# ORTO BOTANICO

Relazione sul progetto di Basi di Dati

Gruppo: I Botanici
Mirko Gibin, matr. 1102450
Federico Caldart, matr. 1097005

Database caricato con username **mgibin**
(Mirko Gibin, 1102450, mgibin-PR)

# Indice

# 1. Abstract

Si vuole creare un database per la gestione di un orto botanico. Il database cataloga tutte le piante ivi presenti raccogliendo le loro informazioni, quali gruppo di appartenenza, provenienza e habitat, caratteristiche (come tipo di foglia e di fiore) ed eventuali utilizzi (industriale, alimentare, farmacologico ecc) e principi contenuti.
Si vogliono fornire inoltre delle procedure per permettere la registrazione di nuovi utenti, dando a questi la possibilità di creare delle proprie liste preferiti.

# 2. Analisi dei requisiti

Il database ruota attorno all'entità delle piante delle quali ci interessano:

- il numero di esemplari presenti nell'orto botanico;
- un nome con cui sono comunemente conosciute;
- un nome scientifico che le identifica univocamente;
  l'altezza media che la pianta può raggiungere, espressa in centimetri;
- una descrizione.

Si vuole indicare il periodo dell'anno più idoneo per:

- la semina,
- la fioritura,
- la raccolta,

considerando che non tutte le piante hanno fiori o frutti e che altre sono prettamente selvatiche e non hanno bisogno di essere seminate;
del periodo dell'anno si vuole indicare la stagione, specificando se inizio, piena o fine stagione.

Le piante considerate si possono dividere in:

- Alghe, vegetali di struttura semplice prive di radice e fusto, presenti prevalentemente in ambienti acquatici;
- Briofite, comunemente conosciute come muschi, prive di fusto, con altezza inferiore ai 6 cm;
- Tracheofite, caratterizzate da fusto, foglie e vere radici, suddivise in:
  - Pteridofite, comprendenti le felci e piante affini, prive di semi e con riproduzione legata alle spore;
  - Spermatofite, piante che producono i semi, a loro volta raggruppate in:
    - Gimnosperme, piante con semi non protetti da frutto;
    - Angiosperme, le quali possiedono il fiore per facilitare l'impollinazione e il frutto per proteggerne il seme e favorirne la disseminazione.

Le piante, in base alla divisione di appartenenza, possono avere o meno:

- fusto, struttura portante delle piante (eretto, rampicante, strisciante…);
- foglie, delle quali si vogliono indicare le seguenti caratteristiche:
  - la composizione (foglie semplici o composte);
  - la forma (ad ago, falcata, cordata…);
  - la superficie (coriacea, tomentosa, scabra);
  - il tipo di margine (ciliato, crenato, dentato);
- radici (ramose, tuberiforme, fascicolate...), organo di ancoraggio e di assorbimento dei nutrienti;
- fiore, organo di riproduzione delle angiosperme, del quale si vuole conoscere:
  - il numero (medio) di petali;
  - l'eventuale nome;
  - il colore;
  - il frutto prodotto, del quale si vuole inserire:
    - l'eventuale nome comune;
    - il colore.

Per ogni pianta si vuole indicare l'eventuale utilizzo e i principi che contiene con i relativi benefici. Una pianta può avere più utilizzi e principi.

Nel database si vogliono includere inoltre informazioni relative ai biomi a cui appartengono le piante in natura, identificandoli tramite un nome.
I biomi possono essere:

- acquatici, che possono distinguersi in
  - bioma di acqua dolce;
  - bioma marino;

  e dei quali si vuole conoscere
  - la temperatura dell’acqua;
  - la salinità (che sarà 0 nei biomi di acqua dolce)
- terrestri, dei quali si vuole conoscere:
  - il clima caratteristico;
  - le precipitazioni medie (in millimetri);
  - il tipo di terreno caratteristico;

Per rintracciare geograficamente le piante si vuole creare un'entità che raccoglie informazioni relative alla loro provenienza, indicando:

- il nome dello stato per quanto riguarda i biomi terrestri;
- il nome dello stato e del fiume, lago, mare o oceano per i biomi acquatici.

Ogni paese può avere più biomi.

Ogni bioma viene riprodotto in una sezione dell’orto botanico, rappresentata dal nome.

Il database prevede una entità per gli utenti iscritti contenente:

- Username;
- Password;
- E-mail;
- Nome;
- Cognome.

Si vuole dare la possibilità agli utenti di creare fino a 5 liste preferiti, delle quali si può decidere il nome.

# 3. Progettazione Concettuale

## 3.1 Lista delle Classi

**Piante**
Classe attorno alla quale ruota il database; raccoglie tutte le informazioni inerenti ad una particolare pianta dell'orto botanico, in modo da fornire una descrizione abbastanza esaustiva di essa. E' una superclasse per Alga, Briofita, Tracheofita.

- *NomeS -string-*
- *NomeC -string-*
- *NumEsemplari -int-*
- *Altezza -int- (in centimetri)*
- *Descrizione -string-*
- *Utilizzo – enum (alimentare, aromatico, farmaceutico, tessile, falegnameria, legname, ornamentale)-*

**Stagioni**
Classe che elenca le varie fasi dell'anno, specificando stagione e periodo.

- *Stagione -enum("Primavera", "Estate", "Autunno", "Inverno")-*
- *Periodo -enum("Inizio", "Piena", "Fine")-*

**Alghe**
La classe è una specializzazione di Pianta, in cui sono contenute tutte le piante conosciute come alghe. Non possiede attributi.

**Briofite**
La classe è una specializzazione di Pianta, in cui sono contenute tutte le piante conosciute come briofite. Non possiede attributi.

**Tracheofite**
La classe è una specializzazione di Pianta, in cui sono contenute tutte le piante conosciute come tracheofite. A differenza delle precedenti, questo tipo di pianta possiede fusto, foglie e vere radici. E' una superclasse per Pteridofite e Spermatofite.

- *Fusto -enum {*"eretto", "rampicante", "strisciante", "rampante"}-
- *Radice -enum{"ramosa", "tuberiforme", "fascicolata", "a fittone", "avventizia", "aerea"}-*

**Pteridofite**
La classe è una specializzazione di Tracheofite, in cui sono contenute tutte le piante che si riproducono tramite spore. Non possiede attributi.

**Spermatofite**
La classe è una specializzazione di Tracheofite. Questo tipo di pianta si differenzia dalle Pteridofite in quanto produce semi. Non possiede attributi.
Spermatofite è a sua volta una superclasse per Gimnosperme e Angiosperme.

**Gimnosperme**
La classe è una specializzazione di Spermatofite, in cui sono contenute tutte le piante che producono semi non protetti da frutto. Non possiede attributi.

**Angiosperme**
La classe è una specializzazione di Spermatofite, in cui sono contenute tutte le piante che possiedono fiore e proteggono i semi prodotti con il frutto. Non possiede attributi.

**Foglie**
La classe raccoglie le informazioni relative alle foglie possedute dalla pianta (se tracheofita).

- *Superficie -enum(“Coriacea”, “Tomentosa”, “Scabra”)-*
- *Forma -enum(“Aghiforme”, “Falcata”, “Orbicolare”, “Romboidale”, “Acuminata”, “Flabellata”, “Ovata”, “Rosetta”, “Alternata”, “Astata”, “Palmata”, “Spatolata”, “Aristata”, “Lanceolata”, “Pedata”, “Sagittata”, “Bipennata”, “Lineare”, “Peltata”, “Lesiniforme”, “Cordata”, “Lobulata”, “Amplessicaule”, “Tripartita”, “Cuneiforme”, “Obcordata”, “Impari-pennata”, “Tripennata”, “Triangolare”, “Obovata”, “Paripennata”, “Troncata”, “Digitata”, “Ottusa”, “Pennatisecta”, “Intera”, “Ellittica”, “Opposte”, “Reniforme”, “Verticillata”)-*
- *Margine -enum(“Ciliato”, “Crenato”, “Dentato”, “Denticolato”, “Doppiamente dentato”, “Intero”, “Lobato”, “Seghettato”, “Finemente seghettato”, “Sinuato”, “Spinoso”, “Ondulato”)-*
- *Composizione -enum{“composta”, “semplice”}-*

**Fiori**
La classe raccoglie le informazioni relative al fiore.

- *Nome –string-*
- *NumPetali -int-*
- *Colore -string-*

**Frutti**
La classe Frutti raccoglie le informazioni relative al frutto prodotto dalle piante Angiosperme.

- *Nome -string-*
- *Colore -string-*

**Principi**
La classe Principi contiene il nome dei principi contenuti da una pianta e il beneficio che porta la sua assunzione.

- *Nome -string-*
- *Beneficio -string-*

**Biomi**
La classe Biomi raccoglie le informazioni relative ai biomi in cui si sviluppa in natura una pianta.
Biomi è una superclasse per BiomaAcquatico e BiomaTerrestre.

- *Nome -string-*

**BiomiAcquatici**
La classe è una specializzazione di Biomi e identifica quelli in cui la presenza d'acqua prevale su quella di terraferma.

- *TemperaturaAcqua -int-*
- *Salinità -float-*

**BiomiTerrestri**
La classe è una specializzazione di Biomi e identifica quelli in cui la presenza di terraferma prevale su quella d'acqua.

- *Clima -string-*
- *Precipitazioni -float-*
- *Terreno –enum {"argilloso", "sabbioso", "limoso", "torboso"}-*

**Stati**
La classe Stati raccoglie i dati relativi alla provenienza di una pianta.

- *Nome -string-*

**SezioniOrto**
La classe SezioniOrto specifica la sezione dell'orto botanico dedicata ad un bioma.

- *NomeSez -string-*

**Utenti**
La classe Utenti raccoglie le informazioni relative agli utenti iscritti.

- *Username -string-*
- *Password -string-*
- *Email -string-*
- *Nome -String-*
- *Cognome -String-*

**ListePreferiti**
La classe ListePreferiti svolge la funzione di raccogliere i nome delle liste create da un utente.

- *Nome -string-*
- *Data Creazione -data-*

## 3.2 Relazioni

Piante – Stagioni: **Semina.** Relazione N:N

- Una pianta può essere seminata in nessuna o più stagioni;
- In una stagione possono essere seminate nessuna o più piante.

Piante – Stagioni: **Raccolta.** Relazione N:N

- Una pianta può essere raccolta in nessuna o più stagioni;
- In una stagione possono essere raccolte nessuna o più piante.

Piante – ListePreferiti: **Composizione.** Relazione N:N

- Una pianta può comporre nessuna o più liste preferiti;
- Una lista preferiti è composta da nessuna o più piante.

Biomi – SezioniOrto: **Localizzazione**. Relazione 1:1

- Un bioma può essere localizzato in una e una sola sezione dell'orto;
- Una sezione dell'orto localizza un solo bioma.

Piante – Biomi – Stati: **Presenza.** Relazione N:N

- Una pianta è presente in almeno uno stato;
- In uno stato è presente almeno una pianta.

- Un bioma è presente in almeno uno stato;
- In uno stato è presente almeno un bioma.

- Una pianta è presente in almeno un bioma;
- In un bioma è presente almeno una pianta.

Piante – Principi: **Appartenenza.** Relazione N:N

- Ad una pianta possono appartenere nessuno o più principi;
- Un principio appartiene ad almeno una pianta.

ListePreferiti-Utenti: **Creazione.** Relazione 1:N

- Un utente può creare nessuna o più liste preferiti;
- Una lista preferiti è creata da uno e un solo utente.

Foglie – Tracheofite: **Possesso.** Relazione 1:1

- Un tipo di foglia è posseduto da una e una sola pianta;
- Una pianta possiede uno e uno solo tipo di foglia.

Angiosperme – Fiori: **Riproduzione.** Relazione 1:1

- Una angiosperma si riproduce tramite uno e un solo fiore (dunque consideriamo fiori maschili e femminili allo stesso modo);
- Un fiore è organo riproduttivo di una e una sola pianta.

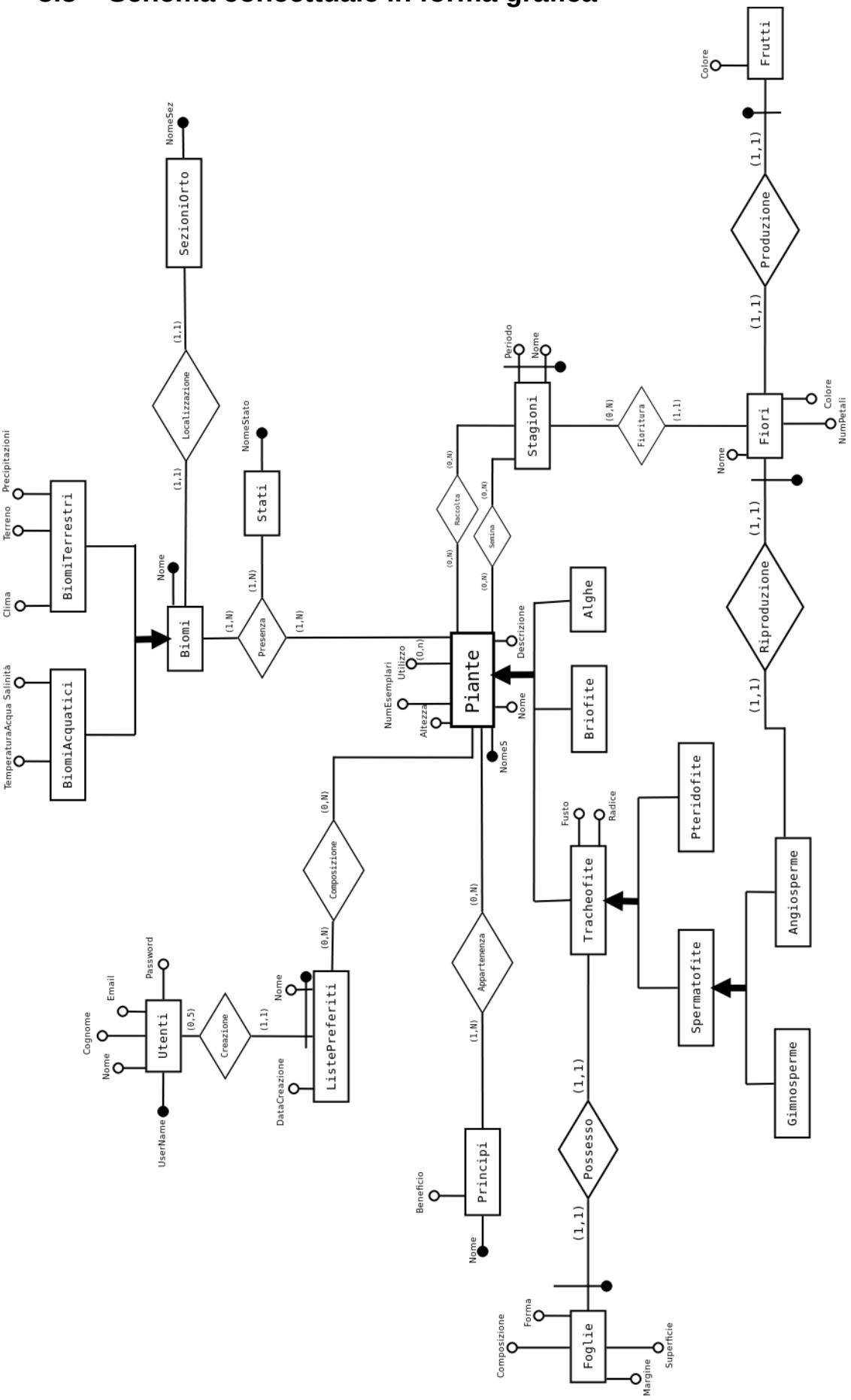
Fiori – Stagioni: **Fioritura.** Relazione 1:N

- Un fiore può fiorire in una e una sola stagione;
- In una stagione può fiorire nessuno o più fiori.

Fiori – Frutti: **Produzione.** Relazione 1:1

- Un fiore produce uno e un solo frutto;
- Un frutto è prodotto da uno e un solo fiore.

## 3.3 – Schema concettuale in forma grafica



# Progettazione Logica

## 4.1- Considerazioni sulla ristrutturazione dello schema

### 4.1.1- Trasformazione delle gerarchie:

- Nel modello concettuale è presente una gerarchia avente Piante come superclasse di Tracheofite, Briofite ed Alghe; Tracheofite a sua volta è una superclasse di Pteridofite e Spermatofite e quest'ultima ha come sottoclassi Gimnosperme e Angiosperme. Nello schema relazionale, essa è stata resa accorpando le figlie nel genitore: così facendo, sono stati aggiunti gli attributi "Gruppo", "Fusto" e "Radice" all'entità Piante; inoltre, per evitare la ridondanza di Divisione e Sottodivisione in Piante, si è deciso di creare una nuova entità chiamata Gruppi, la quale contiene le varie combinazioni possibili di Gruppo, Divisione e SottoDivisione e assegna ad ognuna di esse una chiave sintetica, per renderne più rapida la ricerca.E' stata scelta questa soluzione perché anche se ci saranno valori nulli negli attributi "Fusto" e "Radice", avremo un numero minore di accessi rispetto al caso in cui gli attributi di Pianta fossero distribuiti tra tutte le figlie: la presenza di troppe entità avrebbe complicato inutilmente il database e avrebbe allungato i tempi di ricerca di una particolare pianta.

- Nel modello concettuale è presente una seconda gerarchia, che vede Biomi come genitore delle entità BiomiAcquatici e BiomiTerrestri.
  A differenza del caso precedente, è stato scelto di accorpare l'entità genitore nelle figlie, aggiungendo l'attributo chiave alle due sottoclassi; ciò permette di ridurre i valori nulli in quanto le due entità sono ben distinte.
  Questa scelta ha portato a creare due associazioni, PresenzaT e PresenzaA, che sostituiscono Presenza: esse servono rispettivamente a legare i biomi terrestri ed i biomi acquatici allo stato ed alle sezione dell'orto a cui appartengono.

### 4.1.2- Altre modifiche

- Le relazioni Raccolta, Semina e Fioritura sono state implementate come entità per un migliore ordinamento dei dati. Nella tabella Stagioni è stata inserita una chiave sintetica chiamata ID che permette di identificare univocamente una stagione e un periodo. Tale ID è usato come chiave esterna in Raccolta, Semina e Fioritura per evitare ridondanza dei dati.

- L'entità SezioneOrto, collegata con Biomi da una relazione 1:1 e composta da un unico attributo, è stata accorpata nelle due nuove entità BiomiAcquatici e BiomiTerrestri.

- L'attributo multivalore “utilizzi” è stato implementato creando l'entità PianteUtilizzi.

## 4.2. Schema relazionale

Sono definite le seguenti entità:

(Legenda: **Entita**, Primary Key, Attributo, Attributo* (può essere NULL))

**Piante (**NomeS, NomeC, NumEsemplari, Altezza, Descrizione, Gruppo, Fusto*, Radice*);
*Chiave esterna gruppo riferita a Gruppi.*

**Gruppi** (Id, Gruppo, Divisione*, Sottodivisione*);

**Foglie** (Pianta**,** Forma, Composizione, Margine, Superficie);
*Chiave esterna Pianta riferita a Piante.*

**Principi** (Nome, Beneficio);

**PrincipiPiante** (Principio, Pianta*) ;*
*Chiave esterna Pianta riferita a Piante.*

**UtilizziPiante** (Pianta, Utilizzo);
*Chiave esterna Pianta riferita a Piante.*

**ListePiante** (Pianta, Utente, Lista);
*Chiave esterna Pianta riferita a Piante;*
*Chiave esterna Utente riferita a Utenti;*
*Chiave esterna Lista riferita a ListePreferiti.*

**ListePreferiti** (Utente, Nome, DataCreazione);
*Chiave esterna Utente riferita ad Utenti.*

**Utenti** *(*Username, E-mail, Password, Nome, Cognome*);*

**BiomiTerrestri** (Nome, Clima, Precipitazioni, Terreno, SezioneOrto);

**BiomiAcquatici** (Nome, Temperatura, Salinità, SezioneOrto);

**PresenzeT** (BiomaT, Stato, Pianta);
*Chiave esterna BiomaT riferita a BiomiTerrestri;*
*Chiave esterna Pianta riferita a Piante;*
*Chiave esterna Stato riferita a Stati.*

**PresenzeA** (BiomaA, Stato, Pianta);
*Chiave esterna BiomaA riferita a BiomiAcquatici;*
*Chiave esterna Stato riferita a Stati,*
*Chiave esterna Pianta riferita a Piante.*

**Stati** (Nome);

**Stagione** (ID, Periodo, Stagione);

**Raccolta** (Pianta, Periodo);
*Chiave esterna Pianta riferita a Piante;*
*Chiave esterna Periodo riferita a Stagione.*

**Semina** (Pianta, Periodo);
*Chiave esterna Pianta riferita a Piante;*
*Chiave esterna Periodo riferita a Stagione.*

**Fioritura** (Pianta, Periodo);
*Chiave esterna Pianta riferita a Fiori;*
*Chiave esterna Periodo riferita a Stagione.*

**Fiori** (Pianta**,** Nome*, NumPetali, Colore);
*Chiave esterna Pianta riferita a Piante.*

**Frutti** (Pianta, Colore, Nome);
*Chiave esterna Pianta riferita a Fiore;*
*Chiave esterna Periodo riferita a Stagione*

# 5. Implementazione dello schema logico

```
1.        SET foreign_key_checks = 0;
2.
3.        DROP TABLE IF EXISTS Gruppi;
4.        DROP TABLE IF EXISTS Piante;
5.        DROP TABLE IF EXISTS Foglie;
6.        DROP TABLE IF EXISTS Principi;
7.        DROP TABLE IF EXISTS PrincipiPiante;
8.        DROP TABLE IF EXISTS UtilizziPiante;
9.        DROP TABLE IF EXISTS Utenti;
10.       DROP TABLE IF EXISTS ListePreferiti;
11.       DROP TABLE IF EXISTS ListePiante;
12.       DROP TABLE IF EXISTS Stagioni;
13.       DROP TABLE IF EXISTS Semina;
14.       DROP TABLE IF EXISTS Raccolta;
15.       DROP TABLE IF EXISTS Fioritura;
16.       DROP TABLE IF EXISTS Frutti;
17.       DROP TABLE IF EXISTS Fiori;
18.       DROP TABLE IF EXISTS PresenzeA;
19.       DROP TABLE IF EXISTS PresenzeT;
20.       DROP TABLE IF EXISTS BiomiAcquatici;
21.       DROP TABLE IF EXISTS BiomiTerrestri;
22.       DROP TABLE IF EXISTS Stati;
23.       DROP TABLE IF EXISTS Amministratori;
24.
25.       CREATE TABLE Gruppi (
26.           Id VARCHAR(3) PRIMARY KEY,
27.           Gruppo VARCHAR(20) NOT NULL,
28.           Divisione VARCHAR(20),
29.           Sottodivisione VARCHAR(20)
```

```
30.       );
31.
32.       CREATE TABLE Piante (
33.           NomeScientifico VARCHAR(50) PRIMARY KEY,
34.           NomeComune VARCHAR(50),
35.           NumEsemplari SMALLINT(3) DEFAULT 0,
36.           AltezzaCm SMALLINT(5) DEFAULT 0,
37.           Descrizione VARCHAR(2000) DEFAULT NULL,
38.           Gruppo VARCHAR(3) REFERENCES Gruppi (Id)
39.           ON DELETE CASCADE ON UPDATE CASCADE,
40.           Fusto ENUM('Eretto', 'Rampicante', 'Strisciante', 'Rampante'),
41.           Radice ENUM('Ramosa', 'Tuberiforme', 'Fascicolata', 'A fittone',
42.           'Avventizia', 'Aerea'),
43.           FOREIGN KEY (Gruppo)
44.               REFERENCES Gruppi (Id)
45.               ON UPDATE CASCADE ON DELETE NO ACTION
46.       );
47.
48.       CREATE TABLE Foglie (
49.           Pianta VARCHAR(50) PRIMARY KEY REFERENCES
50.       Piante (NomeScientifico)
51.           ON UPDATE CASCADE ON DELETE CASCADE,
52.           Forma ENUM('Aghiforme', 'Falcata',
53.       'Orbicolare', 'Romboidale', 'Acuminata',
54.       'Flabellata', 'Ovata', 'Rosetta', 'Alternata',
55.       'Astata', 'Palmata', 'Spatolata', 'Aristata',
56.       'Lanceolata', 'Pedata', 'Sagittata', 'Bipennata',
57.       'Lineare', 'Peltata', 'Lesiniforme', 'Cordata',
58.       'Lobulata', 'Amplessicaule', 'Tripartita',
59.       'Cuneiforme', 'Obcordata', 'Impari-pennata',
60.       'Tripennata', 'Triangolare', 'Obovata',
61.       'Paripennata', 'Troncata', 'Digitata', 'Ottusa',
62.       'Pennatisecta', 'Intera', 'Ellittica', 'Opposte',
63.       'Reniforme', 'Verticillata'),
64.           Composizione ENUM('Composta', 'Semplice'),
65.           Margine ENUM('Ciliato', 'Crenato', 'Dentato',
66.       'Denticolato', 'Doppiamente dentato', 'Intero',
67.       'Lobato', 'Seghettato', 'Finemente seghettato',
68.       'Sinuato', 'Spinoso', 'Ondulato'),
69.           Superficie ENUM('Coriacea', 'Tomentosa',
70.       'Scabra')
71.       );
72.
73.       CREATE TABLE Principi (
74.           Nome VARCHAR(20) PRIMARY KEY,
75.           Beneficio VARCHAR(500)
76.       );
77.
78.       CREATE TABLE PrincipiPiante (
79.           Principio VARCHAR(20) REFERENCES Principi (Nome)
80.           ON UPDATE CASCADE,
81.           Pianta VARCHAR(50) REFERENCES Piante
82.       (NomeScientifico)
83.           ON UPDATE CASCADE,
84.           PRIMARY KEY (Principio , Pianta)
85.       );
86.
87.       CREATE TABLE UtilizziPiante (
88.           Pianta VARCHAR(50) REFERENCES Piante (NomeScientifico)
89.           ON UPDATE CASCADE,
90.           Utilizzo ENUM('alimentare', 'aromatico', 'farmacologico',
```

```
91.       'tessile', 'legname', 'ornamentale'),
92.           PRIMARY KEY (Pianta , Utilizzo)
93.       );
94.
95.       CREATE TABLE Utenti (
96.           Username VARCHAR(20) PRIMARY KEY,
97.           Email VARCHAR(30) UNIQUE NOT NULL,
98.           Pass VARCHAR(41) NOT NULL,
99.           Nome VARCHAR(20) NOT NULL,
100.          Cognome VARCHAR(20) NOT NULL
101.      );
102.
103.      CREATE TABLE ListePreferiti (
104.         Utente VARCHAR(30),
105.         Nome VARCHAR(20) NOT NULL,
106.         DataCreazione TIMESTAMP,
107.          PRIMARY KEY (Utente, Nome),
108.         FOREIGN KEY (Utente) REFERENCES Utenti(Username) ON UPDATE
109.      CASCADE ON DELETE CASCADE
110.      );
111.
112.      CREATE TABLE ListePiante (
113.          Pianta VARCHAR(50),
114.          Utente VARCHAR(30),
115.         Lista VARCHAR(20),
116.          PRIMARY KEY (Pianta, Utente, Lista),
117.          FOREIGN KEY (Pianta) REFERENCES Piante(NomeScientifico) ON
118.      UPDATE CASCADE ON DELETE CASCADE,
119.          FOREIGN KEY (Utente,Lista) REFERENCES
120.      ListePreferiti(Utente,Nome) ON UPDATE CASCADE ON DELETE
121.      CASCADE
122.      );
123.
124.      CREATE TABLE BiomiTerrestri (
125.          Nome VARCHAR(20) PRIMARY KEY,
126.          Clima ENUM('Mediterraneo', 'Tropicale', 'Equatoriale',
127.      'Subtropicale', 'Temperato', 'Temperato Umido', 'Oceanico',
128.      'Continentale', 'Subartico', 'Transiberiano', 'Polare',
129.      'Nivale', 'Glaciale', 'Steppico', 'Desertico', 'Monsonico',
130.      'Sinico', 'Della Savana', 'Alpino', 'Boreale', 'Boreale delle
131.      foreste', 'Della tundra'),
132.          Precipitazioni INT,
133.          Terreno ENUM('argilloso', 'sabbioso', 'limoso',
134.      'torboso'),
135.          SezioneOrto VARCHAR(5)
136.      );
137.
138.      CREATE TABLE BiomiAcquatici (
139.          Nome VARCHAR(20) PRIMARY KEY,
140.          Temperatura TINYINT,
141.          Salinità TINYINT,
142.          SezioneOrto VARCHAR(5)
143.      );
144.
145.      CREATE TABLE Stati (
146.          Nome VARCHAR(20) PRIMARY KEY
147.      );
148.
149.      CREATE TABLE PresenzeT (
150.          BiomaT VARCHAR(20) REFERENCES BiomiTerrestri (Nome)
151.          ON DELETE CASCADE ON UPDATE CASCADE,
```

```
        Stato VARCHAR(20) REFERENCES Stati (Nome)
        ON DELETE CASCADE ON UPDATE CASCADE,
        Pianta VARCHAR(50) REFERENCES Piante (NomeScientifico)
        ON UPDATE CASCADE ON DELETE CASCADE,
        PRIMARY KEY (BiomaT , Stato , Pianta)
    );

    CREATE TABLE PresenzeA (
        BiomaA VARCHAR(20) REFERENCES BiomiAcquatici (Nome)
        ON DELETE CASCADE ON UPDATE CASCADE,
        Stato VARCHAR(20) REFERENCES Stati (Nome)
        ON DELETE CASCADE ON UPDATE CASCADE,
        Pianta VARCHAR(50) REFERENCES Piante (NomeScientifico)
        ON UPDATE CASCADE ON DELETE CASCADE,
        PRIMARY KEY (BiomaA , Stato , Pianta)
    );

    CREATE TABLE Fiori (
        Pianta VARCHAR(50) PRIMARY KEY REFERENCES Piante
    (NomeScientifico)
        ON UPDATE CASCADE ON DELETE CASCADE,
        Nome VARCHAR(20),
        NumPetali TINYINT,
        Colore ENUM('Rosso', 'Viola', 'Lilla', 'Giallo',
    'Arancione', 'Bianco', 'Rosa', 'Azzurro', 'Blu', 'Verde')
    );

    CREATE TABLE Frutti (
        Nome VARCHAR(20),
        Colore ENUM('Rosso', 'Viola', 'Lilla', 'Giallo',
    'Arancione', 'Bianco', 'Rosa', 'Grigio', 'Verde', 'Marrone','Blu'),
        Pianta VARCHAR(50) PRIMARY KEY REFERENCES Fiori (Pianta)
        ON UPDATE CASCADE ON DELETE CASCADE
    );

    CREATE TABLE Stagioni (
        ID
    ENUM('IP','PP','FP','IE','PE','FE','IA','PA','FA','II','PI','F
    I') PRIMARY KEY,
        Periodo ENUM('Inizio', 'Piena', 'Pieno', 'Fine'),
        Stagione ENUM('Primavera', 'Estate', 'Autunno', 'Inverno')
    );


    CREATE TABLE Semina (
         Pianta VARCHAR(50) REFERENCES Piante(NomeScientifico)
         ON DELETE CASCADE ON UPDATE CASCADE,
         Periodo CHAR(2) REFERENCES Stagioni(ID)
         ON DELETE NO ACTION ON UPDATE CASCADE,
         PRIMARY KEY (Pianta, Periodo)
    );


    CREATE TABLE Fioritura (
         Pianta VARCHAR(50) REFERENCES Fiori(Pianta)
         ON DELETE CASCADE ON UPDATE CASCADE,
         Periodo CHAR(2) REFERENCES Stagioni(ID)
         ON DELETE NO ACTION ON UPDATE CASCADE,
         PRIMARY KEY (Pianta, Periodo)
    );

```

```
213.      CREATE TABLE Raccolta (
214.           Pianta VARCHAR(50) REFERENCES Piante(NomeScientifico)
215.           ON DELETE CASCADE ON UPDATE CASCADE,
216.           Periodo CHAR(2) REFERENCES Stagioni(ID)
217.           ON DELETE NO ACTION ON UPDATE CASCADE,
218.           PRIMARY KEY (Pianta, Periodo)
219.      );
220.
221.      SET foreign_key_checks = 1;
```

# 6. View, Procedure, Funzioni e Trigger

## 6.1 View

### 6.1.1 - UseNomePro

Trovare il nome scientifico e gli utilizzi delle diverse piante che compongono le liste create da utenti che hanno creato almeno 2 liste composte da almeno 4 piante:

```
1.        CREATE VIEW UseNomePro AS
2.        SELECT DISTINCT(CONCAT(LP.pianta, ', utilizzo: ', U.utilizzo))
3.        as Piante_Ricercate
4.        FROM  ListePiante LP, UtilizziPiante U, ListePreferiti LPR
5.        WHERE LP.pianta=U.pianta AND LPR.Utente=LP.Utente
6.        AND LPR.Utente IN(
7.        SELECT PR.Utente
8.        FROM ListePreferiti PR
9.        WHERE (PR.Nome, PR.Utente) IN(
10.       SELECT LP1.Lista, LP1.Utente
11.       FROM ListePiante LP1
12.       GROUP BY LP1.Lista, LP1.Utente
13.       HAVING COUNT(LP1.Pianta) >= 4)
14.       GROUP BY PR.Utente
15.       HAVING COUNT(PR.Nome) >= 2);
```

OUTPUT:

```
+-----------------------------------------------+
| Piante_Ricercate                              |
+-----------------------------------------------+
| Elaeocarpus Angustifolius, utilizzo: alimentare  |
| Elaeocarpus Angustifolius, utilizzo: ornamentale |
| Eriobotrya Japonica, utilizzo: alimentare        |
| Eriobotrya Japonica, utilizzo: farmacologico     |
| Gossypium Herbaceum, utilizzo: tessile           |
| Juniperus Communis, utilizzo: alimentare         |
| Cedrus Libani, utilizzo: ornamentale             |
| Porphyra Tenera, utilizzo: alimentare            |
| Eucalyptus, utilizzo: farmacologico              |
| Eucalyptus, utilizzo: legname                    |
| Eucalyptus, utilizzo: ornamentale                |
| Melaleuca Alternifolia, utilizzo: farmacologico  |
+-----------------------------------------------+
```

### 6.1.2 - OvUnAlt

Trovare tutte le piante terrestri con un' altezza superiore a 200cm, che crescono in un clima Monsonico e producono un fiore di colore bianco; oltre a queste, voglio estrarre le piante di acqua dolce che stanno invece sotto i 100cm di altezza e crescono in un clima dalla temperatura superiore ai 20 gradi:

```
1.        CREATE VIEW OvUnAlt AS
2.        SELECT PT.BiomaT AS Bioma, P.NomeScientifico, P.Altezzacm AS Altezza
1.        FROM  Piante P, PresenzeT PT, BiomiTerrestri BT
2.        WHERE P.NomeScientifico=PT.Pianta AND BT.Nome=PT.BiomaT AND
3.        BT.Clima='Temperato'
4.        AND P.NomeScientifico IN (
5.        SELECT Pianta
6.        FROM  Fiori
7.        WHERE Colore='Bianco') AND P.Altezzacm>200
8.        UNION SELECT PA.BiomaA AS Bioma, P.NomeScientifico, P.Altezzacm AS
9.        Altezza
10.       FROM Piante P, PresenzeA PA, BiomiAcquatici BA
11.       WHERE  Salinità=0 AND Temperatura>=20 AND Altezzacm<100
12.       AND PA.BiomaA=BA.Nome
13.       AND P.NomeScientifico=PA.Pianta;
```

OUTPUT:

| Bioma | NomeScientifico | Altezza |
|---|---|---|
| Foresta Temperata | Elaeocarpus Angustifolius | 350 |
| Foresta Temperata | Eucalyptus | 300 |
| Foresta Temperata | Melaleuca Alternifolia | 600 |
| Fiume Mississipi | Vallisneria Gigantea | 50 |

### 6.1.3 - PrinEff

Estrarre i principi e gli effetti delle piante il cui raccolto non avviene in estate e appartenenti alla sottodivisione piu' numerosa:

```
1.        CREATE VIEW Conta AS
2.        SELECT G.Sottodivisione as Sottodivisione, COUNT(P.NomeScientifico)
as Numero
3.        FROM Piante P JOIN Gruppi G ON (P.Gruppo=G.Id)
4.        WHERE G.Sottodivisione IS NOT NULL
5.        GROUP BY G.Sottodivisione;
6.
7.        CREATE VIEW PrinEff AS
8.        SELECT P.Nome as Principio, P.Beneficio, I.NomeScientifico as Pianta
9.        FROM Gruppi G, Conta, Piante I
10.       LEFT JOIN PrincipiPiante PP ON I.NomeScientifico=PP.Pianta JOIN
Principi P
```

```
11.       ON P.Nome=PP.Principio
12.       WHERE I.Gruppo=G.Id AND I.NomeScientifico NOT IN (SELECT
Raccolta.Pianta
13.       FROM Raccolta JOIN Stagioni ON Raccolta.Periodo=Stagioni.ID
14.       WHERE Stagioni.Stagione='Estate') AND
Conta.Sottodivisione=G.Sottodivisione
15.       AND Conta.Numero=(SELECT MAX(Numero)
16.       FROM Conta)
17.       ORDER BY Principio;
```

OUTPUT:

| Principio | Beneficio | Pianta |
|---|---|---|
| Cineolo | Antimicotico | Melaleuca Alternifolia |
| Flavonoidi | Antiossidante | Eriobotrya Japonica |
| Flavonoidi | Antiossidante | Eucalyptus |
| Olio Essenziale | Analgesico | Eucalyptus |
| Olio Essenziale | Analgesico | Melaleuca Alternifolia |
| Terpinolo | Antisettico | Melaleuca Alternifolia |

### 6.1.4 - StAlim

Estrarre lo stato da cui provengono le piante che possiedono esattamente due principi e che hanno solo un utilizzo alimentare:

```
1.          CREATE VIEW StAlim AS
2.          SELECT DISTINCT(S.Nome)
3.          FROM Stati S, PresenzeT T, PresenzeA A, Piante P
4.          WHERE ((S.Nome=T.Stato AND P.NomeScientifico=T.Pianta) OR
  (S.Nome=A.Stato
5.          AND P.NomeScientifico=A.Pianta)) AND P.NomeScientifico IN (
6.              SELECT U.Pianta
7.              FROM UtilizziPiante U
8.              WHERE U.Utilizzo='alimentare' and U.Pianta not in (
9.                    SELECT UT.Pianta
10.                   FROM UtilizziPiante UT
11.                   WHERE UT.Utilizzo<>'alimentare'
12.             )) AND P.NomeScientifico IN (
13.             SELECT Distinct(PP.Pianta)
14.             FROM PrincipiPiante PP
15.             GROUP BY PP.Pianta
16.             HAVING COUNT(PP.Principio)=2
17.         );
```

OUTPUT:

| Nome |
|---|
| Francia |
| Germania |

## 6.2. Funzioni

### 6.2.1 - QtaGrup

Funzione che restituisce il numero di piante appartenenti ad un gruppo, famiglia o divisione (in base all'input fornito: G=gruppo, D=divisione, S=sottodivisione) dato in input:

```
1.        DELIMITER $
2.
3.        CREATE FUNCTION QtaGrup(gr char(1), fm VARCHAR(20))
4.        RETURNS SMALLINT UNSIGNED
5.        BEGIN
6.
7.        DECLARE Quantita SMALLINT UNSIGNED;
8.        IF(gr='G') THEN
9.            SELECT COUNT(Piante.NomeScientifico) INTO Quantita
10.           FROM Piante, Gruppi
11.           WHERE Gruppi.Gruppo=fm and Piante.Gruppo=Gruppi.Id;
12.       ELSEIF (gr='D') THEN
13.           SELECT COUNT(Piante.NomeScientifico) INTO Quantita
14.           FROM Piante, Gruppi
15.           WHERE Gruppi.Divisione=fm and Piante.Gruppo=Gruppi.Id;
16.       ELSEIF(gr='S') THEN
17.           SELECT COUNT(Piante.NomeScientifico) INTO Quantita
18.           FROM Piante, Gruppi
19.           WHERE Gruppi.Sottodivisione=fm and Piante.Gruppo=Gruppi.Id;
20.       ELSE
21.       SET Quantita=0;
22.       SIGNAL SQLSTATE VALUE '45000'
23.       SET MESSAGE_TEXT = 'Errore nell''indicazione di Gruppo, Famiglia,
24.       Divisione';
25.       END IF;
26.       RETURN Quantita;
27.
28.       END $
29.
30.       DELIMITER ;
```

### 6.2.2 - Control

Funzione di controllo delle credenziali, riceve in input un nome utente e una password, verifica che siano correlati confrontandoli con quelli salvati nel database e restituisce 1 o 0 a seconda che siano corretti o meno.

```
1.        DELIMITER $
2.
3.        CREATE FUNCTION Control(UT VARCHAR(20), psw VARCHAR(20))
4.        RETURNS BOOL
5.        BEGIN
6.
7.        DECLARE a BOOL;
8.        DECLARE pss VARCHAR(41);
9.        SET pss=password(psw);
```

```
10.       IF((Ut NOT IN (SELECT Username
11.                         FROM Utenti)) OR
12.         (pss<>(SELECT Pass
13.                 FROM Utenti
14.                 WHERE Username=Ut))) THEN
15.         SET a=0;
16.       ELSE
17.         SET a=1;
18.       END IF;
19.       RETURN a;
20.
21.       END$
22.
23.       DELIMITER ;
```

## 6.3 Procedure

**NOTA: Per gli utenti già inseriti la password corrisponde al nome utente.**

### 6.3.1- AddPref

Input richiesto: username, password, lista preferiti, pianta. La procedura permette l'inserimento della pianta desiderata nella lista preferiti indicata. Richiede username e password per effettuare l'inserimento della preferenza in una lista appartente all'utente indicato. Utilizza la funzione Control per verificare la correttezza di username e password e segnala l'eventuale errore.

```
1.        DELIMITER ||
2.
3.        CREATE PROCEDURE AddPref(In Ut VARCHAR(20),In psw VARCHAR(20), IN
listaI
4.        VARCHAR(20), IN piantaS VARCHAR(50))
5.        BEGIN
6.        IF(Control(Ut,psw)) THEN
7.            IF (piantaS IN (SELECT DISTINCT LP.Pianta
8.                                FROM ListePiante LP
9.                                WHERE LP.lista=listaI AND Ut=LP.utente))
THEN
10.           SIGNAL SQLSTATE VALUE '45000'
11.           SET MESSAGE_TEXT = 'Pianta gia'' presente nella lista indicata';
12.           ELSE
13.           INSERT INTO ListePiante values (piantaS, Ut, listaI);
14.            END IF;
15.       ELSE
16.       SIGNAL SQLSTATE VALUE '45000'
17.       SET MESSAGE_TEXT = 'Username o password errati';
18.       END IF;
19.       END ||
20.
21.       DELIMITER ;
```

### 6.3.2- DelPref

Input richiesto: username, password, lista preferiti, pianta
La procedura rimuove la pianta indicata dalla lista preferiti fornita. Qualora la lista o la pianta non esistano, viene generato un messaggio d'errore. Viene utilizzata la funzione Control per verificare la correttezza di username e password.

```
1.        DELIMITER ||
2.
3.        CREATE PROCEDURE DelPref(In Ut VARCHAR(20), In psw VARCHAR(20),IN
listaI
4.        VARCHAR(20), IN piantaS VARCHAR(50))
5.        BEGIN
6.        IF(Control(Ut,psw)) THEN
7.
8.            IF(listaI NOT IN (SELECT DISTINCT LP.Lista
9.                                FROM ListePiante LP
10.                                   WHERE Ut=LP.Utente)) THEN
11.           SIGNAL SQLSTATE VALUE '45000'
12.           SET MESSAGE_TEXT = 'La lista preferiti indicata non esiste';
13.           ELSEIF(piantaS NOT IN (SELECT DISTINCT LP.Pianta
14.                                                 FROM ListePiante LP
15.                                                 WHERE LP.lista=listaI AND
16.       Ut=LP.Utente)) THEN
17.           SIGNAL SQLSTATE VALUE '45000'
18.           SET MESSAGE_TEXT = 'Pianta non presente nella lista indicata';
19.           ELSE
20.           DELETE LP.* FROM ListePiante LP
21.           WHERE listaI=LP.lista and LP.Utente=Ut and LP.pianta=piantaS;
22.            END IF;
23.       ELSE
24.       SIGNAL SQLSTATE VALUE '45000'
25.       SET MESSAGE_TEXT = 'Username o password errati';
26.       END IF;
27.       END ||
28.
29.       DELIMITER ;
```

### 6.3.3- AddList

Input richiesto: username, password, lista preferiti.
La procedura permette di inserire una nuova lista preferiti della quale si indica il nome, dopo aver controllato con Control la correttezza di username e password.

```
1.        DELIMITER ||
2.
3.        CREATE PROCEDURE AddList(In Ut VARCHAR(20), In psw VARCHAR(20), IN
listaI VARCHAR(20))
4.        BEGIN
5.        IF(Control(Ut,psw)) THEN
6.        INSERT INTO ListePreferiti VALUES(Ut,listaI,current_timestamp());
7.        ELSE
8.        SIGNAL SQLSTATE VALUE '45000'
9.            SET MESSAGE_TEXT = 'Username o password errati';
```

```
10.       END IF;
11.       END ||
12.
13.       DELIMITER ;
```

### 6.3.4- DelList

Input richiesto: username, password, lista preferiti.
La procedura permette di cancellare la lista preferiti indicata, dopo aver controllato con la funzione Control la correttezza di username e password.

```
1.
2.        DELIMITER ||
3.        CREATE PROCEDURE DelList(In Ut VARCHAR(20), In psw VARCHAR(20),IN
listaI
4.        VARCHAR(20))
5.        BEGIN
6.        IF(Control(Ut,psw)) THEN
7.        IF(listaI NOT IN (SELECT DISTINCT LP.Nome
8.                                FROM ListePreferiti LP
9.                                   WHERE Ut=LP.Utente)) THEN
10.               SIGNAL SQLSTATE VALUE '45000'
11.           SET MESSAGE_TEXT = 'La lista preferiti indicata non esiste';
12.       ELSE
13.       DELETE LP.*
14.       FROM ListePreferiti LP
15.       WHERE listaI=LP.Nome and LP.Utente=Ut;
16.       END IF;
17.       ELSE
18.       SIGNAL SQLSTATE VALUE '45000'
19.           SET MESSAGE_TEXT = 'Username o password errati';
20.       END IF;
21.       END||
22.
23.       DELIMITER ;
```

### 6.3.5- Registrazione

Input richiesto: username, email, password, nome, cognome.
La procedura permette la registrazione di un nuovo utente.

```
1.     DELIMITER ||
2.
3.     CREATE PROCEDURE Registrazione(IN Ut VARCHAR(20), IN mail
4.     VARCHAR(30), IN
5.     psw VARCHAR(20), IN nome VARCHAR(20), IN cognome VARCHAR(20))
6.     BEGIN
7.     INSERT INTO Utenti VALUES(Ut, mail, password(psw), nome,
       cognome);
8.
9.     END ||
10.
11.    DELIMITER ;
```

### 6.3.6- DelUser

Input richiesto: username, password.
La procedura permette di eliminare l'utente indicato, dopo aver controllato con Control la correttezza di username e password.

```
1.        DELIMITER ||
2.        CREATE PROCEDURE DelUser(IN Ut VARCHAR(20), IN psw VARCHAR(41))
3.        BEGIN
4.        IF(Control(Ut,psw)) THEN
5.        DELETE Utenti.* FROM Utenti WHERE Ut=Username;
6.        ELSE
7.        SIGNAL SQLSTATE VALUE '45000'
8.        SET MESSAGE_TEXT = 'Username o password errati';
9.        END IF;
10.       END ||
11.
12.       DELIMITER ;
```

### 6.3.6- CountPlants

Input richiesto: sezione orto.
Tramite questa procedura è possibile, dato in input il nome di una sezione dell'orto botanico, sapere quante specie di piante contenute in esso hanno il fusto eretto e quante invece diverso da eretto (dunque si considerano solo le piante tracheofite). Oltre a questi dati, la procedura mostra il nome della sezione ricercata e il bioma in essa rappresentato.

```
1.        DELIMITER ||
2.        CREATE PROCEDURE CountPlants (IN sezorto varchar(5))
3.        BEGIN
4.        IF(sezorto NOT IN (SELECT SezioneOrto FROM BiomiTerrestri) AND
sezorto NOT IN (SELECT SezioneOrto
5.        FROM BiomiAcquatici)) THEN
6.        SIGNAL SQLSTATE VALUE '45000'
7.        SET MESSAGE_TEXT = 'La sezione indicata non esiste.';
8.        ELSE
9.        (SELECT IFNULL(PianteFustoEretto, 0) AS PianteFustoEretto,
10.       IFNULL(PianteFustoNonEretto,0) AS PianteFustoNonEretto, Bioma,
SezioneOrto
11.       FROM (
12.       SELECT COUNT(DISTINCT PT.Pianta) AS PianteFustoEretto, PT.BiomaT as
Bioma,
13.       BT.SezioneOrto AS SezioneOrto
14.       FROM BiomiTerrestri BT, PresenzeT PT, Piante P
15.       WHERE BT.Nome=PT.BiomaT AND P.Fusto='Eretto' AND
16.       P.NomeScientifico=PT.Pianta
17.       AND BT.SezioneOrto=sezorto
18.       GROUP BY Bioma
19.       ) AS TAB
20.       LEFT JOIN (
21.       SELECT COUNT(DISTINCT PT1.Pianta) AS PianteFustoNonEretto,
PT1.BiomaT AS Bioma1
22.       FROM PresenzeT PT1, Piante P1
23.       WHERE P1.Fusto<>'Eretto' AND P1.NomeScientifico=PT1.Pianta
24.       GROUP BY Bioma1) AS TAB1
25.       ON Bioma=Bioma1)
```

```
26.       UNION (SELECT IFNULL(PianteFustoEretto, 0) AS PianteFustoEretto,
27.       IFNULL(PianteFustoNonEretto,0) AS PianteFustoNonEretto, Bioma,
SezioneOrto
28.       FROM (
29.       SELECT COUNT(DISTINCT PA.Pianta) AS PianteFustoEretto, PA.BiomaA as
Bioma,
30.       BA.SezioneOrto AS SezioneOrto
31.       FROM BiomiAcquatici BA, PresenzeA PA, Piante P
32.       WHERE BA.Nome=PA.BiomaA AND P.Fusto='Eretto' AND
33.       P.NomeScientifico=PA.Pianta
34.       AND BA.SezioneOrto=sezorto
35.       GROUP BY Bioma
36.       ) AS TABA
37.       LEFT JOIN (
38.       SELECT COUNT(DISTINCT PA1.Pianta) AS PianteFustoNonEretto,
PA1.BiomaA AS Bioma1
39.       FROM PresenzeA PA1, Piante P1
40.       WHERE P1.Fusto<>'Eretto' AND P1.NomeScientifico=PA1.Pianta
41.       GROUP BY Bioma1) AS TABA1
42.       ON Bioma=Bioma1);
43.       END IF;
44.       END ||
45.
46.       DELIMITER ;
```

## 6.4 Trigger

6.3.1- ControlExist

Il trigger si attiva prima dell'inserimento di una pianta in una lista preferiti. Il suo scopo è quello di creare la nuova lista preferiti qualora questa non fosse presente tra le liste preferiti dell'utente.

```
1.        DELIMITER $
2.
3.        CREATE TRIGGER ControlExist
4.        BEFORE INSERT ON ListePiante
5.        FOR EACH ROW
6.        BEGIN
7.        IF
8.        (New.Lista NOT IN (SELECT DISTINCT LP.Nome
9.                           FROM ListePreferiti LP
10.                           WHERE LP.Utente=New.Utente)) THEN
11.       INSERT INTO ListePreferiti VALUES
(New.Utente,New.Lista,current_timestamp);
12.       END IF;
13.       END $
14.
15.       DELIMITER ;
```

### 6.3.2- ControlPref

Il trigger ControlPref si attiva prima dell'inserimento di una nuova lista preferiti, impedendone la creazione tramite un messaggio d'errore qualora l'utente che l'ha richiesta ne possegga già almeno 5.

```
1.    DELIMITER $
2.
3.    CREATE TRIGGER ControlPref
4.    BEFORE INSERT ON ListePreferiti
5.    FOR EACH ROW
6.    BEGIN
7.    DECLARE qta TINYINT;
8.    SELECT COUNT(LP.Nome) INTO qta
9.    FROM ListePreferiti LP
10.   WHERE LP.Utente=New.Utente;
11.   IF(qta>=5) THEN
12.   SIGNAL SQLSTATE VALUE '45000'
13.   SET MESSAGE_TEXT = 'Non si possono inserire più di 5 liste
      preferiti per
14.   utente!';
15.   END IF;
16.   END$
17.
18.   DELIMITER ;
```

### 6.3.3- ControlUser

Il trigger ControlUser si attiva prima della creazione di un nuovo utente e ne impedisce la creazione se:

- il nome utente è già stato utilizzato;
- l'email è già stata utilizzata;
- il formato email non è valido (almeno un carattere + @ + almeno due caratteri + . + almeno due caratteri).

Viene utilizzato in sostituzione del controllo di integrità referenziale del database in quanto fornisce un messaggio più semplice e chiaro dell'errore che si è verificato.

```
1.    DELIMITER $
2.
3.    CREATE TRIGGER ControlUser
4.    BEFORE INSERT ON Utenti
5.    FOR EACH ROW
6.    BEGIN
7.    IF(New.Username IN (SELECT Username FROM Utenti)) THEN
8.    SIGNAL SQLSTATE VALUE '45000'
9.    SET MESSAGE_TEXT = 'Username già utilizzato';
10.   ELSEIF(New.Email NOT LIKE '%_@__%.__%') THEN
11.   SIGNAL SQLSTATE VALUE '45000'
12.   SET MESSAGE_TEXT = 'Formato email non valido';
13.   ELSEIF(New.Email IN (SELECT Email FROM Utenti)) THEN
14.   SIGNAL SQLSTATE VALUE '45000'
15.   SET MESSAGE_TEXT = 'Email già registrata';
16.   END IF;
17.   END$
18.   DELIMITER ;
```