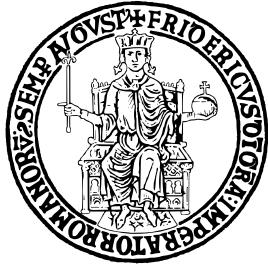


UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II



SCUOLA POLITECNICA E DELLE SCIENZE DI BASE  
DIPARTIMENTO DI INGEGNERIA ELETTRICA E TECNOLOGIE  
DELL'INFORMAZIONE

CORSO DI LAUREA TRIENNALE IN INFORMATICA



**Autori**

Anthony Arenella N86004512

Andrea Campanile Sidero N86004091

Anno Accademico 2024–2025



# Indice

Analisi . . . . .	1
<b>1 Scelte progettuali e architetturali</b>	<b>3</b>
1.1 Scelte progettuali . . . . .	3
1.2 Progettazione concettuale . . . . .	4
1.2.1 Funzionalità dell'applicativo . . . . .	4
1.3 Progettazione del software : Class Diagramm . . . . .	5
1.3.1 Analisi delle entità . . . . .	5
1.3.2 Diagramma delle classi del dominio del problema . . . . .	6
1.4 Diagramma di dettaglio della classi nel dominio della soluzione . . . . .	7
1.4.1 Package GUI . . . . .	8
1.4.2 Package Controller e Factory . . . . .	8
1.4.3 Package DAO e DAOImpl . . . . .	8
1.4.4 Package Stats e ReportPDF . . . . .	9
<b>2 Interfaccia Utente</b>	<b>11</b>
2.1 Progettazione dell'Interfaccia Utente . . . . .	11
2.2 Tecnologie utilizzate . . . . .	11
2.2.1 Styling . . . . .	12
<b>3 Versioning e Github</b>	<b>13</b>
3.1 Strumenti di Versioning . . . . .	13



# **Introduzione**

## **Analisi**

Si sviluppi un applicativo Java con interfaccia grafica (Swing o JavaFX) per la gestione delle coltivazioni e delle attività nella piattaforma UninaBioGarden. Il sistema dovrà essere collegato a un database relazionale pre-popolato contenente informazioni su utenti (proprietari di lotti e coltivatori), lotti e culture. Il sistema deve permettere l'autenticazione dei proprietari di lotti tramite credenziali (username e password). Una volta autenticati, i proprietari potranno creare progetti stagionali, selezionando uno dei propri lotti, una o più colture da associare e definendo le attività da svolgere. Per ogni attività, sarà necessario specificare i coltivatori coinvolti e, nel caso di attività di raccolta, anche la quantità prevista. I proprietari avranno inoltre la possibilità di visualizzare i progetti stagionali esistenti, applicando filtri per lotto. Dopo aver selezionato un progetto, sarà possibile aggiornare lo stato delle attività collegate (ad esempio: pianificata, in corso, completata). Infine, il sistema deve fornire un report che, per ogni lotto, riassuma per ciascuna coltura: il numero totale di raccolte effettuate, la quantità media, minima e massima raccolta. Il report deve fornire una rappresentazione grafica, chiara e intuitiva dei dati, realizzata utilizzando una libreria come JFreeChart.



# - 1 -

## Scelte progettuali e architetturali

CONTENTS: **1.1 Scelte progettuali.** **1.2 Progettazione concettuale.** 1.2.1 Funzionalità dell'applicativo. **1.3 Progettazione del software : Class Diagramm.** 1.3.1 Analisi delle entità – 1.3.2 Diagramma delle classi del dominio del problema. **1.4 Diagramma di dettaglio della classi nel dominio della soluzione.** 1.4.1 Package GUI – 1.4.2 Package Controller e Factory – 1.4.3 Package DAO e DAOImpl – 1.4.4 Package Stats e ReportPDF.

### 1.1 Scelte progettuali

L'applicativo è stato sviluppato attenendosi al design pattern **BCE** (*Boundary-Controller-Entity*), con l'obiettivo di garantire una chiara separazione delle responsabilità tra presentazione, logica di controllo e gestione dei dati. L'interfaccia grafica è stata realizzata tramite **JavaFX**, che consente una definizione modulare e moderna delle viste. I componenti **Boundary** sono responsabili dell'interazione con l'utente e dell'invocazione dei servizi di controllo, attraverso interfacce definite in FXML e gestite tramite i controller associati. I **Controller** implementano la logica applicativa, coordinano il flusso delle operazioni e mediane tra la vista ed il modello. Gli **Entity Model** rappresentano le strutture dati persistenti, mappando le entità del dominio corrispondenti alle tabelle del database relazionale.

Per l'accesso ai dati è stato adottato anche il design pattern **DAO**, al fine di incapsulare la logica di persistenza e separarla dalla logica di business. Ogni entità è gestita da una corrispondente classe *DAO*, che fornisce metodi **CRUD** (*Create, Read, Update, Delete*) e query personalizzate. Questa combinazione di pattern ha favorito una maggiore manutenibilità del codice, una migliore organizzazione dei moduli ed una più agevole integrazione tra interfaccia utente ed il livello di persistenza.

L'applicazione si interfaccia ad un database relazionale pre-popolato, ospitato sulla piattaforma **NeonDB**.

---

## 1.2 Progettazione concettuale

Durante la fase di progettazione dell'applicativo è stato adottato il linguaggio di **UML** (*Unified Modeling Language*) al fine di rappresentare in modo formale e strutturale l'architettura del sistema.

### 1.2.1 Funzionalità dell'applicativo

All'interno dell'applicativo sono disponibili le seguenti funzionalità principali:

1. **Login:** L'utente proprietario dispone di un *username* e di una *password* per accedere all'applicativo e usufruire dei propri vantaggi, come la creazione e la visualizzazione dei progetti. Gli utenti non proprietari non possono effettuare l'accesso.
2. **Visualizzazione dei progetti:** Dopo il login, il proprietario ha due possibilità, tra cui una comprende:
  - (a) Visualizzare i propri progetti già creati.
  - (b) Effettuare una ricerca tra i progetti tramite filtri.

La visualizzazione dei progetti carica tutti i dettagli relativi ad essi. L'utente può oltre vedere i dettagli, anche :

- Modificare lo stato delle attività nella sezione *Dettagli attività*.
- Scaricare un report riguardante le raccolte effettuate in ogni coltura di ogni lotto del progetto.

2.1. **Ricerca tramite filtri:** Sono disponibili due tipi di filtro:

- *Stagione del progetto:* permette di filtrare i progetti in base alla stagione.
- *Superficie del lotto:* consente di specificare un valore in metri quadrati; verranno mostrati solo i progetti con superficie maggiore del valore inserito.

2.2. **Modifica dello stato delle attività:** Accanto allo stato di ciascuna attività è presente un pulsante che apre un menu per modificarlo in modo semplice e immediato.

3. **Creazione del progetto:** L'altra scelta disponibile al proprietario è la creazione di un nuovo progetto. Questo processo è suddiviso in più fasi sbloccabili progressivamente:

- 3.1. **Dettagli del progetto:** inserimento delle informazioni principali del progetto.
- 3.2. **Selezione del lotto:** l'applicativo carica tutti i lotti disponibili del proprietario, che può decidere quali includere nel progetto.

- 3.3. **Selezione delle colture:** vengono elencate tutte le colture disponibili. Accanto ad ognuna è presente un menu che permette di assegnarla a un lotto. Dopo aver effettuato le assegnazioni, il proprietario conferma tramite il pulsante in basso a destra.
- 3.4. **Creazione delle attività:** tramite un pulsante è possibile creare attività legate a specifiche colture. Una schermata dedicata consente di:
  - Selezionare il coltivatore coinvolto.
  - Specificare la data di inizio e di fine.
  - Indicare il tipo di attività da svolgere (*Semina, Raccolta, Irrigazione*).
  - Inserire le quantità raccolte e utilizzate, seguendo queste regole:
    - **Semina:** quantità raccolte = 0.
    - **Raccolta:** quantità utilizzate = 0.
    - **Irrigazione:** entrambe le quantità = 0.
  - Scegliere lo stato dell'attività.È possibile ripetere l'operazione per creare più attività.
- 3.5. **Conferma della creazione:** dopo aver inserito tutte le attività desiderate, il proprietario finalizza la creazione del progetto.
4. **Logout:** Dalla schermata iniziale il proprietario può effettuare un semplice logout dall'applicativo.

## 1.3 Progettazione del software : Class Diagramm

### 1.3.1 Analisi delle entità

Le entità che possono essere individuate nel problema sono :

1. **Utente:** utente che può essere proprietario oppure coltivatore. Il proprietario ha il vantaggio di entrare nel programma con la propria username e password. Per tutti i tipi di utenti si salveranno: nome, cognome, e-mail, password, username e il proprio ruolo.
2. **Progetto:** creato dall'utente proprietario. Si salveranno: chi l'ha creato, la stagione, il titolo, la data di inizio e di fine e i lotti ospitati.
3. **Lotto:** pezzo di terreno all'interno dei progetti. Si salveranno: nome, progetto di appartenenza, indirizzo, superficie occupata, proprietario e colture ospitate.
4. **Colture:** luogo delle attività dei coltivatori. Si salveranno: indirizzo, lotto di appartenenza, tempo di maturazione, stagione, nome e data di inizio coltura.
5. **Attività:** operazioni svolte dai coltivatori. Si salveranno: tipo, stato, quantità raccolta e usata, data di inizio e di fine.

### 1.3.2 Diagramma delle classi del dominio del problema

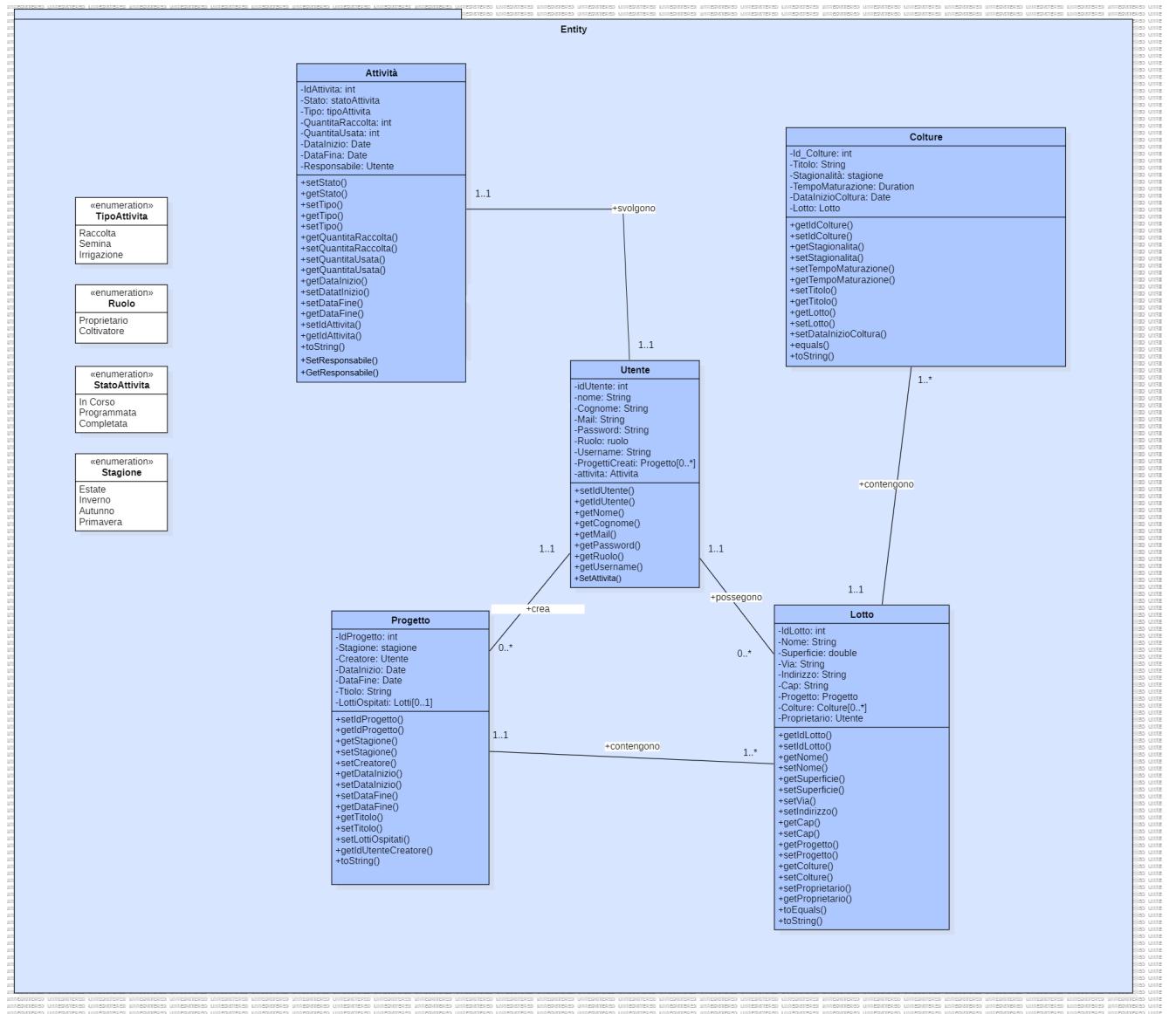


Figura 1.1: diagramma delle entità.

## 1.4 Diagramma di dettaglio della classi nel dominio della soluzione

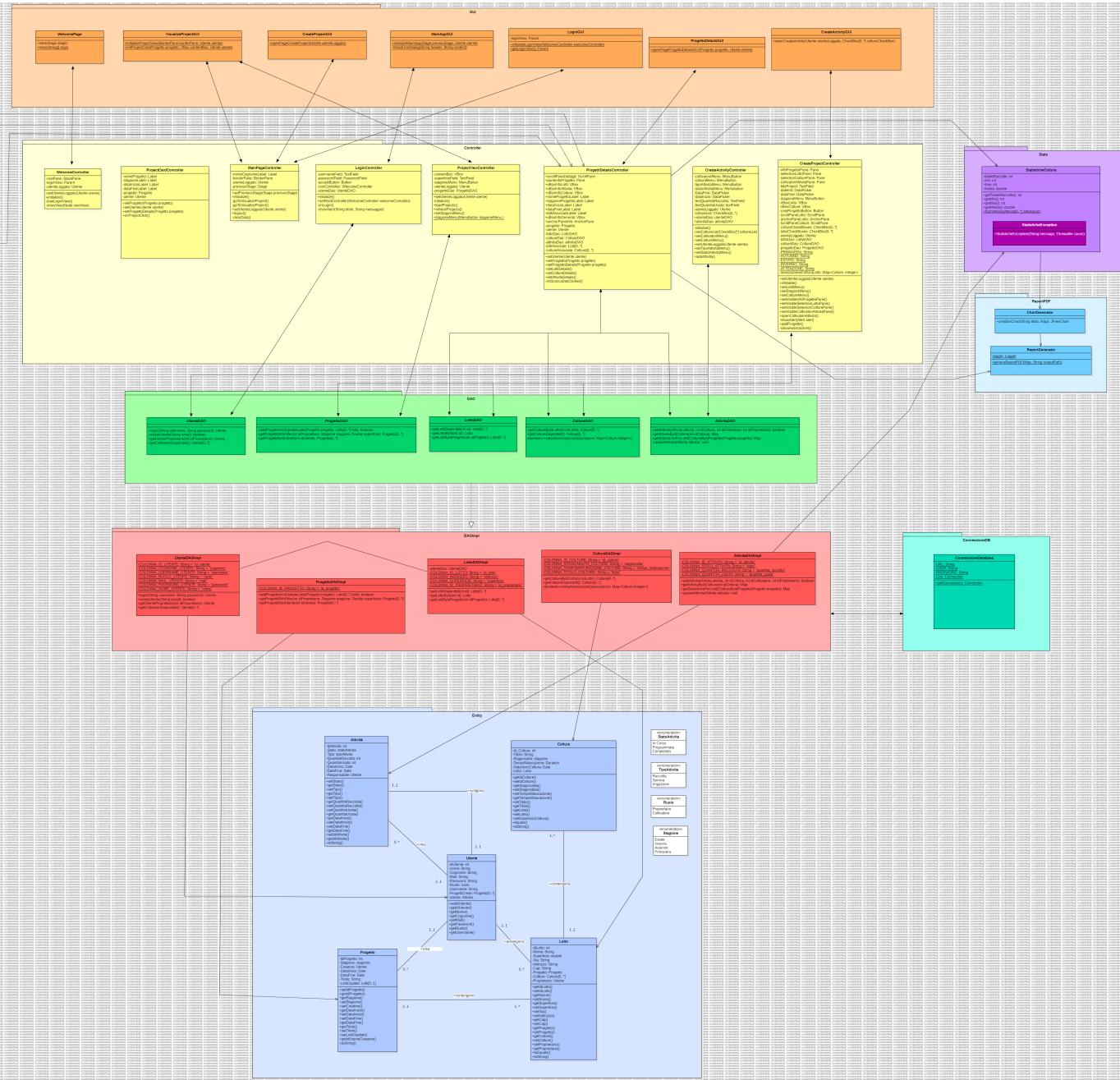


Figura 1.2: Diagramma completo del dominio della soluzione del problema.

Di seguito verranno appositamente mostrati tutti i vari package nello specifico

### 1.4.1 Package GUI

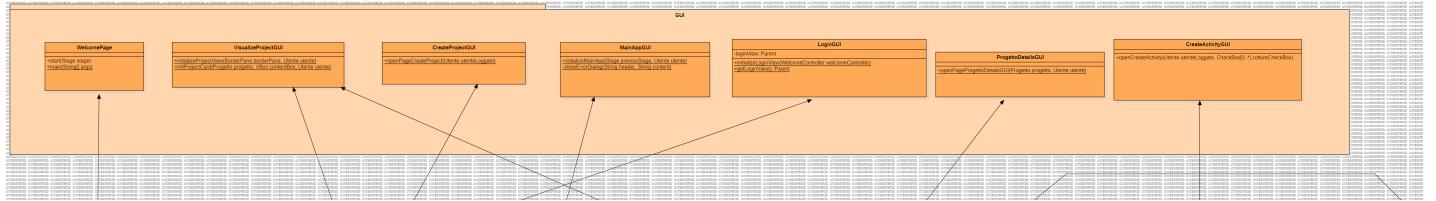


Figura 1.3: diagramma del package GUI

### 1.4.2 Package Controller e Factory

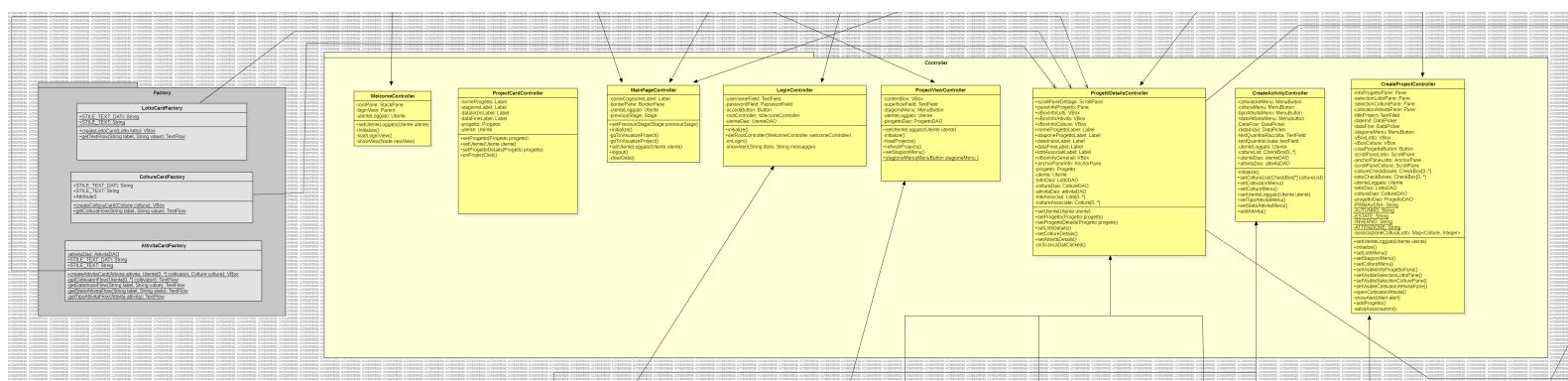


Figura 1.4: diagramma dei package Controller e Factory

### 1.4.3 Package DAO e DAOImpl

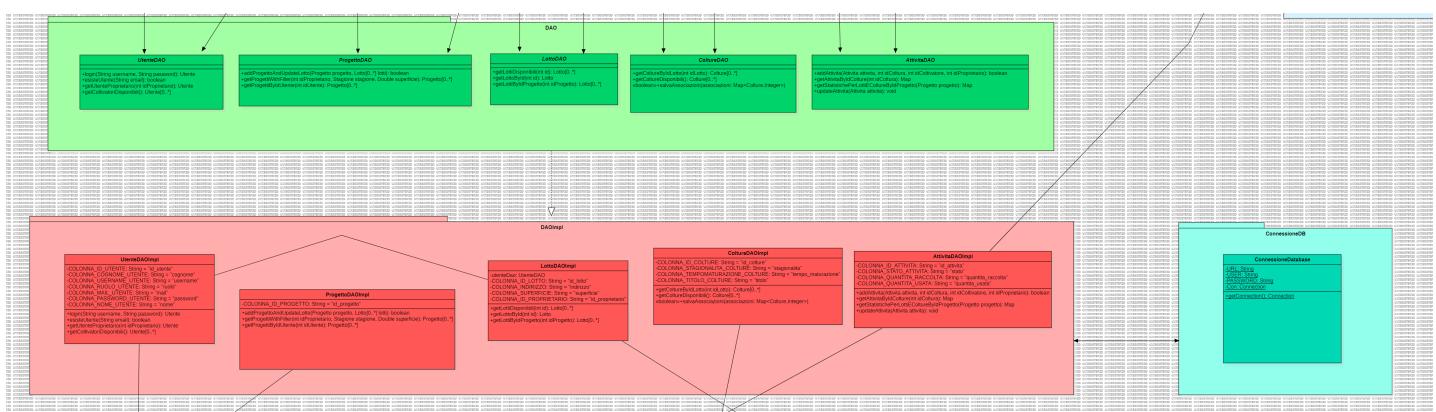


Figura 1.5: diagramma dei package DAO e DAOImpl

#### 1.4.4 Package Stats e ReportPDF

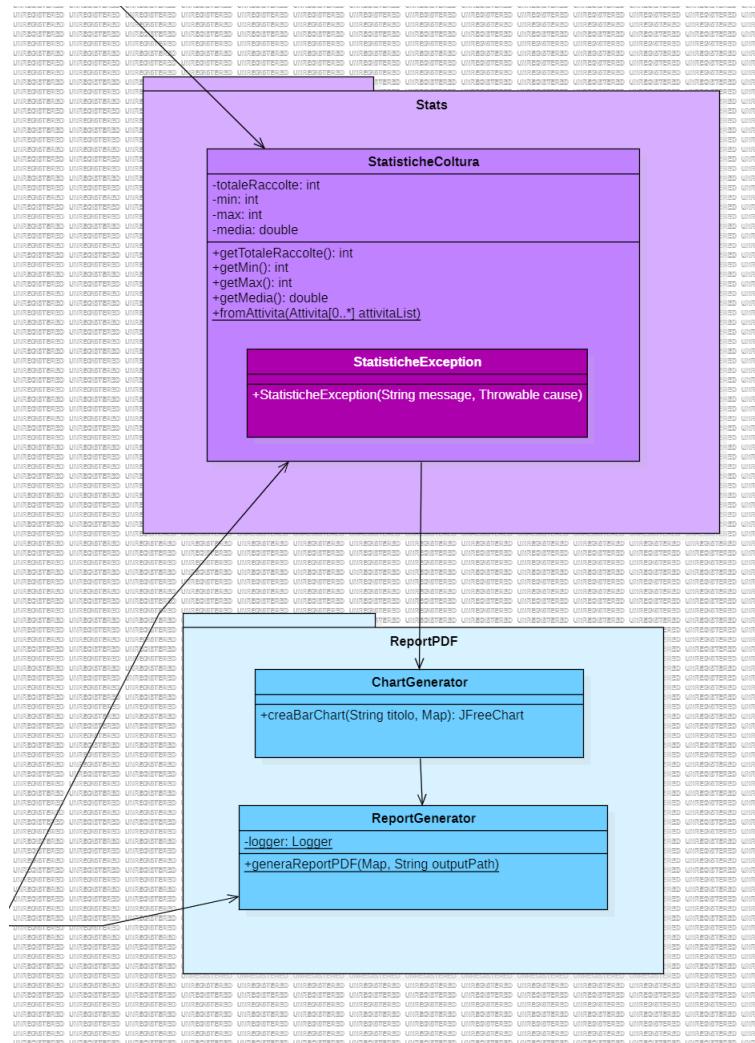


Figura 1.6: diagramma dei package Stats e ReportPDF



# **-2-**

## **Interfaccia Utente**

CONTENTS: **2.1 Progettazione dell'Interfaccia Utente.** **2.2 Tecnologie utilizzate.** 2.2.1 Styling.

### **2.1 Progettazione dell'Interfaccia Utente**

L'interfaccia grafica dell'applicativo è stata progettata e sviluppata utilizzando **JavaFX**, un framework moderno per la creazione di GUI in Java. JavaFX offre un insieme ricco di componenti grafici e funzionalità avanzate che hanno permesso di costruire un'interfaccia utente interattiva, responsiva e coerente con i requisiti funzionali del sistema. La progettazione è stata orientata alla chiarezza e all'usabilità, con particolare attenzione alla disposizione dei controlli, alla gestione degli eventi e all'organizzazione delle schermate, in modo da facilitare l'interazione da parte dei proprietari di lotti e degli altri utenti. Le schermate principali, come login, registrazione, gestione dei progetti stagionali e visualizzazione dei report, sono state realizzate con l'ausilio di FXML, sfruttando il pattern MVC per mantenere separata la logica applicativa dalla presentazione grafica.

### **2.2 Tecnologie utilizzate**

Per semplificare la progettazione delle schermate, è stato impiegato **Scene Builder**, un tool visuale che permette di costruire e modificare interfacce grafiche in modalità drag-and-drop, generando automaticamente i file FXML utilizzati dall'applicazione. *Scene Builder* ha facilitato il lavoro di design, permettendo di definire la struttura della UI in modo intuitivo e rapido, mantenendo allo stesso tempo una chiara separazione tra logica applicativa e presentazione.

---

### 2.2.1 Styling

Per personalizzare l'aspetto grafico dell'interfaccia, è stato utilizzato il linguaggio **CSS** (**Cascading Style Sheets**), che JavaFX supporta nativamente per definire stili, colori, font, margini e altri aspetti visivi dei componenti UI. L'utilizzo del CSS ha permesso di mantenere separati il contenuto e la logica dalla parte estetica, garantendo una maggiore flessibilità e facilità di manutenzione. Inoltre, sono stati creati stili coerenti per assicurare un'esperienza utente gradevole e uniforme in tutte le schermate dell'applicazione.

# –3–

## Versioning e Github

### 3.1 Strumenti di Versioning

Per il versianamento del codice sorgente è stato usato **Git**, in combinazione con la piattaforma remota **GitHub**. Questa scelta ha permesso una tracciabilità accurata delle modifiche, facilitando lo sviluppo dell'applicativo e la collaborazione tra i due membri del team.

TonyKing1510 Sistematico cose		
	db4c09a · last week	74 Commits
idea	feat: add enter button as a valid input when fields are corre...	3 months ago
Applicativo	Sistematico cose	last week
Documentazioni	Sistemato il model e sistemato anche il back	2 months ago

Figura 3.1: Repository di Github del progetto