esistenti.

1.1.2 Design Patterns

Per favorire una maggiore minimalità e riusabilità del codice abbiamo deciso di utilizzare i seguenti design pattern.

1.1.2.1 Façade

Ogni sottosistema che andiamo a realizzare implementa parte della logica di business dell'intero sistema. Per fornire un accesso più compatto ai

sottosistemi, abbiamo deciso di mascherare la business logic tramite dei Façade, implementati dalle classi Manager a cui i Control effettueranno l'accesso. Tali classi Manager incapsulano tutta la business logic, invocando le operazioni delle classi che effettuano l'accesso al database e delle classi Entity.

1.2 Linee guida per la documentazione dell'interfaccia

1.2.1 Nomenclatura delle componenti

È richiesto agli sviluppatori di seguire le seguenti linee guida al fine di essere consistenti nell'intero progetto e facilitare la comprensione delle funzionalità di ogni componente.

1.2.1.1 Nomi delle classi

- Ogni classe deve avere nome in *UpperCamelCase*
- Ogni classe deve avere nome univoco
- Ogni classe di tipo entità deve avere un nome sostantivo per essere associata ad un'entità del dominio
- Ogni classe che modella la logica di business deve avere nome composto dal pacchetto per cui espone servizi seguito da "Manager"
- Ogni classe che modella la connessione al database e le relative query deve avere nome composto dall'entità che va a modellare seguito da "*Dao*"
- Ogni classe che modella un servizio offerto via web deve avere un nome composto dal nome del servizio che espone seguito dal suffisso "*Control*"

1.2.1.2 Nomi dei metodi

- Ogni metodo deve avere nomi in *lowerCamelCase*

1.2.1.3 Nomi delle eccezioni

- Ogni eccezione deve avere un nome che esplicita il problema che viene segnalato

1.2.1.4 Nomi delle pagine front-end

- Ogni pagina deve avere nome in UpperCamelCase
- Ogni pagina deve avere un nome composto dal nome della pagina seguito da "*Page*"

1.2.1.4 Nomi dei form front-end

- Ogni form deve avere nome in UpperCamelCase
- Ogni form deve avere un nome composto dal suo tipo seguito da "*Form*" (es. SignUpForm)

1.2.1.5 Organizzazione delle componenti

- Tutte le classi che realizzano un sottosistema devono essere racchiuse nello stesso package
- Tutte le componenti che realizzano l'interfaccia grafica devono essere collocate in *\reactClient\src\app*

1.2.1.5 Organizzazione del codice

- Il codice deve essere indentato tramite tabulazione corrispondente a quattro spazi
- L'apertura di un blocco di codice deve avvenire nella stessa riga in cui è definita la signature del metodo o il nome della classe
- Il codice front-end (React) deve soddisfare gli standard principali utilizzati dalla comunità

1.3 Definizioni, acronimi ed abbreviazioni

- OCL → Object Constraint Language
- JDBC → Java Database Connectivity
- MVC → Model – View – Controller
- API → Application Programming Interface
- DAO → Data Access Object

1.4 Relazioni con gli altri documenti

Viene fatto riferimento al *System Design Document* per la suddivisione in sottosistemi, al *Requirement Analysis Document* per il *Class Diagram*

2. Packages

Presentiamo la divisione in sottosistemi e l'organizzazione del codice in file.

2.1 Divisione in pacchetti

Come definito nel *System Design Document*, il sistema si basa su un'architettura *three-tier* con *MVC*.
Il sistema di accesso ai dati persistenti viene realizzato implementando il *Dao* con un DataSource che si collega a JDBC.

La suddivisione in package con le relative classi è la seguente:

- *control* (Servlets):
  - *AddReviewControl.java*
  - *AllSpecControl.java*
  - *CreateSpecControl.java*
  - *DeleteSpecControl.java*
  - *DoraControl.java*
  - *ProfileControl.java*
  - *ReviewControl.java*
  - *ReviewInspectionControl.java*
  - *SearchControl.java*
  - *SetScoresControl.java*
  - *SignInControl.java*
  - *SignUpControl.java*
  - *SpecControl.java*
- *dora* (business logic del sottositema DoraIA):
  - *genetic*
    - *Crossover.java*
    - *FitnessHelper.java*
    - *FitnessHelperSpec.java*

        - *GEHelper.java*
        - *Mutation.java*
        - *OPTCalculator.java*
        - *Population.java*
        - *RandomMutation.java*
        - *Selection.java*
        - *SinglePointCrossover.java*
        - *SpecGene.java*
        - *TruncationSelection.java*
    - *DoraManager.java*
- *filter* (Interceptors):
    - *Error.java*
    - *Message.java*
    - *ResponseFormatter.java*
    - *RestrictedToManager.java*
    - *RestrictedToReviewer.java*
    - *RestrictedToUser.java*
- *review* (business logic del sottosistema Gestione Recensioni):
    - *Review.java*
    - *ReviewDao.java*
    - *ReviewManager.java*
    - *ReviewMismatchException.java*
- *security* (codifica/decodifica token):
    - *JWTHandler.java*
- *spec* (business logic del sottosistema Gestione Schede Tecniche):
    - *Spec.java*
    - *SpecDao.java*
    - *SpecManager.java*
    - *SpecMismatchException.java*
    - *EmptyFieldException.java*
    - *PriceException.java*
- *user* (business logic del sottosistema Gestione Utenza):
    - *User.java*
    - *UserDao.java*
    - *UserManager.java*

- *Manager.java*
- *Reviewer.java*
- *CredentialsException.java*

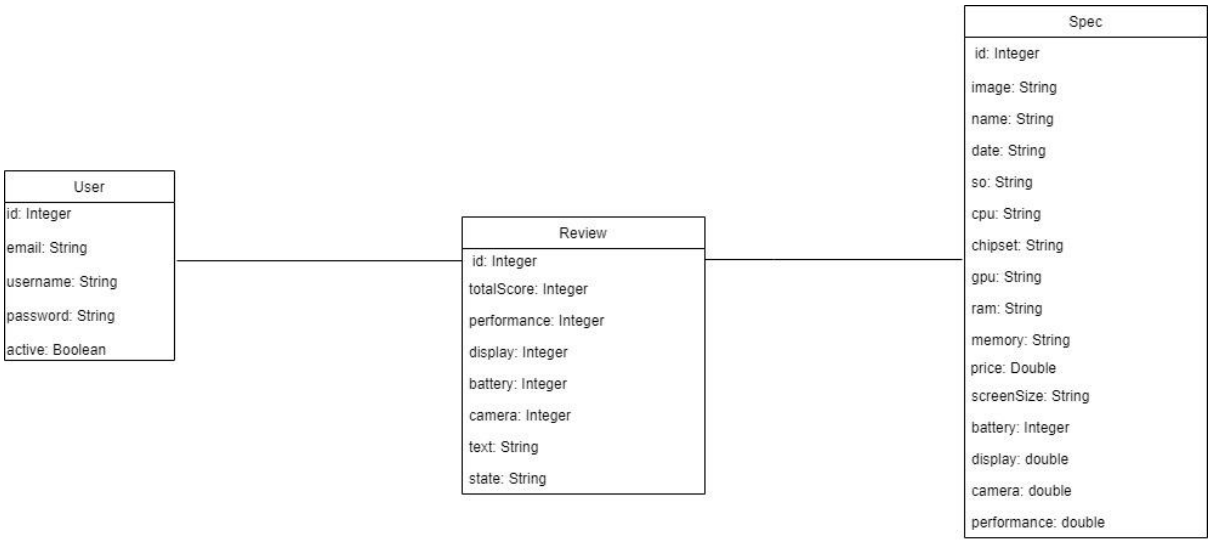
2.2 Organizzazione del codice in file

- Ogni classe sarà collocata nel relativo file.
- Ogni pacchetto avrà come prefisso *it.unisa.di.smartblog.*
- Ogni package sarà mappato nel relativo percorso *src/main/java/it/unisa/di/smartblog*
- L'interfaccia utente che sarà all'interno della cartella *\reactClient\src\app*

3. Interfacce delle classi

3.1 Trasformazione class diagram in modello relazionale

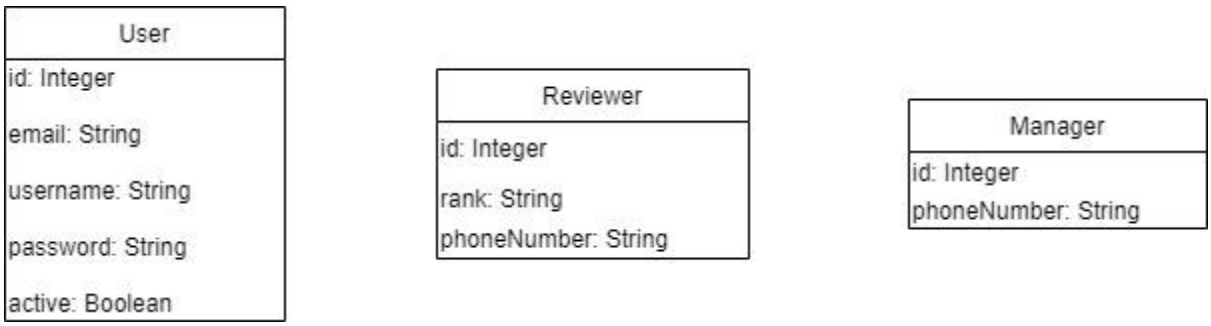
A partire dal *Class Diagram* prodotto durante la fase di *Requirement Analysis*, è necessario effettuare un'operazione di ristrutturazione al fine di mappare tale modello su un modello relazionale.
In primo luogo, abbiamo accorpato la tabella *SpecScore* sulla tabella *Spec*, distribuendo gli attributi della prima tabella sulla seconda in quanto la relazione tra le due tabelle è *1 a 1*.
La classe associativa *Review* è stata trasformata in una tabella che si trova in relazione *1 a molti* con *User* e con *Spec*.



Successivamente ci siamo posti il problema di sciogliere la generalizzazione presente tra le tabelle *User*, *Reviewer* e *Manager*. Abbiamo valutato due opzioni:

1. Creare 3 tabelle separate in cui *Reviewer* e *Manager* contengono gli attributi specializzati e una chiave esterna verso la tabella *User*
2. Creare 3 tabelle separate ognuna delle quali contiene tutti gli attributi

Scegliendo la prima opzione avremmo privilegiato la *manutenibilità* in quanto la modifica della tabella *User*, quindi la modifica degli attributi che sono in comune tra le diverse classi, non avrebbe implicato la modifica delle altre tabelle. Scegliendo la seconda opzione avremmo privilegiato le *prestazioni* in quanto per accedere ai dati completi di un *Reviewer* o di un *Manager* non avremmo dovuto effettuare alcun *Join* invece necessario nel primo caso.
Abbiamo optato per la prima opzione in quanto sono rari i casi in cui abbiamo bisogno di accedere ai dati di un *Reviewer* e di un *Manager* rispetto ai casi in cui abbiamo bisogno dei dati di uno User.
Il razionale di questa scelta, dunque, è stato il privilegio della *manutenibilità*.



Abbiamo, inoltre, ritenuto opportuna l'introduzione di una chiave artificiale su alcune tabelle come *Spec*, *User* e *Reviewer* (in tutti i 3 casi, la chiave artificiale è stata chiamata *id*).

3.2 Specifica interfacce

Mostriamo ora i metodi con *pre-condizioni*, *post-condizioni* ed *invariante* per ogni classe. Ogni tabella è realizzata con sintassi OCL.

3.2.1 Bean

Nome Classe	User
Sottosistema	GestioneUtenza
Variabili di istanza	- username:String - password:String - email:String - reviews:List<Review> - id: Integer - active: Boolean
Descrizione	Questa entity contiene i metodi per creare un oggetto di tipo User, con i relativi metodi per ottenere e

	impostare i suoi parametri
Signature metodi	+ User(): User + User(username: String, password: String, email: String, reviews: List<Review>): User + getUsername(): String + getPassword() :String + getEmail(): String + getReviews(): List<Review> + getId(): Integer + isActive(): Boolean + getReviews(): List<Review> + setUsername(username: String): void + setPassword(password: String): void + setEmail(email: String): void + setId(id: Integer): void + setActive(active: Boolean) + addReview(review:Review): Boolean
Pre-Condizioni	N/A
Post-Condizioni	● context User()::User **post:** reviews->size=0 ● context User::User(username, password, email, reviews) **post:** result.username=username and result.email=email and result.password=password and result.reviews=reviews ● context User::getUsername() **post:** result=this.username ● context User::getPassword() **post:** result=this.password ● context User::getEmail() **post:** result=this.email ● context User::getReviews() **post:** result=this.reviews ● context User::getId() **post**: result=this.id ● context User::isActive() **post:** result=this.active

	• context User::setUserName(username) **post:** this.username=username • context User::setPassword(password) **post:** this.password=password • context User::setEmail(email) **post:** this.email=email • context User::setId(id) **post:** this.id=id • context User:: setActive(active) **post:** this.active=active • context User::addReview(review) **post:** reviews->includes(review)
Invarianti	N/A

Nome Classe	Reviewer <<extends>> User
Sottosistema	GestioneUtenza
Variabili di Istanza	- phoneNumber: String - rank: String
Descrizione	Questa entity contiene i metodi per creare un oggetto di tipo Reviewer, con i relativi metodi per ottenere e impostare i suoi parametri. Deriva da User
Signature metodi	+ Reviewer(): Reviewer + Reviewer(username: String, password: String, email: String, reviews: List<Reviews>, phoneNumber: String, rank: String): Reviewer + getPhoneNumber(): String + getRank(): String + setPhoneNumber(phoneNumber: String): void + setRank(rank: String): void
Pre-	N\A

Condizioni	
Post-Condizioni	• context Reviewer()::Reviewer **post**: N/A • context Reviewer::Reviewer(username, password, email, reviews, phoneNumber, rank) **post:** result.username=username and result.email=email and result.password=password and result.reviews=reviews and result.phoneNumber=phoneNumber and result.rank=rank • context Reviewer::getPhoneNumber() **post:** result=this.phoneNumber • context Reviewer::getRank() **post:** result=this.rank • context Reviewer::setPhoneNumber(phoneNumber) **post:** this.phoneNumber=phoneNumber • context Reviewer::setRank(rank) **post:** this.rank=rank

Nome Classe	Manager <<extends>> User
Sottosistema	GestioneUtenza
Variabili di istanza	- phoneNumber:String
Descrizione	Questa entity contiene i metodi per creare un oggetto di tipo Manager, con i relativi metodi per ottenere e impostare i suoi parametri. Deriva da User
Signature metodi	+ Manager(): Manager + Manager(username: String, password: String, email: String, reviews: List<Review>, phoneNumber: String) + getPhoneNumber(): String + setPhoneNumber(phoneNumber: String): void
Pre-Condizioni	N\A
Post-	• context Manager:: Manager()

Condizioni	**post:** N/A ● context Manager::Manager(username, password, email, reviews, phoneNumber) **post:** result.username=username and result.email=email and result.password=password and result.reviews=reviews and result.phoneNumber=phoneNumber ● context Manager::getPhoneNumber() **post**: result=this.phoneNumber ● context Manager::setPhoneNumber(phoneNumber) **post**: this.phoneNumber=phoneNumber
Invarianti	N/A

Nome Classe	Spec
Sottosistema	GestioneSchedeTecniche
Variabili di istanza	- id: Integer - image: String - name: String - date: String - so: String - cpu: String - chipset: String - gpu: String - ram: String - memory: String - battery: Integer - screenSize: String - price: Double - reviewer: Reviewer - display: Double - camera: Double - performance: Double

	- reviews: List<Review> - fit: Double - normalizedBattery: Double
Descrizione	Questa entity contiene i metodi per creare un oggetto di tipo Spec, con i relativi metodi per ottenere e impostare i suoi parametri
Signature metodi	+ Spec(): Spec + Spec(image: String, name: String, date: String, so: String, cpu: String, chipset: String, gpu: String, ram: String, memory: String, battery: Integer, screenSize: String, price: Double): Spec + getId(): Integer + getImage(): String + getName(): String + getDate(): String + getSo(): String + getCpu(): String + getChipset(): String + getGpu(): String + getRam(): String + getMemory(): String + getBattery(): Integer + getScreenSize(): String + getPrice(): Double + getReviewer(): Reviewer + getDisplay(): Double + getCamera(): Double + getPerformance(): Double + getReviews() List<Review> + getFitValue(): Double + getNormalizedBattery: Double() + setId(id: Integer):void + setImage(image: String): void + setName(name: String): void + setDate(date: String): void + setSo(so: String): void + setCpu(cpu: String): void + setChipset(chipset: String): void + setGpu(gpu: String): void + setRam(ram: String): void

	+ setMemory(memory :String): void + setBattery(battery: Integer): void + setScreenSize(screenSize: String): void + setPrice(price: Double): void + setReviewer(reviewer: Reviewer): void + setDisplay(display: Double): void + setCamera(camera: Double): void + setPerformance(performance: Double): void + setReviews(reviews: List<Review>) + setFitValue(fit: Double) + setNormalizedBattery(normalizedBattery): Double() + addReview(review:Review): Boolean
Pre-Condizioni	N/A
Post-Condizioni	● context Spec:: Spec() **post:** N/A ● context Spec::Spec(image: String, name: String, date: String, so: String, cpu: String, chipset: String, gpu: String, ram: String, memory: String, battery: Integer, screenSize: String, price: Double): Spec **post:** result.id=id and result.image=image and result.name=name and result.date=date and result.so=so and result.cpu=cpu and result.chipset=chipset and result.gpu=gpu and result.ram=ram and result.memory=memory and result.battery=battery and result.screnn Size=screenSize and result.price=price and result.reviewer=reviewer ● context Spec::getId() **post:** result=this.id ● context Spec::getImage() **post:** result=this.image ● context Spec::getName() **post:** result=this.name ● context Spec::getDate() **post:** result=this.date ● context Spec::getSo() **post:** result=this.so

	• context Spec::getCpu() **post:** result=this.cpu • context Spec::getChipset() **post:** result=this.chipset • context Spec::getGpu() **post:** result=this.gpu • context Spec::getRam() **post:** result=this.ram • context Spec::getMemory() **post:** result=this.memory • context Spec::getBattery() **post:** result=this.battery • context Spec::getScreenSize(): **post:** result=this.screenSize • context Spec::getPrice(): **post:** result=this.price • context Spec::getReviewer() **post:** result=this.reviewer • context Spec::getDisplay() **post:** result=this.display • context Spec::getCamera() **post:** result=this.camera • context Spec::getPerformance() **post:** result=this.performance • context Spec::getReviews() **post:** result=this.reviews • context Spec::getFit() **post:** result=this.fit • context Spec::getNormalizedBattery() **post:** result=this.normalizedBattery • context Spec::setId(id) **post:** this.id=id • context Spec::setImage(image) **post:** this.image=image • context Spec::setName(name) **post:** this.name=name • context Spec::setDate(date) **post:** this.date=date • context Spec::setSo(so) **post:** this.so=so • context Spec::setCpu(cpu) **post:** this.cpu=cpu

	• context Spec::setChipset(chipset) **post:** this.chipset=chipset • context Spec::setGpu(gpu) **post:** this.gpu=gpu • context Spec::setRam(ram) **post:** this.ram=ram • context Spec::setMemory(memory) **post:** this.memory=memory • context Spec::setBattery(battery) **post:** this.battery=battery • context Spec::setScreenSize(screenSize) **post:** this.screenSize=screenSize • context Spec::setPrice(price) **post:** this,price=price • context Spec::setReviewer(reviewerId) **post:** this.reviewer=reviewer • context Spec::setDisplay(display) **post:** this.display=display • context Spec::setCamera(camera) **post:** this.camera=camera • context Spec::setPerformance(performance) **post:** this.performance=performance • context Spec::setReviews(reviews) **post:**this.reviews=reviews • context Spec::setFit(fit) **post:**this.fit=fit • context Spec::setNormalizedBattery(normalizedBattery) **post:** this.normalizedBattery=normalizedBattery • context Spec::addReview(review) **post:** reviews->includes(review)
Invarianti	N/A

Nome Classe	Review
Sottosistema	GestioneRecensioni
Variabili di istanza	- id: Integer - state: String - totalScore: Integer - battery: Integer - performance: Integer - display: Interger - camera: Integer - text: String - user: User - spec: Spec
Descrizione	Questa entity contiene i metodi per creare un oggetto di tipo Review, con i relativi metodi per ottenere e impostare i suoi parametri
Signature metodi	+ Review(): Review + Review(totalScore: Integer, battery: Integer, performance: Integer, display: Integer, camera: Integer, text: String, user: User, spec: Spec):

	Review + getId(): Integer + getState(): String + getTotalScore(): Integer + getBattery(): Integer + getPerformance(): Integer + getDisplay(): Integer + getCamera(): Integer + getText(): String + getUser(): User + getSpec(): Spec + setId(): void + setState(state: String): void + setTotalScore(totalScore: Double): void + setBattery(battery: Double): void + setPerformance(performance: Double): void + setDisplay(display: Double): void + setCamera(camera: Double): void + setText(text: String): void + setUser(user: User): void + setSpec(spec: Spec): void
Pre-Condizioni	N\A
Post-Condizioni	● context Review:: Review() **post:** N/A ● context Review::Review(state, totalScore, battery, preformance, display, camera, text, user, spec): Review **post**: result.state=state and result.totalScore=totalScore and result.battery=battery and result.performance=performance and result.display=display and result.camera=camera and result.text=text and result.user=user and result.spec=spec ● context Review::getId() **post**: result=this.id ● context Review::getState() **post**: result=this.state ● context Review::getTotalScore()

	post: result=this.totalScore • context Review::getBattery() **post**: result=this.battery • context Review::getPerformance() **post**: result=this.performance • context Review::getDisplay() **post**: result=this.display • context Review::getCamera() **post**: result=this.camera • context Review::getText() **post**: result=this.text • context Review::getUser() **post**: result=this.user • context Review::getSpec() **post**: result=this.spec • context Review::setId(id) **post**: this.id=id • context Review::setState(state) **post**: this.state=state • context Review::setTotalScore(totalScore) **post**: this.totalScore=totalScore • context Review::setBattery(battery) **post**: this.battery=battery • context Review::setPerformance(performance) **post**: this.performance=performance • context Review::setDisplay(display) **post**: this.display=display • context Review::setCamera(camera) **post**: this.camere=camera • context Review::setText(text) **post**: this.text=text • context Review::setUser(user) **post:** this.user=user • context Review::setSpec(spec) **post:** this.spec=spec
Invarianti	N/A

Nome classe	Error
Sottosistema	**Control**
Variabili di istanza	- message: String
Descrizione	Classe che unifica i messaggi di errore
Signature Metodi	+ Error(message: String): Error + getMessage(): String + setMessage(message : String)
Pre-condizioni	**N/A**
Post-Condizioni	● context Error:: Error(message) **post:**result.message=message ● context Error:: getMessage() **post:** result=this.getMessage ● context Error:: setMessage(message) **post:** result=this.message

Nome classe	Message
Sottosistema	**Control**
Variabili di istanza	- message: String
Descrizione	Classe con lo scopo di unificare i messaggi
Signature Metodi	+ Message(message: String): Message + getMessage(): String + setMessage(message : String)
Pre-condizioni	**N/A**
Post-Condizioni	● context Message:: Error(message) **post:**result.message=message ● context Message:: getMessage() **post:** result=this.getMessage ● context Message:: setMessage(message) **post:** result=this.message

3.2.2 Manager

Nome classe	ReviewManager
Sottosistema	GestioneRecensioni
Variabili di istanza	- dao: ReviewDao
Descrizione	Questo manager contiene la logica di business che gestisce tutte le Review. Accede alle informazioni all'interno del database utilizzando un ReviewDao
Signature Metodi	+ createReview(votoComplessivo: Integer, prestazioni: Integer, display: Integer, batteria: Integer, fotocamera: Integer, text: String, spec: Spec, user: User): Boolean + searchReviewsByUser(id: Integer): List<Review> + searchReviewsBySpec(id: Integer): List<Review> + searchPendingReviews(): List<Review> + searchReviewInfo(id: Integer): Review + approvation(id: Integer, approved: Boolean):

	Boolean
Pre-Condizioni	• context ReviewManager::createReview(votoComplessivo, display, batteria, fotocamera, descrizione,spec,user) **pre**: votoComplessivo > 0 and votoComplessivo <= 5 and display > 0 and display <= 5 and camera > 0 and camera <= 5 and batteria > 0 and batteria <= 5 and text->size > 0 and text->size <= 200 and spec <> null and user <> null • context ReviewManager::searchReviewsByUser(id) **pre**: N/A • context ReviewManager::searchReviewsBySpec(id) **pre**: N/A • context ReviewManager:: searchPendingReviews() **pre**: N/A • context ReviewManager::searchReviewInfo(id) **pre**: N/A • context ReviewManager::approvation(id, approved) **pre**: N/A
Post-Condizioni	• context ReviewManager::createReview(votoComplessivo, display, batteria, fotocamera, descrizione, spec, user) **post:** result = dao.saveReview(Review(votoComplessivo, display, batteria, fotocamera, descrizione, spec, user)) • context ReviewManager::searchReviewsByUser(id) **post:** result = dao.getByUser(id) • context ReviewManager::searchReviewsBySpec(id) **post:** result = dao.getBySpec(id) • context ReviewManager::searchPendingReviews() **post:** result = dao.getPending()

	● context ReviewManager::searchReviewInfo(id) **post:** result = dao.getById(id) ● context ReviewManager::approvation(id, approved) **post:** dao.getById(id).status = approved
Invariante	N/A

Nome classe	SpecsManager
Sottosistema	GestioneSchedeTecniche
Variabili di istanza	- dao: SpecDao
Descrizione	Questo manager contiene la logica di business che gestisce tutte le Spec. Accede alle informazioni all'interno del database utilizzando un SpecsDao
Signature Metodi	+ searchByName(name: String): List<Spec> + searchAll(): List<Spec> + searchById(id: Integer): Spec + searchMinBattery(): Integer + searchMaxBattery(): Integer + searchByPrice(price: Double): List<Spec> + setScores(reviewerId: Integer, performance: Double, display: Double, camera: Double, specId: Integer): Boolean + createSpec(deviceName: String, releaseDate:

	String, OS: String, CPU: String, chipset: String, GPU: String, RAM: String, internalMemory:String, battery: Integer, displayInches: String, price: Double, image: String): Boolean - checkSpecValidation(deviceName: String, releaseDate: String, OS: String, CPU: String, chipset: String, GPU: String, RAM: String, internalMemory:String, battery: Integer, DisplayInches: String, price: Double): Boolean + deleteSpec(id: Integer): Boolean
Pre-Condizioni	● context SpecsManager::searchByName(name) **pre:** N/A ● context SpecsManager::searchAll() **pre:** N/A ● context SpecsManager::searchById(id) **pre:** N/A ● context SpecsManager::searchMinBattery() **pre:** N/A ● context SpecsManager::searchMaxBattery() **pre:** N/A ● context SpecsManager::searchByPrice(price) **pre:** N/A ● context SpecsManager::setScores(reviewerId, performance, display, camera, specId) **pre:** performance >= 0 and performance <= 5 and display >= 0 and display <= 5 and camera >= 0 and camera <= 5 ● context SpecsManager::createSpec(deviceName, releaseDate, OS, CPU, chipset, GPU, RAM, internalMemory, battery, displayInches, price, image): Boolean **pre:** SpecManager.checkSpecValidation(deviceName, releaseDate, OS, CPU, chipset, GPU, RAM, internalMemory, battery, displayInches, price, image)=true ● context SpecsManager::checkSpecValidation(deviceName, releaseDate, OS, chipset, GPU, RAM, internalMemory, battery, displayInches, price)

	pre: N/A ● context SpecsManager::deleteSpec(id: Integer) **pre:** N/A
Post-Condizioni	● context SpecsManager::searchByName(name) **post:** result = dao.getByName(name) ● context SpecsManager::searchAll() **post:** result = dao.getAll() ● context SpecsManager::searchById(id) **post:** result = dao.getById(id) ● context SpecsManager::searchMinBattery() **post:** result = dao.getMinBattery() ● context SpecsManager::searchMaxBattery() **post:** result = dao.getMaxBattery() ● context SpecsManager::searchByPrice(price) **post:** result=dao.getByPrice(price) ● context SpecsManager::setScores(id, performance, display, camera, specId) **post:** result = dao.updateSpecScores(id, performance, display, camera, specId) ● context SpecsManager::createSpec(deviceName, releaseDate, OS, chipset, GPU, RAM, internalMemory, battery, displayInches,price) **post:** result = dao.saveSpec(Spec(deviceName, releaseDate, OS, chipset, GPU, RAM, internalMemory, battery, displayInches,price)) ● context SpecsManager::deleteSpec(id) **post:** result = dao.deleteSpec(id)
Invariante	N/A

Nome classe	UserManager
Sottosistema	GestioneUtenza
Variabili di istanza	- dao: UserDao
Descrizione	Questo manager contiene la logica di business che

	gestisce gli utenti. Accede alle informazioni all'interno del database utilizzando un UserDao
Signature Metodi	- checkCredentialsFormat(username: String, email: String, password: String, repeatPassword: String): Boolean - emailAlreadyUsed(email: String): Boolean + createUser(username: String, email: String, password: String, repeatPassword: String): Boolean + auth(email: String, password: String): User + isManager(user: User): Manager + isReviewer(user: User): Reviewer + getUserInfoByEmail(email: String):User
Pre-Condizioni	● context UserManager:: checkCredentialsFormat(username, email, password, repeatPassword) **pre:** N/A ● context UserManager::emailAlreadyUsed(email) **pre:** email <> null ● context UserManager:: createUser(username,email,password) **pre:** emailAlreadyUsed(email)=false and checkCredentialsFormat(username, email, password, repeatPassword) = false ● context UserManager::auth(email, password) **pre:** password <> null and password=dao.getByEmail(email).password ● context UserManager::isManager(user) **pre:** N/A ● context UserManager::isReviewer(user) **pre:** N/A ● context UserManager::getUserInfoByEmail(email) **pre:** N/A

Post-Condizioni	● context UserManager::emailAlreadyUsed(email)
	post:
	if users->exists(u|u.email=email) then
	result=true
	else
	result=false

	endif ● context UserManager::createUser(username: String, email: String, password: String, repeatPassword: String) **post:** result=dao.saveUser(User(username, email, password)) ● context UserManager::saveUser(username: String, email: String, password: String, repeatPassword: String) **post:** users->exists(u|u.username=username and u.email=email and u.password=password) ● context UserManager::auth(email, password) **post:** result = dao.getByEmail(email) ● context UserManager::isManager(user) **post:** result=dao.getManager(user) ● context UserManager::isReviewer(user) **post:** result=dao.getReviewer(user) ● context UserManager::getUserInfoByEmail(email) **post:** result = dao.getByEmail(email)
Invariante	N/A

Nome classe	DoraManager
Sottosistema	DoraIA
Variabili di istanza	N/A
Descrizione	Questo manager contiene al suo interno la logica di business per collegarsi al modulo di intelligenza artificiale DoraIA. Oltre all'algoritmo genetico di DoraIA è collegato a SpecManager per alcune funzionalità
Signature Metodi	+ findSpecsByParams(battery: Double, performance: Double, camera: Double, display: Double, maxBudget: Double): ArrayList<Spec> - singleRun(fp: FitnessHelper, specs: ArrayList<Spec>): SpecGene
Pre-	● context

Condizioni	DoraManager::findSpecsByParams(battery, performance, camera, display, maxBudget) **pre:** N/A • context DoraManager::singleRun(fp, specs) **pre:** N/A
Post-Condizioni	• context DoraManager::findSpecsByParams(battery, performance, camera, display, maxBudget) **post:** result->size() = 6 • context DoraManager::singleRun(fp, specs) **post:** N/A
Invariante	N/A

3.2.3 Dao

Nome classe	UserDAO
Sottosistema	GestioneUtenza
Variabili di istanza	- ds: DataSource
Descrizione	Questo DAO ci consente di interfacciarsi al database tramite un datasource JDBC e gestire tutte le query riguardanti gli utenti
Signature Metodi	+ getByEmail(email: String): User + saveUser(user: User): Boolean + getManager(user: User): Manager + getReviewer(user: User): Reviewer

Pre-Condizioni	● context UserDAO::getByEmail(email) **pre:** email <> null and users->exists(u|u.email=email) ● context UserDAO::saveUser(user) **pre:** user <> null ● context UserDAO::getManager(user) **pre:** user <> null ● context UserDAO::getReviewer(user) **pre:** user <> null

Post-Condizioni	● context UserDAO::getByEmail(email)
	post: result = users.select(u|u.email=email)
	● context UserDAO::saveUser(user)
	post: users->exists(u|u.email=user.email)
	● context UserDAO::getManager(user)
	post:
	if(managers->includes(user))
	then result=managers->select(m|m.id=user.id)
	● context UserDAO::getReviewer(user)
	post:
	if(reviewers>includes(user))
	then result=reviewers>select(r|r.id=user.id)

Invariante	N/A

Nome classe	SpecDao
Sottosistema	GestioneSchedeTecniche
Variabili di istanza	- ds: DataSource
Descrizione	Questo DAO ci consente di interfacciarsi al database tramite un datasource JDBC e gestire tutte le query riguardanti le Spec
Signature Metodi	+ getByName(name: String): List<Spec> + getAll(): List<Spec> + getById(id: Integer): Spec + saveSpec(spec: Spec): Boolean + deleteSpec(id: Integer): Boolean

	+ updateSpecScores(reviewerId: Integer, performance: Double, display: Double, camera: Double, specId: Integer): Boolean + getByPrice(price: Double): List<Spec> + getMaxBattery(): Integer + getMinBattery(): Integer
Pre-Condizioni	● context SpecDao::getByName(name) **pre:** name <> null and specs->exists(s|s.name=name) ● context SpecDao::getAll() **pre:** N/A ● context SpecDao::getById(id) **pre:** specs->exists(s|s.id=id) ● context SpecDao::saveSpec(spec) **pre:** spec <> null ● context SpecDao::deleteSpec(id) **pre:** specs->exists(s|s.id=id) ● context SpecDao::updateSpecScores(reviewerId, performance: double, display: double, camera: double, specId:int) **pre:** N/A ● context SpecDao::getByPrice(price: Double) **pre:** price > 0 ● context SpecDao::getMaxBattery() **pre:** N/A ● context SpecDao::getMinBattery() **pre:** N/A
Post-Condizioni	● context SpecDao::getByName(name) **post:** result = specs -> select(s|s.name -> includes(name)) ● context SpecDao::getAll() **post:** result=specs ● context SpecDao::getById(id) **post:** result = specs -> select(s|s.id=id) ● context SpecDao::saveSpec(spec) **post:** specs -> exists(s|s.id=spec.id) ● context SpecDao::deleteSpec(id) **post:** specs -> not exists(s|s.id=id) ● contextSpecDao::updateSpecScores(display, performance, camera, specId)

	post: result = specs -> exists(s|s.id=id and s.display=display and s.performance=performance and s.camera=camera) ● context SpecDao::getByPrice(price: Double) **post:** result=specs->select(s|s.price<= price) ● context SpecDao::getMaxBattery() **post:** specs->forAll(s|s.battery<=result) ● context SpecDao::getMinBattery() **post:** specs->forAll(s|s.battery>=result)
Invariante	specs.size >= 0

Nome classe	ReviewDao
Sottosistema	GestioneRecensioni
Variabili di istanza	- ds: DataSource
Descrizione	Questo DAO ci consente di interfacciarsi al database tramite un datasource JDBC e gestire tutte le query riguardanti gli utenti
Signature Metodi	+ getById(id: Integer): Review + getBySpecId(id:int): List<Review> + getByUser(userId: Integer): List<Review> + getPending(): List<Review>

	+ saveReview(review: Review): Boolean + deleteReview(id: int): boolean + approveReview(id: int, approved: Boolean): Boolean
Pre-Condizioni	● context ReviewDao::getById(id) **pre:** reviews->exists(r|r.id=id) ● context ReviewDao::getBySpecId(id) **pre:** reviews->exists(r|r.specId=id) ● context ReviewDao::getByUser(userId) **pre:** userId<>null and reviews->exists(r|r.userId=userId) ● context ReviewDao::getPending() **pre:** N/A ● context ReviewDao::saveReview(review) **pre:** review <> null ● context ReviewDao::deleteReview(id) **pre:** reviews->exists(r|r.id=id) ● context ReviewDao::approveReview(id, approved) pre: reviews->exists(r|r.id=id)
Post-Condizioni	● context ReviewDao::getById(id) **post:** result=reviews->select(r|r.id=result.id) ● context ReviewDao::getBySpecId(id) **post:** result=reviews->select(r|r.specId=spec.id) ● context ReviewDao::getbyUser(email) **post:** result=reviews->select(r|r.userId=user.id) ● context ReviewDao::getPending() **post:** result=reviews->select(r|r.status=”pending”) ● context ReviewDao::saveReview(review): **post:** reviews->includes(review) ● context ReviewDao::deleteReview(id) **post:** reviews->not exists(r|r.id=id) ● context ReviewDao::approveReview(id, approved) **post** reviews->exists(r|r.id=id and r.state=approved)
Invariante	N/A

3.2.4 Control

Nome classe	**AddReviewControl**
Sottosistema	**Control**
Variabili di istanza	- rm: ReviewManager - um: UserManager - sm SpecManager
Descrizione	Questo Control gestisce l'aggiunta di una Review da parte di uno User
Signature Metodi	+ doPost(request: HttpServletRequest, response: HttpServletResponse): void
Pre-condizioni	● context AddReviewControl::doPost(request, response) **pre:** RestrictedToUser.doFilter(request, response, chain) = true
Post-condizioni	● context AddReviewControl::doPost(request, response) **post:** result = response

Nome classe	**AllSpecControl**
Sottosistema	**Control**
Variabili di istanza	- sm: SpecManager
Descrizione	Questo control gestisce la raccolta di tutte le spec
Signature Metodi	+ doGet(request: HttpServletRequest, response: HttpServletResponse): void
Pre-	N/A

condizioni	
Post-Condizioni	● context AddReviewControl::doGet(request, response) **post:** result = response

Nome classe	**CreateSpecControl**
Sottosistema	**Control**
Variabili di istanza	- sm: SpecManager
Descrizione	Questo control gestisce creazione della Spec da parte del Manager
Signature Metodi	+ doPost(request: HttpServletRequest, response: HttpServletResponse): void
Pre-condizioni	● context CreateSpecControl::doPost(request, response): **pre:** RestrictedToManager.doFilter(request, response, chain) = true
Post-Condizioni	● context CreateSpecControl::doPost(request, response): **post:** result = response

Nome classe	**DeleteSpecControl**
Sottosistema	**Control**
Variabili di istanza	- sm: SpecManager
Descrizione	Questo control gestisce la cancellazione di una spec
Signature Metodi	+ doGet(request: HttpServletRequest, response: HttpServletResponse)
Pre-condizioni	● context DeleteSpecControl::doGet(request, response):

	pre: RestrictedToManager.doFilter(request, response, chain) = true
Post-Condizioni	● context DeleteSpecControl::doGet(request, response): **post:** result=response

Nome classe	**DoraControl**
Sottosistema	**Control**
Variabili di istanza	**N/A**
Descrizione	Questo control gestisce l'interazione con il tool di Intelligenza Artificiale
Signature Metodi	+ doGet(request: HttpServletRequest, response: HttpServletResponse)
Pre-condizioni	**N/A**
Post-Condizioni	● context DoraControll:: doGet(request, response) **post**:result=response

Nome classe	**ProfileControl**
Sottosistema	**Control**
Variabili di istanza	- rm :ReviewManager - um :UserManager
Descrizione	Questo control gestisce la creazione della pagina User
Signature Metodi	+ doGet(request: HttpServletRequest, response: HttpServletResponse)
Pre-condizioni	● context ProfileControl:: doGet(request, response) **pre:** RestrictedToUser.doFilter(request,

	response, chain) = true
Post-Condizioni	● context ProfileControl:: doGet(request, response) **post:** result=response

Nome classe	**ReviewControl**
Sottosistema	**Control**
Variabili di istanza	- rm: ReviewManager
Descrizione	Questo control gestisce la raccolta delle Review in stato “pending” da parte del Reviewer
Signature Metodi	+ doGet(request: HttpServletRequest, response: HttpServletResponse)
Pre-condizioni	● context ReviewControl:: doGet(request, response) **pre N/A**
Post-Condizioni	● context ReviewControl:: doGet(request, response) **post:** result = response

Nome classe	**ReviewInspectionControl**
Sottosistema	**Control**
Variabili di istanza	- rm: ReviewManager
Descrizione	Questo control gestisce l’accettazione o il rifiuto di una Review
Signature Metodi	+ doGet(request: HttpServletRequest, response: HttpServletResponse)

Pre-condizioni	● context ReviewInspectionControl:: doGet(request, response) **pre:** RestrictedToReviewer.doFilter(request, response, chain)
Post-Condizioni	● context ReviewInspectionControl:: doGet(request, response) **post:** result=response

Nome classe	**SearchControl**
Sottosistema	**Control**
Variabili di istanza	- sm: SpecManager
Descrizione	Questo control gestisce la cancellazione di una spec
Signature Metodi	+ doGet(request: HttpServletRequest, response: HttpServletResponse)
Pre-condizioni	**N/A**
Post-Condizioni	● context SearchControl:: doGet(request, response) **post:** result=response

Nome classe	**SetSpecScoreControl**
Sottosistema	**Control**
Variabili di istanza	**N/A**
Descrizione	Questo control gestisce l’accesso alla aggiunta di un punteggio
Signature Metodi	+ doPost(request: HttpServletRequest, response: HttpServletResponse)
Pre-condizioni	**N/A**
Post-Condizioni	● context SearchControl:: doPostt(request, response) **post:** result=response

Nome classe	**SignInControl**
Sottosistema	**Control**
Variabili di istanza	- **um: UserManager**
Descrizione	Questo control gestisce l’autenticazione di uno User
Signature Metodi	+ doGet(request: HttpServletRequest, response: HttpServletResponse)
Pre-condizioni	**N/A**
Post-Condizioni	● context SignInControl:: doGet(request, response) **post:** result=response

Nome classe	**SignUpControl**
Sottosistema	**Control**
Variabili di istanza	- **um: UserManager**
Descrizione	Questo control gestisce la registrazione da parte di User
Signature Metodi	+ doGet(request: HttpServletRequest, response: HttpServletResponse)
Pre-condizioni	**N/A**
Post-Condizioni	● context SignUpControl:: doGet(response,request) **post:** result=response

Nome classe	**SpecControl**
Sottosistema	**Control**
Variabili di istanza	- **sm: SpecManager** - **rm: RevieweManager**
Descrizione	Questo control crea la pagina della Spec selezionata da un User
Signature Metodi	+ doGet(request: HttpServletRequest, response: HttpServletResponse)
Pre-condizioni	N/A
Post-Condizioni	● context SpecControll:: doGet(request, response)

Nome classe	ResponseFormatter <<implement>> Filter
Sottosistema	Control
Variabili di istanza	- gson: Gson
Descrizione	
Signature Metodi	+ init(confing: FilterConfig): void + doFilter(req: ServletRequest, resp: ServletResponse, chain):void + sendAsJson(response: ServletResponse, Obj: Object): void
Pre-condizioni	context ResponseFormatter:: init(config) **pre:**N/A context ResponseFormatter:: doFilter(req, resp, chain) **pre:**N/A context ResponseFormatter::sendAsJson(response, Obj) **pre:**N/A
Post-Condizioni	context ResponseFormatter::init(config) **post:**this.gson=config context ResponseFormatter::doFilter(req,resp,chain) **post:**chain.doFilter(req,resp) context ResponseFormatter::sendAsJson(response, obj) **post:** N/A

Nome classe	RestrictedToManager <<implement>> Filter
Sottosistema	Control
Variabili di istanza	- jwt: JWTHandler
Descrizione	
Signature Metodi	+ init(confing: FilterConfig) :void + doFilter(req: ServletRequest, resp: ServletResponse, chain: FilterChain) :void + errorResponse() :void
Pre-condizioni	● context RestrictedToManager:: init(config) **pre:** N/A ● context RestrictedToManager:: doFilter(req, resp) **pre:** N/A ● context RestrictedToManager:: errorResponse() **pre:** N/A
Post-Condizioni	● context RestrictedToManager:: init(config) **post:**this.jwt=config ● context RestrictedToManager:: doFilter(req, resp) **post:**chain.doFilter(req,resp) ● context RestrictedToManager:: errorResponse() **post:** N/A

Nome classe	RestrictedToReviewer <<implement>> Filter
Sottosistema	Control
Variabili di istanza	- jwt: JWTHandler
Descrizione	
Signature Metodi	+ init(confing: FilterConfig): void + doFilter(req: ServletRequest, resp: ServletResponse, chain: FilterChain) :void + errorResponse() :void
Pre-condizioni	● context RestrictedToReviewer:: init(config) **pre:**N/A ● context RestrictedToReviewer:: doFilter(req, resp) **pre:**N/A ● context RestrictedToReviewer:: errorResponse() **pre:**N/A
Post-Condizioni	● context RestrictedToReviewer:: init(config) **post:**this.jwt=config ● context RestrictedToReviewer:: doFilter(req, resp) **post:**chain.doFilter(req,resp) ● context RestrictedToReviewer:: errorResponse() **post:** N/A

Nome classe	RestrictedToUser <<implement>> Filter
Sottosistema	Control
Variabili di istanza	- jwt: JWTHandler
Descrizione	
Signature Metodi	+ init(confing: FilterConfig):void + doFilter(req: ServletRequest, resp: ServletResponse, chain: FilterChain): void + errorResponse(): void
Pre-condizioni	● context RestrictedToUser:: init(confing) **pre:**N/A ● context RestrictedToUser:: doFilter(req, resp) **pre:**N/A ● context RestrictedToUser:: errorResponse() **pre:**N/A
Post-Condizioni	● context RestrictedToUser:: init(confing) **post:** this.jwt=config ● context RestrictedToUser:: doFilter(req, resp,chain) **post:**chain.doFilter(req,resp) ● context RestrictedToUser:: errorResponse() **post:** N/A

3.2.5 Exception

Nome classe	SpecMismatchException <<extend>> Exception
Sottosistema	GestioneSpec
Descrizione	Questa classe modella l'eccezione che viene lanciata nel momento in cui un Manager inserisce i dati di una Spec in un formato non valido
Signature Metodi	+ SpecMismatchException(message: String)
Pre-condizioni	**N\A**
Post-Condizioni	• context SpecMismatchException::SpecMismatchException (message) **post**:result.message=message

Nome classe	EmptyFieldException <<extend>> Exception
Sottosistema	GestioneSpec
Descrizione	Questa classe modella l'eccezione che viene lanciata nel momento in cui un utente lascia il campo della ricerca vuoto
Signature Metodi	+ EmptyFieldException(message: String)
Pre-condizioni	**N\A**
Post-Condizioni	context EmpyFieldException:: EmpyFieldException(message) **post**:result.message=message

Nome classe	PriceException <<extend>> Exception
Sottosistema	GestioneSpec
Descrizione	Questa classe modella l'eccezione che viene lanciata nel momento in cui un utente inserisce un prezzo negativo
Signature Metodi	+ PriceException(message: String)
Pre-condizioni	**N\A**
Post-Condizioni	● context PriceException:: PriceException(message) **post**:result.message=message

Nome classe	ReviewMismatchException <<extend>> Exception
Sottosistema	GestioneReview
Descrizione	Questa classe modella l'eccezione che viene lanciata nel momento in cui un utente inserisce una Review avente campi non validi
Signature Metodi	+ ReviewMismatchException(message: String)
Pre-condizioni	**N\A**
Post-Condizioni	● context ReviewMismatchException:: ReviewMismatchException(message) **post**:result.message=message

Nome classe	CredentialsException <<extend>> Exception
Sottosistema	GestioneUtenza
Descrizione	Questa classe modella l'eccezione che viene lanciata nel momento in cui un utente prova a registrarsi o prova ad effettuare il LogIn inserendo dei campi non validi
Signature Metodi	+ CredentialsException(message: String)
Pre-condizioni	**N\A**
Post-Condizioni	context CredentialsException:: CredentialException(message) ● **post:** result.message=message

3.2.6 DoraIA

Nome classe	FitnessHelperSpec
Sottosistema	DoraIA
Variabili di istanza	- performance: Double - camera: Double - display: Double - battery: Double - now Date
Descrizione	Questa classe fornisce metodi per calcolare i valori di fit delle Spec utili al nostro tool di Intelligenza Artificiale
Signature Metodi	- getMonthsDifference(date1: Date, date2: Date): Integer - expand(in: Double): Double - aging(specDate String, now: Date): Double + computeSpecFit(spec: Spec): Double + computeVariance(gene: SpecGene): Double + computeFit(gene: SpecGene): Double + FitnessHelperSpec(performance: Double, camera: Double, display: Double, now: Date,

	battery: Double, minBattery: Integer, maxBattery: Integer): FitnessHelperSpec
Pre-Condizioni	• context FitnessHelperSpec::getMonthsDifference(date1, date2) **pre:** date1 <> null and date2 <> null • context FitnessHelperSpec::expand(in) **pre:** N/A • context FitnessHelperSpec::aging(specDate, now) **pre:** specDate <> null and now <> null • context FitnessHelperSpec::computeSpecFit(spec) **pre:** spec <> null • context FitnessHelperSpec::computeVariance(gene) **pre:** gene <> null • context FitnessHelperSpec::computeFit(gene) **pre:** gene <> null

Post-Condizioni	• context FitnessHelperSpec::getMonthsDifference(date1, date2)
	post: result = date2.month - date1.month
	• context FitnessHelperSpec::expand(in)
	post:
	if in < 2. 67 then
	result=in*1.2
	else
	result=(in/2)^4
	endif
	• context FitnessHelperSpec::aging(specDate, now)
	post: result=(getMonthsDifference(specDate, now)/25)^4
	• context FitnessHelperSpec::computeSpecFit(spec)
	post: result=expand(this.performance-spec.performance) + expand(this.camera - spec.camera) + expand(this.display - spec.display) + expand(this.battery - spec.battery) + aging(spec.date, now)

	● context FitnessHelperSpec::computeVariance(gene) **post:** N/A ● context FitnessHelperSpec::computeFit(gene) **post:** result = gene -> count(g|computeFit(g))/size(gene)
Invariante	N/A

Nome classe	GEHelper
Sottosistema	DoraIA
Variabili di istanza	- mutation: Mutation - crossover: Crossover - selection: Selection - fitnessHelper: FitnessHelper - specs: ArrayList<Spec>
Descrizione	Questa classe fornisce metodi di utility alla creazione di una popolazione per l'algoritmo genetico e per l'avvio dell'algoritmo stesso con i parametri scelti
Signature Metodi	+ getMutation(): Mutation + getCrossover(): Crossover + getSelection(): Selection + getFitnessHelper(): FitnessHelper + GEHelper(specs: ArrayList<Spec>): GEHelper + run(population: Population): SpecGene + generatePopulation(genes ArrayList<Spec>, geneLength: Integer): Population + setMutation(mutation: Mutation): void + setCrossover(crossover: Crossover): void + setSelection(selection : Selection): void + setFitnessHelper(fitnessHelper: FitnessHelper): void - generateNewPopulation(oldPopulation: Population): Population - generateRandomSpec(): Spec
Pre-Condizioni	● context GEHelper::setMutation(mutation) **pre:** mutation <> null ● context GEHelper::setCrossover(crossover)

	pre: crossover <> null ● context GEHelper::setSelection(selection) **pre:** selection <> null ● context GEHelper::setFitnessHelper(fitnessHelper) **pre:** fitnessHelper <> null ● context GEHelper::generateNewPopulation(oldPopulation) **pre:** oldPopulation <> null
Post-Condizioni	● context GEHelper:: getMutation() **post**: result=this.mutation ● context GEHelper::getCrossover() **post**:result=this.crossover ● context GEHelper::getSelection() **post**:result=this.selection ● context GEHelper::getFitnessHelper(): **post**:result=this.fitnessHelper ● context GEHelper::GEHelper(specs) **post**:result.specs=specs ● context GEHelper::run(population) **post**: N/A ● context GEHelper::generatePopulation(genes,geneLength) **post**:result.genes=genes and result.geneLength ● context GEHelper::setMutation(mutation) **post**:this.mutation=mutation ● context GEHelper::setCrossover(crossover) **post**:this.crossover=crossover ● context GEHelper::setSelection(selection) **post**:this.selection=selection ● context GEHelper::setFitnessHelper(fitnessHelper) **post**:this.fittnessHelper=fittnessHelper ● context GEHelper::generateNewPopulation(oldPopulation) **post**:result=Popolation(oldPopulation.genes,oldPopulation.geneLenght) ● context GEHelper::generateRandomSpec() **post:** specs->includes(result)

Nome classe	SpecGene
Sottosistema	DoraIA
Variabili di istanza	- gene: ArrayList - fit: Double
Descrizione	Questa classe rappresenta l'individuo dell'algoritmo genetico
Signature Metodi	+ getGene(): ArrayList + setGene(gene: ArrayList): void + GeneSpec(data ArrayList) + getFit(): Double + setFit(fit: Double): void + toString(): String
Pre-Condizioni	● context GeneSpec::getGene() **pre:** N/A ● context GeneSpec::setGene(gene) **pre:** gene <> null ● context GeneSpec::GeneSpec(data) **pre:** N/A ● context GeneSpec::getFit() **post:** N/A ● context GeneSpec::setFit(fit) **pre:** fit <> null
Post-Condizioni	● context GeneSpec::getGene() **post:** result=this.gene ● context GeneSpec::setGene(gene) **post:** this.gene=gene ● context GeneSpec::GeneSpec(data) gene=data ● context GeneSpec::getFit() **post:** result=this.fit ● context GeneSpec::setFit(fit) **post:** this.fit=fit
Invariante	N/A

Nome classe	Population
Sottosistema	DoraIA
Variabili di istanza	- population: ArrayList<SpecGene>
Descrizione	Questa classe rappresenta una popolazione composta da più individui
Signature Metodi	+ Population(population: ArrayList<SpecGene>): Population + Population(): Population + addGene(g1: SpecGene): void + getPopulation(): ArrayList<SpecGene> + toString(): String + shuffle(): void + getRandomSpecGene(): SpecGene - randomInt(Min: Integer, Max: Integer): Integer
Pre-Condizioni	● context Population::Population(population) **pre:** N/A ● context Population::Population() **pre:** N/A ● context Population::addGene(g1) **pre:** g1 <> null ● context Population::getPopulation() **pre:** N/A ● context Population::toString() **pre:** N/A ● context Population::shuffle() **pre:** N/A ● context Population::getRandomSpecGene() **pre:** N/A ● context Population::randomInt(min, max) **pre:** N/A
Post-Condizioni	● context Population::Population(population) **post:** this.population=population ● context Population::Population() **post:** N/A ● context Population::addGene(g1) **post:** population->includes(g1) ● context Population::getPopulation()

	post: result=population ● context Population::toString() **post:** N/A ● context Population::shuffle() **post:** N/A ● context Population::getRandomSpecGene() **post:** population->includes(result) ● context Population::randomInt(min, max) **post:** result=min or result=max
Invariante	N/A

Nome classe	RandomMutation
Sottosistema	DoraIA
Variabili di istanza	N/A
Descrizione	Questa classe rappresenta un componente dell'algoritmo genetico che si occupa della mutazione tramite mutazione randomica di un gene. Implementa l'interfaccia mutate
Signature Metodi	+ mutate(gene: SpecGene, mutation: Object): SpecGene
Pre-Condizioni	● context RandomMutation::mutate(gene, mutation) **pre:** gene<>null and mutation<>null
Post-Condizioni	● context RandomMutation::mutate(gene, mutation) **post:** result->includes(mutation)
Invariante	N/A

Nome classe	SinglePointCrossover
Sottosistema	DoraIA
Variabili di istanza	N/A
Descrizione	Questa classe rappresenta un componente dell'algoritmo genetico che si occupa della generazione di un nuovo individuo dati due individui genitori. Implementa l'interfaccia Crossover
Signature Metodi	+ cross(g1: SpecGene, g2:SpecGene): SpecGene
Pre-Condizioni	● context SinglePointCrossover::cross(g1, g2) **pre:** g1 <> null and g2 <> null
Post-Condizioni	● context SinglePointCrossover::cross(g1, g2) **post:** N/A
Invariante	N/A

Nome classe	TruncationSelection
Sottosistema	DoraIA
Variabili di istanza	N/A
Descrizione	Questa classe rappresenta un componente dell'algoritmo genetico che si occupa della selezione degli individui data una popolazione. Implementa l'interfaccia Selection
Signature Metodi	+ select(population: Population, maxPopulationSize: Integer): Population + sortByFit(p: ArrayList<SpecGene>): void
Pre-Condizioni	● context TruncationSelection::select(population, maxPopulationSize) **pre:** population <> null and maxPopulationSize>=0 ● context TruncationSelection::sortByFit(p)

	pre: p <> null
Post-Condizioni	• context TruncationSelection::select(population, maxPopulationSize) **post:** result->size=maxPopulationSize and population->includes(result)
Invariante	N/A

Nome classe	Crossover
Sottosistema	DoraIA
Descrizione	Questa interfaccia rappresenta un componente dell'algoritmo genetico che si occupa della generazione di un figlio dati due genitori
Signature Metodi	+ cross(g1: SpecGene, g2:SpecGene): SpecGene

Nome classe	FitnessHelper
Sottosistema	DoraIA
Descrizione	Questa interfaccia rappresenta un componente dell'algoritmo genetico che si occupa della computazione del valore di fit.
Signature Metodi	+ computeFit(gene: SpecGene): Double

Nome classe	Mutation
Sottosistema	DoraIA
Descrizione	Questa interfaccia rappresenta un componente dell'algoritmo genetico che si occupa della mutazione di un gene.
Signature Metodi	+ mutate(gene: SpecGene, mutation: Object): SpecGene

Nome classe	Selection
Sottosistema	DoraIA
Descrizione	Questa interfaccia rappresenta un componente dell’algoritmo genetico che si occupa della selezione di un numero di individui data una popolazione.
Signature Metodi	+ select(population: Population, maxPopulationSize: Integer): Population