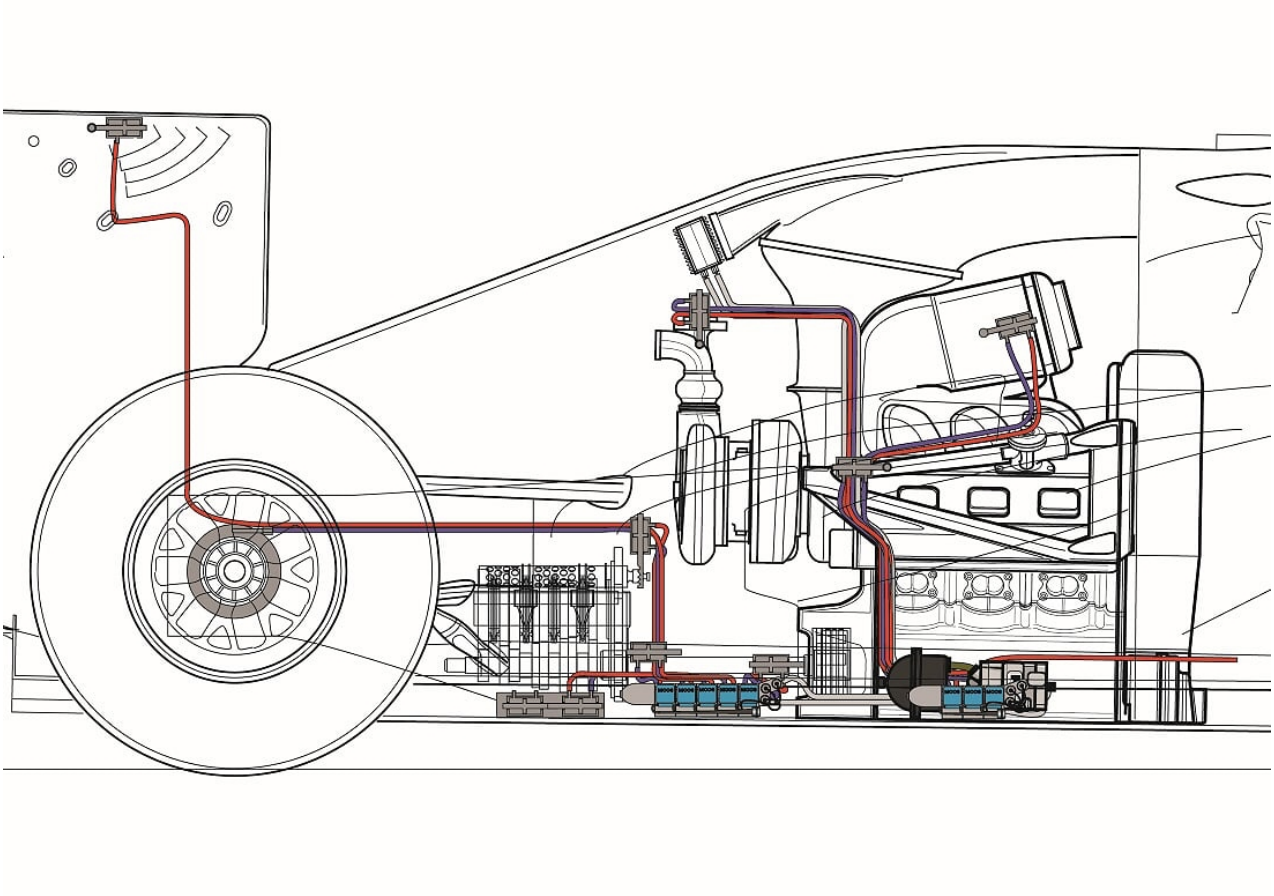


Object Design Document

AM-GP

Versione	Cambiamenti	Autori
0.1	Aggiunta Introduzione	Pietro Negri
0.2	Aggiunta Interfacce	Mafalda Ingenito, Antonio Trovato, Giovanni Renzulli
0.3	Suddivisione in Package	Pietro Negri, Antonio Trovato, Mafalda Ingenito, Giovanni Renzulli
0.9	Correzione interfacce, aggiunta class diagram	Pietro Negri, Antonio Trovato, Giovanni Renzulli
1.0	Aggiunta linee guida	Pietro Negri

Indice generale

1. Introduzione

1.1 Trade-Off

Scalabilità vs Prestazioni

Non è da sottovalutare l'impatto che molti utenti concorrenti potrebbero avere sul sistema. Essendo però limitato il numero di utenti connessi, in quanto si tratta di una piattaforma privata, è alquanto improbabile che esso possa scalare oltre una certa soglia. Di conseguenza, abbiamo preferito cercare di limitare i tempi di risposta, favorendo le prestazioni.

Tempi vs Costi

Data l'esperienza limitata del team di sviluppo, abbiamo deciso di utilizzare strumenti conosciuti da ogni membro, per ridurre il costo derivante dall'apprendimento di nuovi strumenti. Ciò significa che abbiamo rinunciato ad alternative che, al netto di uno sviluppo più rapido, avrebbero avuto un costo superiore in termini di apprendimento, in favore di strumenti già conosciuti e ampiamente utilizzati.

Efficienza vs Portabilità

Da un lato, gli strumenti e il linguaggio utilizzati consentono alla piattaforma di avere un certo grado di portabilità, dall'altro ciò significa fare meno assunzioni sulla piattaforma utilizzata e di conseguenza ciò riduce l'efficienza generale del codice.

Sicurezza vs Tempi

Si è preferito, data la natura privata della piattaforma, inserire più controlli sui form e sull'autenticazione in maniera tale da favorire la sicurezza, al netto di una perdita in termini di tempo.

Usabilità vs Costi

Si è scelto di rendere il sistema il più intuitivo possibile e il più semplice da utilizzare possibile, in maniera tale da rendere più semplice l'utilizzo da parte dell'utente.

1.2 Linee Guida

1.2.1 Package, Model, Bean

Vengono adoperate le seguenti convenzioni per le classi Java:

Le classi sono schematizzate in questo modo:

1. Clausole di Import
2. Dichiarazione della classe
3. Metodi
4. Variabili d'istanza

Il nome della classe inizia con la lettera maiuscola. In caso il nome sia formato da più parole, tutte le parole dopo la prima inizieranno con la lettera maiuscola.

Il nome di un metodo o di una variabile d'istanza fa uso della camelCase notation, con la prima lettera della prima parola minuscola e le parole successive in maiuscolo.

I nomi di classi, attributi e metodi devono rispecchiare in maniera più significativa possibile il concetto che esprimono all'interno del sistema.

L'insieme delle dichiarazioni delle variabili d'istanza è preceduto e seguito da una riga vuota.

L'implementazione di un metodo è preceduta e seguita da una riga vuota.

Ogni classe bean possiede getters e setters, secondo lo standard JavaBean.

Si cerca di evitare il più possibile l'uso dei commenti, per quanto ammessi, per una questione di leggibilità.

1.2.2 Pagine JSP e HTML

Le pagine JSP devono, quando compilate, produrre un documento conforme allo standard HTML5.

Il codice dell'HTML statico viene indentato facendo uso dei caratteri di tabulazione.

Gli stili CSS devono essere collocati in file CSS separati per favorire la manutenibilità e la correzione rapida dello stile visivo delle pagine.1.2.3 Database

Le tabelle del DB sono implementate secondo le seguenti regole:

Il nome di una tabella inizia con una lettera maiuscola, in caso sia composta da più parole allora ogni parola successiva alla prima presenta la prima lettera in maiuscolo.

Il nome dovrà essere un sostantivo singolare.

Gli attributi invece seguono le seguenti convenzioni:

Devono essere costituiti esclusivamente da lettere maiuscole.

In caso l'attributo sia composto da più parole, è consentito separare le parole da un underscore. Esempio: nome_composto.

Il nome deve essere un sostantivo singolare.

1.3 Definizioni, Acronimi, Abbreviazioni

ODD: Object Design Documento

DAO: Data Access Object

CRUD: Create, Retrieve, Update, Delete

JDBC: Java Database Connectivity

1.4 Riferimenti

- Bernd Bruegge & Allen H.Dutoit, Object-Oriented Software Engineering: Using UML, Patterns and Java, (3rd Edition), Prentice Hall 2010.
- AM-GP RAD V1.4
- AM-GP SSD V1.1

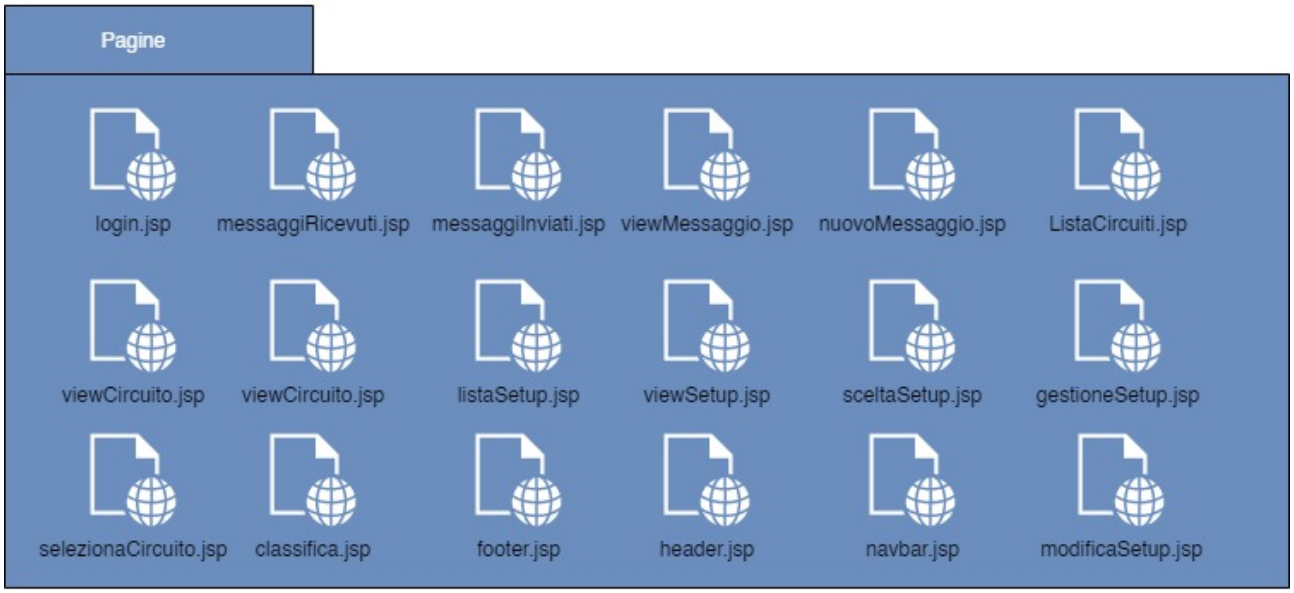
2. Packages

La suddivisione in packages deriva direttamente dai sottosistemi descritti nel documento di System Design.

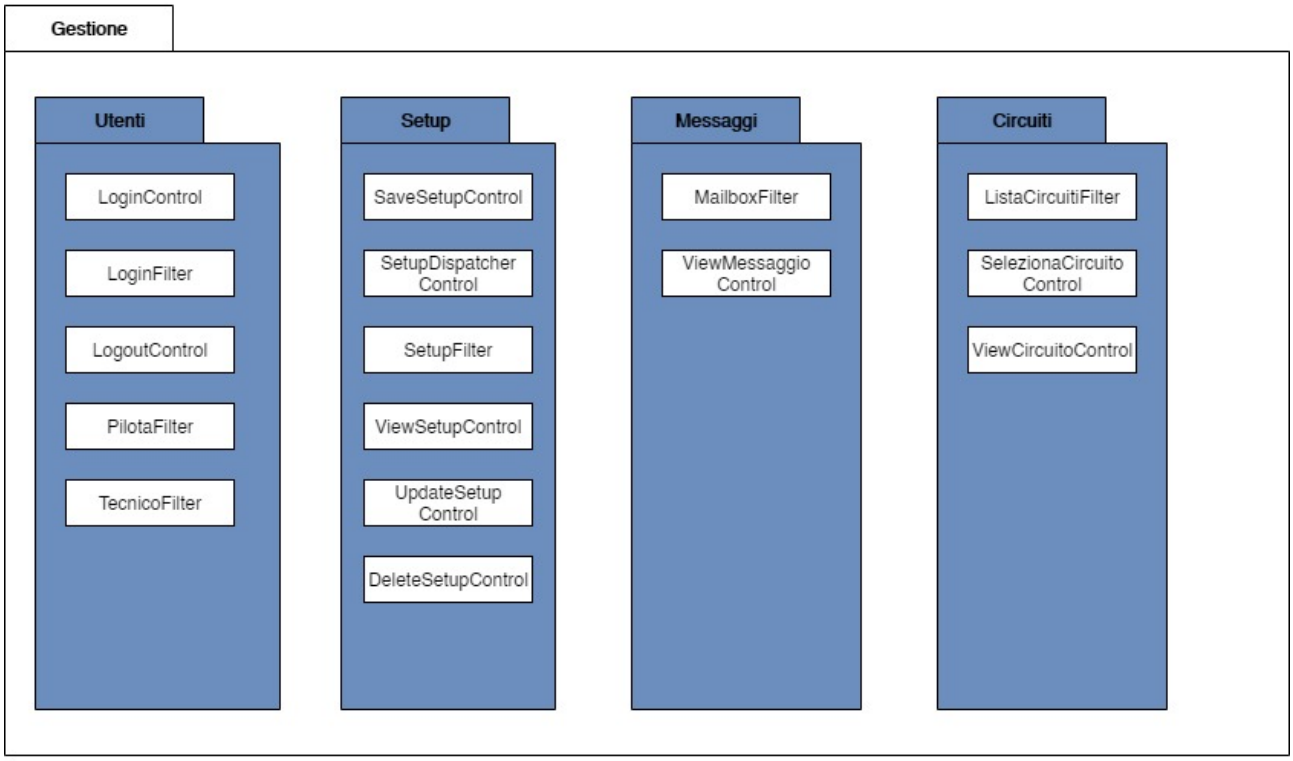
I package di gestione conterranno i vari gestori di controllo, le Servlet, mentre nel model sono presenti i package per i DAO e per gli oggetti Entity.

Va notato che i sottosistemi per il parsing dei file e per la gestione degli strateghi, in quanto requisiti a media priorità, non sono stati ancora implementati all'interno del sistema.

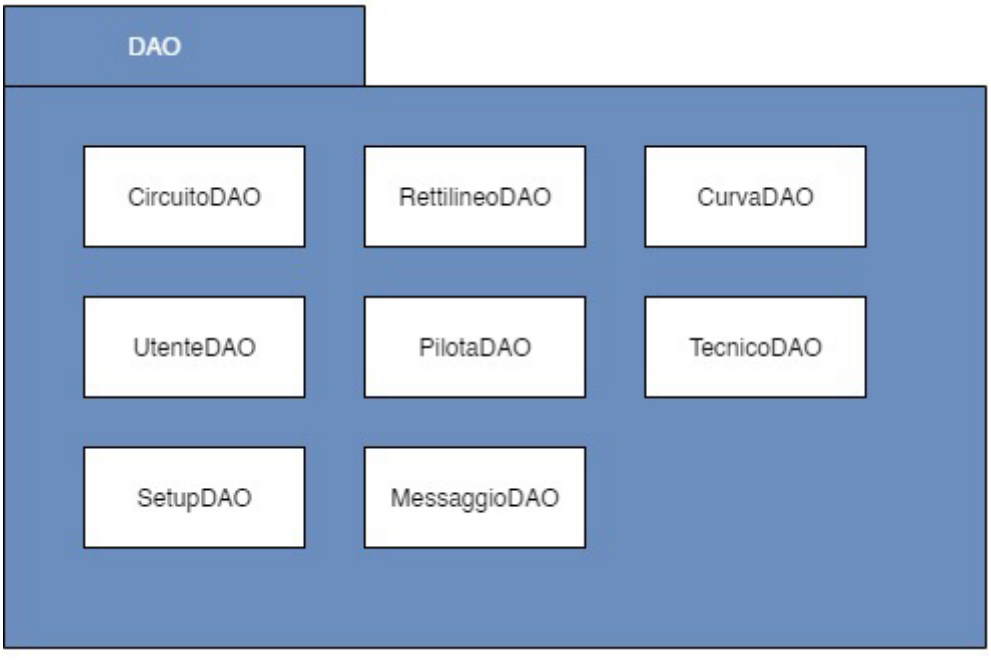
2.1.1 Pagine



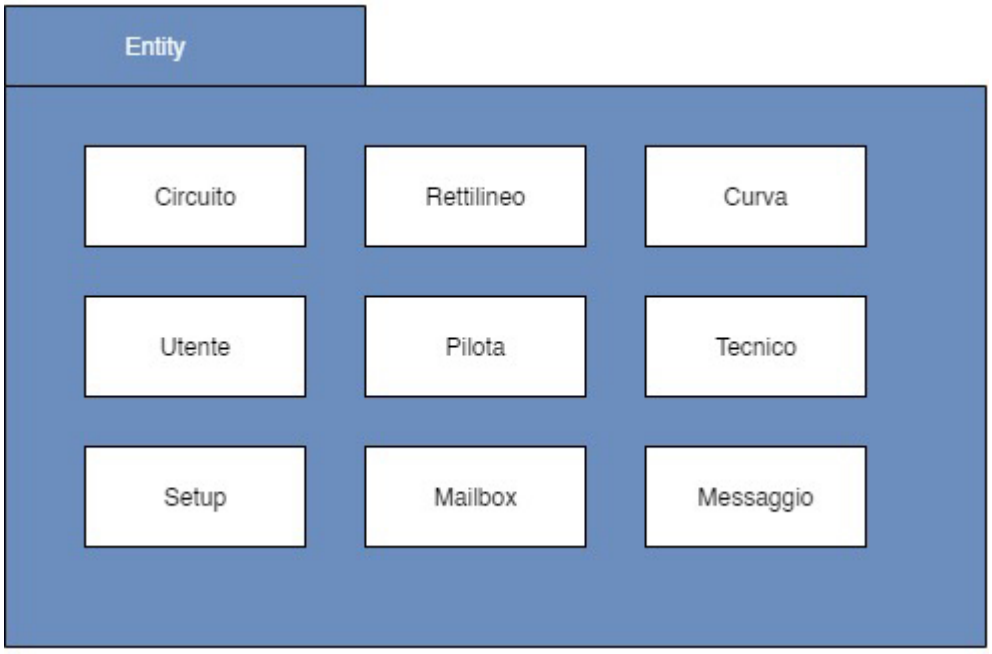
2.1.2 Package Control



2.1.3 Package DAO



2.1.4 Package Entity



3. Interfacce delle Classi

1.1 Utente

Nome Classe	Utente
Descrizione	La classe fornisce tutte le informazioni necessarie per la gestione dell'utente registrato al sistema
Metodi	+getId(): String +setId(String id):void + getPassword():String + setPassword(String password):void + getRuolo():String + setRuolo(String ruolo):void +getNome():String + setNome(String nome):void + getCognome():String + setCognome(String cognome):String + getMailbox():Mailbox + setMailbox(Mailbox mailbox):void

Nome Classe	Utente
Nome Metodo	+ setId(String id):void
Descrizione	Il metodo setta l'id associato all'utente chiamante
Pre-Condizioni	**context**: Utente:: setId(String id):void **pre**: ((id != null) && (id.match("^[A-Za-z0-9]{14}$")
Post-Condizioni	-
Invarianti	-

Nome Classe	Utente
Nome Metodo	+ setPassword(String password):void
Descrizione	Il metodo modifica la password associata all'utente chiamante
Pre-Condizioni	**context**: Utente::setPassword(password) **pre**: ((password != null) && (password.match("(?=.*[0-9])(?=.*[a-z])(?=.*[A-Z])(?=.*[@#$%^&+=])(?=\\S+$).{8,}"))
Post-Condizioni	-
Invarianti	-

Nome Classe	Utente
Nome Metodo	+ setRuolo(String ruolo):void
Descrizione	Il metodo setta il ruolo dell'utente associato
Pre-Condizioni	**context**: Utente:: setRuolo(String ruolo):void **pre**: ((ruolo != null) %% (ruolo != ""))

Post-Condizioni	-
Invarianti	-

Nome Classe	Utente
Nome Metodo	+ setNome(String nome):void
Descrizione	Il metodo setta il nome dell'utente associato
Pre-Condizioni	**context**: Utente:: setNome(String nome):void **pre**: ((nome != null) && (nome != ""))
Post-Condizioni	-
Invarianti	-

Nome Classe	Utente
Nome Metodo	+ setCognome(String cognome):String
Descrizione	Il metodo setta il cognome dell'utente associato
Pre-Condizioni	**context**: Utente:: setCognome(String cognome):String **pre**: ((cognome != null) && (cognome != ""))
Post-Condizioni	-
Invarianti	-

Nome Classe	Utente
Nome Metodo	+ setMailbox(Mailbox mailbox):void
Descrizione	Il metodo setta la mailbox dell'utente associato
Pre-Condizioni	**context**: Utente:: setMailbox(Mailbox mailbox):void **pre**: (mailbox != null)
Post-Condizioni	-
Invarianti	-

1.2 UtenteDAO

Nome Classe	UtenteDAO
Descrizione	La classe gestisce il model dell'entità utente
Metodi	+ doRetriveByKey(String id):Utente + match(String id): boolean

2.1 Messaggio

Nome Classe	Messaggio
Descrizione	La classe fornisce tutte le informazioni relative all'oggetto di tipo messaggio
Metodi	+ getId(): String +setId(int id):void + getOggetto():String + setOggetto(String oggetto):void + getCorpo():String + setCorpo(String corpo):void + getMittente():String +setMittente(Utente mittente):void +getDestinatario():Utente + setDestinatario(Utente destinatario):void

Nome Classe	Messaggio
Nome Metodo	+ setOggetto(String oggetto):void
Descrizione	Il metodo modifica l'oggetto associato al messaggio chiamante
Pre-Condizioni	**context**: Messaggio:: setOggetto(String oggetto):void **pre**: ((oggetto != null) && (oggetto != "") && (oggetto.length<=30))
Post-Condizioni	-
Invarianti	-

Nome Classe	Messaggio
Nome Metodo	+ setCorpo(String corpo):void
Descrizione	Il metodo modifica il corpo associato al messaggio chiamante
Pre-Condizioni	**context**: Messaggio:: setCorpo(String corpo):void **pre**: ((corpo != null) && (corpo != ""))
Post-Condizioni	-
Invarianti	-

Nome Classe	Messaggio
Nome Metodo	+ setMittente(Utente mittente):void
Descrizione	Il metodo modifica il corpo associatao al messaggio chiamante
Pre-Condizioni	**context**: Messaggio:: setMittente(Utente mittente):void **pre**: (u != null)
Post-Condizioni	-
Invarianti	-

Nome Classe	Messaggio
Nome Metodo	+ setDestinatario(Utente destinatario):void
Descrizione	Il metodo modifica il destinatario associatao al messaggio chiamante
Pre-Condizioni	**context**: Messaggio:: setDestinatario(Utente destinatario):void **pre**: (u != null)
Post-Condizioni	-
Invarianti	-

2.2 MessaggioDAO

Nome Classe	MessaggioDAO
Descrizione	La classe gestisce il model dell'entità Messaggio
Metodi	+ doSave(Messaggio messaggio): void + doRetriveByKey(String id):Messaggio + doRetriveAllBySender(String idUtente):Collection<Messaggio> + doRetrieveAllByReceiver(Utente destinatario): ArrayList<Messaggio>

2.3 Mailbox

Nome Classe	Mailbox
Descrizione	La classe permette di modificare e ottenere le informazioni relative alla Recezione del Messaggio
Metodi	+getMessaggiRicevuti(): Collection<Messaggio> +setMessaggiRicevuti(Collection<Messaggio> messaggiRicevuti):void +getMessaggiInviati(): Collection<Messaggio> +setMessaggiInviati(Collection<Messaggio> messaggiInviati):void

Nome Classe	Mailbox
Nome Metodo	+setMessaggiRicevuti(Collection<Messaggio> messaggiRicevuti):void
Descrizione	Il metodo setta i messaggi ricevuti associati alla mailbox
Pre-Condizioni	**context**: Messaggio:: setMessaggiRicevuti(Collection<Messaggio> messaggiRicevuti):void **pre**: (messaggi != null)
Post-Condizioni	-
Invarianti	-

Nome Classe	Mailbox
Nome Metodo	+setMessaggiInviati(Collection<Messaggio> messaggiInviati):void
Descrizione	Il metodo setta i messaggi inviati associati alla mailbox
Pre-Condizioni	**context**: Messaggio::setMessaggiInviati(Collection<Messaggio> messaggiInviati):void **pre**: (messaggi != null)
Post-Condizioni	-
Invarianti	-

3.1 Pilota

Nome Classe	Pilota
Descrizione	La classe fornisce tutte le informazioni necessarie per la gestione del pilota registrato al sistema.
Metodi	+ getPunteggio():int + setPunteggio(int punteggio):void +getNumeroVittorie():int +setNumeroVittorie(int numeroVittorie) : void +getNumeroPole() :int +setNumeroPole(int numeroPole):void +getNumeroPiazzamenti() : int +setNumeroPiazzamenti(int numeroPiazzamenti) : void +getNumeroRitiri(): int +setNumeroRitiri(int numeroRitiri):void

Nome Classe	Pilota
Nome Metodo	+ setPunteggio(int punteggio):void
Descrizione	Il metodo setta il punteggio del pilota
Pre-Condizioni	**context**: Pilota :: setPunteggio(int punteggio):void **pre:** (punteggio>=0)
Post-Condizioni	-
Invarianti	**context** Pilota **inv**:0<=self.punteggio <=575

Nome Classe	Pilota
Nome Metodo	+ setNumeroVittorie(int numeroVittorie):void
Descrizione	Il metodo setta il numero di vittore del pilota
Pre-Condizioni	**context**: Pilota :: setNumeroVittorie(int numeroVittorie):void **pre:** (numeroVittorie >=0)
Post-Condizioni	-
Invarianti	**context** Pilota **inv**:0<=self.numeroVittorie <=23

Nome Classe	Pilota
Nome Metodo	+ setNumeroPole(int numeroPole):void
Descrizione	Il metodo setta il numero di pole effettuate dal pilota
Pre-Condizioni	**context**: Pilota :: setNumeroPole(int numeroPole):void **pre:** (numeroPole >= 0)
Post-Condizioni	-
Invarianti	**context**: Pilota **inv**: 0<= self.numeroPole <= 23

Nome Classe	Pilota
Nome Metodo	+setNumeroPiazzamenti(int numeroPiazzamenti) : void
Descrizione	Il metodo setta il numero di piazzamenti effettuati dal pilota
Pre-Condizioni	**context**: Pilota :: setNumeroPiazzamenti(int numeroPiazzamenti) : void **pre:** (numeroPiazzamenti >=0)
Post-Condizioni	-
Invarianti	**context** Pilota **inv**:0<=self.numeroPiazamenti <=23

Nome Classe	Pilota
Nome Metodo	+ setNumeroRitiri(int numeroRitiri):void
Descrizione	Il metodo setta il numero di ritiri effettuati dal pilota
Pre-Condizioni	**context**: Pilota :: setNumeroRitiri(int numeroRitiri):void **pre:** (numeroRitiri >=0)
Post-Condizioni	-
Invarianti	**context** Pilota **inv**:0<=self.numeroRitiri <=23

3.2 PilotaDAO

Nome Classe	PilotaDAO
Descrizione	La classe gestisce l'interazione tra la classe pilota e il database.
Metodi	+ doSave(Pilota pilota): void + doRetrieveByPilota(Pilota pilota):Pilota

4.1 Tecnico

Nome Classe	Tecnico
Descrizione	La classe fornisce tutte le informazioni necessarie per la gestione del tecnico registrato al sistema.
Metodi	+getSetupList(): Collection<Setup> + setSetupList(Collection<Setup> setupList):void + addSetup(Setup s):void + removeSetup(Setup s):void
Nome Metodo	+ setSetupList(Collection<Setup> setupList):void
Descrizione	Il metodo setta il numero totale di setup per il tecnico.
Pre-Condizioni	**context**: Pilota ::setSetupList(Collection<Setup > setupList):void **pre:** (setupList != null)
Post-Condizioni	-
Invarianti	-

Nome Classe	Tecnico
Nome Metodo	+ addSetup(Setup s):void
Descrizione	Il metodo aggiunge un nuovo setup per il tecnico
Pre-Condizioni	**context**: Tecnico :: addSetup(Setup s):void **pre:** (s != null)
Post-Condizioni	-
Invarianti	-

Nome Classe	Tecnico
Nome Metodo	+ removeSetup(Setup s):void
Descrizione	Il metodo rimuove un setup per il tecnico
Pre-Condizioni	**context**: Tecnico :: removeSetup(Setup s):void **pre:** (s != null)
Post-Condizioni	-
Invarianti	-

5.1 Setup

Nome Classe	Setup
Descrizione	La classe ingloba il concetto di Setup e fornisce una serie di operazioni necessarie per la gestione dei setup
Metodi	+ getId(): String +getCaricoAreodinamicoAnteriore(): Int +getCaricoAreodinamicoPosteriore(): Int +getCampanaturaAnteriore(): Int +getCampanaturaPosteriore(): Int +getConvergenzaAnteriore(): Int +getConvergenzaPosteriore(): Int +getPressioneFreni(): Int +getBarraAntirollioAnteriore(): Int +getBarraAntirollioPosteriore(): Int +getCircuito(): Circuito +getTecnico(): Tecnico +setCaricoAreodinamicoAnteriore(Int cAA): void +setCaricoAreodinamicoPosteriore(Int cAP): void +setCampanaturaAnteriore(Int cA): void +setCampanaturaPosteriore(Int cP): void +setConvergenzaAnteriore(Int cA): void +setConvergenzaPosteriore(Int cP):

	void +setPressioneFreni(Int pF): void +setBarraAntirollioAnteriore(Int bAA): void +setBarraAntirollioPosteriore(Int bAP): void +setCircuito(Circuito circuito): void +setTecnico(Tecnico tecnico): void

Nome Classe	Setup
Nome Metodo	+ setCaricoAreodinamicoAnteriore(Int cAA):void
Descrizione	Il metodo sostituisce il carico areodinamico anteriore del setup chiamante
Pre-Condizioni	**context**: Setup::setCaricoAreodinamicoAnteriore(Int cAA): void **pre**: (1<=cAA<=10)
Post-Condizioni	-
Invarianti	**context** Setup **inv**: 1 <= self.caricoAreodinamicoAnteriore <= 10

Nome Classe	Setup
Nome Metodo	+ setCaricoAreodinamicoPosteriore(Int cAP):void
Descrizione	Il metodo sostituisce il carico areodinamico posteriore del setup chiamante
Pre-Condizioni	**context**: Setup::setCaricoAreodinamicoPosteriore(Int cAP): void **pre**: (1<=cAP<=10)
Post-Condizioni	-
Invarianti	**context** Setup **inv**: 1 <= self.caricoAreodinamicoPosteriore <= 10

Nome Classe	Setup
Nome Metodo	+ setCampanaturaAnteriore(Int cA):void
Descrizione	Il metodo sostituisce la campanatura anteriore del setup chiamante
Pre-Condizioni	**context**: Setup::setCaricoCampanaturaAnteriore(Int cA): void **pre**: -5<=cA<=5
Post-Condizioni	-
Invarianti	**context** Setup **inv**: -5 <= self.campanaturaAnteriore <= 5

Nome Classe	Setup
Nome Metodo	+ setCampanaturaPosteriore(Int cP):void
Descrizione	Il metodo sostituisce la campanatura posteriore del setup chiamante
Pre-Condizioni	**context**: Setup::setCampanaturaPosteriore(Int cP): void **pre**: -5<=cP<=5
Post-Condizioni	-
Invarianti	**context** Setup **inv**: -5 <= self.campanaturaPosteriore <= 5

Nome Classe	Setup
Nome Metodo	+ setConvergenzaAnteriore(Int cA):void
Descrizione	Il metodo sostituisce la convergenza anteriore del setup chiamante
Pre-Condizioni	**context**: Setup::setConvergenzaAnteriore(Int cA): void **pre**: -1<=cA<=1
Post-Condizioni	-
Invarianti	**context** Setup **inv**: -1 <= self.convergenzaAnteriore <= 1

Nome Classe	Setup
Nome Metodo	+ setConvergenzaPosteriore(Int cP):void
Descrizione	Il metodo sostituisce la convergenza posteriore del setup chiamante
Pre-Condizioni	**context**: Setup::setConvergenzaPosteriore(Int cP): void **pre**: -1<=cP<=1
Post-Condizioni	-
Invarianti	**context** Setup **inv**: -1 <= self.convergenzaPosteriore <= 1

Nome Classe	Setup
Nome Metodo	+ setPressioneFreni(Int pF):void
Descrizione	Il metodo sostituisce la pressione dei freni del setup chiamante
Pre-Condizioni	**context**: Setup::setPressioneFreni(Int pF): void **pre**: 0<=pF<=100
Post-Condizioni	-
Invarianti	**context** Setup **inv**: 0 <= self.pressioneFreni <= 100

Nome Classe	Setup
Nome Metodo	+ setBarraAntirollioAnteriore(Int bAA):void
Descrizione	Il metodo sostituisce la barra antirollio anteriore del setup chiamante
Pre-Condizioni	**context**: Setup::setBarraAntirollioAnteriore(Int bAA): void **pre**: 1<=bAA<=10
Post-Condizioni	-
Invarianti	**context** Setup **inv**: 1 <= self.barraAntirollioAnteriore <= 10

Nome Classe	Setup
Nome Metodo	+ setBarraAntirollioPosteriore(Int bAP):void
Descrizione	Il metodo sostituisce la barra antirollio posteriore del setup chiamante
Pre-Condizioni	**context**: Setup::setBarraAntirollioPosteriore(Int bAP): void **pre**: 1<=bAP<=10
Post-Condizioni	-
Invarianti	**context** Setup **inv**: 1 <= self.campanaturaAnteriore <= 10

Nome Classe	Setup
Nome Metodo	+ setTecnico(Tecnico tecnico):void
Descrizione	Il metodo sostituisce il tecnico del setup chiamante
Pre-Condizioni	**context**: Setup::setTecnico(Tecnico tecnico): void **pre**: (tecnico != null)
Post-Condizioni	-
Invarianti	-

Nome Classe	Setup
Nome Metodo	+ setCircuito(Circuito circuito):void
Descrizione	Il metodo sostituisce il circuito del setup chiamante
Pre-Condizioni	**context**: Setup::setCircuito(Circuito circuito): void **pre**: (circuito != null)
Post-Condizioni	-
Invarianti	-

5.2 SetupDAO

Nome Classe	SetupDAO
Descrizione	La classe gestisce l'interazione tra la classe Setup ed il Database
Metodi	+doSave(Setup setup): void +doRetrieveByTecnico(Tecnico t): Collection<Setup>

6.1 Rettilineo

Nome Classe	Rettilieo
Descrizione	La classe ingloba il concetto di Rettilineo e fornisce una serie di operazioni necessarie per la gestione dei rettilinei
Metodi	+ getId(): Int +getNome(): String +getLunghezza(): Int +setId(Int id): void +setNome(String nome): void +setLunghezza(Int lunghezza): void

Nome Classe	Rettilineo
Nome Metodo	+ setNome(String nome):void
Descrizione	Il metodo sostituisce il nome del rettilineo chiamante
Pre-Condizioni	**context**: Rettilineo::setNome(String nome): void **pre**: (nome != null) and (nome!="")
Post-Condizioni	-
Invarianti	-

Nome Classe	Rettilineo
Nome Metodo	+ setLunghezza(Int lunghezza):void
Descrizione	Il metodo sostituisce la lunghezza del circuito chiamante
Pre-Condizioni	**context**: Rettilineo::setLunghezza(Int lunghezza): void **pre**: (lunghezza > 0)
Post-Condizioni	-
Invarianti	-

6.2 RettilineoDAO

Nome Classe	RettilineoDAO
Descrizione	La classe gestisce l'interazione tra la classe Rettilineo ed il Database
Metodi	+doRetrieveByCircuito(Circuito circuito): Collection<Rettilineo>

7.1 Curva

Nome Classe	Curva
Descrizione	La classe ingloba il concetto di Curva e fornisce una serie di operazioni necessarie per la gestione delle curve
Metodi	+ getId(): Int +getNome(): String +getAngolo(): Int +setId(Int id): void +setNome(String nome): void +setAngolo(Int angolo): void

Nome Classe	Curva
Nome Metodo	+ setNome(String nome):void
Descrizione	Il metodo sostituisce il nome del rettilineo chiamante
Pre-Condizioni	**context**: Curva::setNome(String nome): void **pre**: (nome != null) and (nome!="")
Post-Condizioni	-
Invarianti	-

Nome Classe	Curva
Nome Metodo	+ setAngolo(Int angolo):void
Descrizione	Il metodo sostituisce l'angolo del circuito chiamante
Pre-Condizioni	**context**: Curva::setAngolo(Int angolo): void **pre**: (angolo > 0)
Post-Condizioni	-

7.2 CurvaDAO

Nome Classe	CurvaDAO
Descrizione	La classe gestisce l'interazione tra la classe Curva ed il Database
Metodi	+doSave(Curva curva, Circuito circuito): void +doRetrieveByCircuito(Circuito circuito): Curva +doRetrieveAllByCircuito(Circuito circuito): Collection<Curva>

8.1 Circuito

Nome Classe	Circuito
Descrizione	La classe ingloba il concetto di Circuito e fornisce una serie di operazioni necessarie per la gestione dei circuiti
Metodi	+ getSede(): String +getLunghezza(): Int +getMeteo(): String +getNumeroCurve(): Int +getTPM(): Long +getRPM(): Long +getUmidita(): Int +getNumeroGiri(): Int +getRettilinei(): Rettilineo +getCurve(): Curva +setSede(String sede): void +setLunghezza(Int lunghezza): void +setMeteo(String meteo): void +setUmidita(Int umidita): void +setNumeroCurve(Int numeroCurve): void +setTPM(Long tpm): void +setRPM(Long rpm): void +setNumeroGiri(Int numeroGiri): void +setRettilinei(Collection<Rettilineo> rettilinei): void +setCurve(Collection<Curva> curve): void

Nome Classe	Circuito
Nome Metodo	+ setSede(String sede):void
Descrizione	Il metodo sostituisce la sede del circuito chiamante
Pre-Condizioni	**context**: Circuito::setSede(String sede): void **pre**: (sede != null) and (sede!="")
Post-Condizioni	-
Invarianti	-

Nome Classe	Circuito
Nome Metodo	+ setLunghezza(Int lunghezza):void
Descrizione	Il metodo sostituisce la lunghezza del circuito chiamante
Pre-Condizioni	**context**: Circuito::setLunghezza(Int lunghezza): void **pre**: (lunghezza > 0)
Post-Condizioni	-
Invarianti	-

Nome Classe	Circuito
Nome Metodo	+ setMeteo(String meteo):void
Descrizione	Il metodo sostituisce il meteo del circuito chiamante
Pre-Condizioni	**context**: Circuito::setMeteo(String meteo): void **pre**: (meteo != null)
Post-Condizioni	-

Invarianti	-

Nome Classe	Circuito
Nome Metodo	+ setTPM(Long tpm):void
Descrizione	Il metodo sostituisce il tpm del circuito chiamante
Pre-Condizioni	**context**: Circuito::setTPM(Long tpm): void **pre**: (tpm >= 0)
Post-Condizioni	-
Invarianti	-

Nome Classe	Circuito
Nome Metodo	+ setRPM(Long rpm):void
Descrizione	Il metodo sostituisce il rpm del circuito chiamante
Pre-Condizioni	**context**: Circuito::setRPM(Long rpm): void **pre**: (rpm > 0)
Post-Condizioni	-
Invarianti	-

Nome Classe	Circuito
Nome Metodo	+ setUmidita(Int umidita):void
Descrizione	Il metodo sostituisce l'umidità del circuito chiamante
Pre-Condizioni	**context**: Circuito::setUmidita(Int umidita): void **pre**: (umidita != null)
Post-Condizioni	-
Invarianti	-

Nome Classe	Circuito
Nome Metodo	+ setNumeroGiri(Int numeroGiri):void
Descrizione	Il metodo sostituisce il numero di giri del circuito chiamante
Pre-Condizioni	**context**: Circuito::setNumeroGiri(Int numeroGiri): void **pre**: (numeroGiri > 0)
Post-Condizioni	-
Invarianti	-

Nome Classe	Circuito
Nome Metodo	+ setRettilinei(Collection<Rettilineo> rettilinei): void
Descrizione	Il metodo sostituisce i rettilinei del circuito chiamante
Pre-Condizioni	**context**: setRettilinei(Collection<Rettilineo> rettilinei): void **pre**: (rettilinei != null) and (rettilinei.isEmpty() != 0)
Post-Condizioni	-
Invarianti	-

Nome Classe	Circuito
Nome Metodo	+ setCurve(Collection<Curva> curve): void
Descrizione	Il metodo sostituisce le curve del circuito chiamante
Pre-Condizioni	**context**: setCurve(Collection<Curva> curve): void **pre**: (curve != null) and (curve.isEmpty() != 0)
Post-Condizioni	-
Invarianti	-

8.2 CircuitoDAO

Nome Classe	CircuitoDAO
Descrizione	La classe gestisce l'interazione tra la classe Circuito ed il Database
Metodi	+doRetrieveByKey(Int id): Circuito +doSave(Circuito circuito): void +doRetrieveAll():Collection<Circuito>

4. Class Diagram