Ghini Documentation

Release 1.0.x

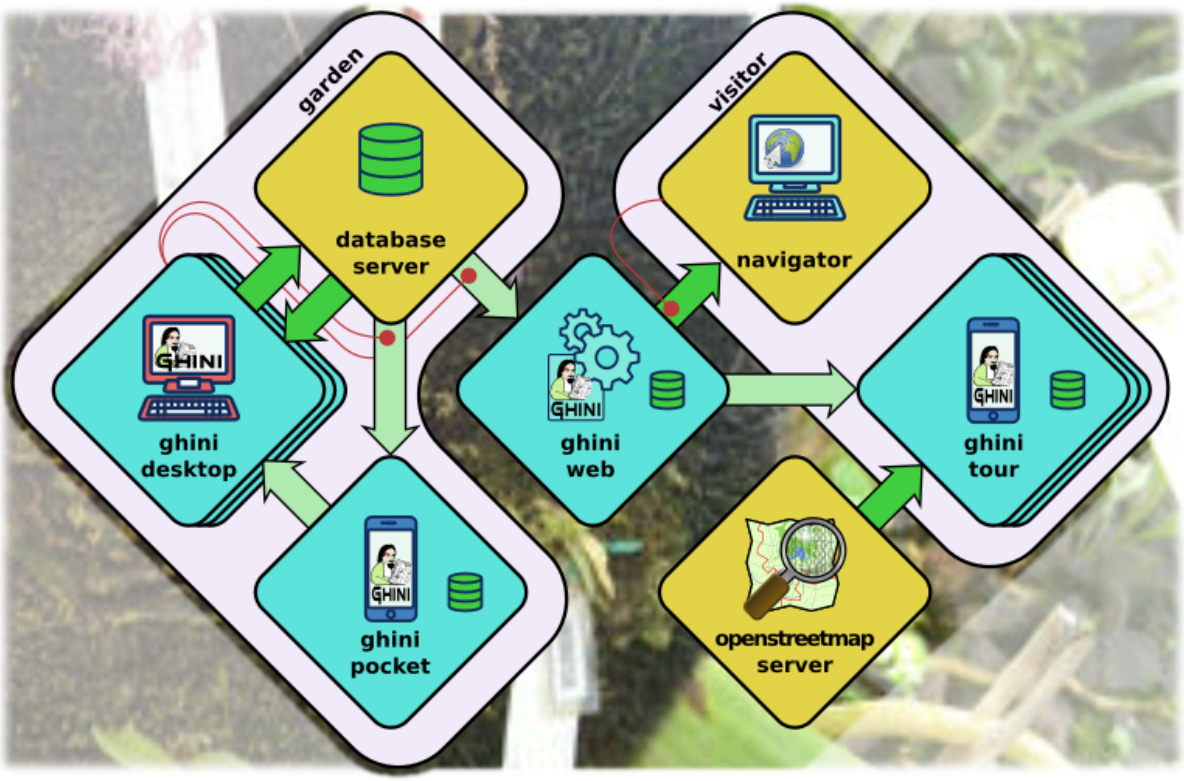
Mario Frasca

gennaio 11, 2023

Indice

Ghini is a suite of applications for managing botanical specimen collections.

- **ghini.desktop** lets you create and query a database representing objects and events in your plant collection.
- **ghini.web** publishes highlights from your database on the web.
- **ghini.pocket** puts a snapshot of your database in your handheld device.
- **ghini.tour** assists garden visitors with a map and spoken virtual panels.



The bulk of this documentation focuses on ghini.desktop. One final chapter presents the rest of the Ghini family: *ghini.pocket*, *ghini.web*, *ghini.tour*, and the *data streams between software components*.

Tutto il software Ghini è aperto e libero ed è distribuito secondo la GPL: Licenza Pubblica GNU. Inoltre, il software Ghini sviluppato secondo il modello servente/cliente segue la licenza GNU Affero Public License.

CAPITOLO 1

Statements

1.1 Obiettivi e punti salienti

Dovreste usare questo software? Questa domanda è per voi di rispondere. Siamo sicuri che se si gestisce una collezione botanica, troverete Ghini eccessivamente utile e ci auguriamo che questa pagina vi convincerà su di esso.

Questa pagina Mostra come Ghini rende il software soddisfare le esigenze di un giardino botanico.

If you already know, and all you want is to do something practical, just install the software, then check our user-contributed recipes.

1.1.1 Giardino botanico

Secondo Wikipedia,» un botanic(al) giardino è un giardino dedicato per la raccolta, la coltivazione e la visualizzazione di una vasta gamma di piante etichettati con i loro nomi botanici «e ancora secondo il Wikipedia,» un giardino è uno spazio progettato, solitamente all'aperto, mettere da parte per la visualizzazione, la coltivazione e il godimento delle piante e altre forme di natura. «

Così abbiamo in un giardino botanico sia lo spazio fisico, il giardino, come sua dinamica, le attività a cui è dedicato il giardino, attività che ci fa chiamare il giardino un giardino botanico.

1.1.2 Giardino botanico Software

A altra estremità del nostro ragionamento abbiamo l'applicazione programma Ghini e ancora una volta citando Wikipedia,» un programma applicativo è un programma per computer

Fig. 1: **il fisico giardino**

Fig. 2: **raccolta attività correlate in giardino**

progettato per eseguire un gruppo di funzioni di coordinamento, attività o attività a vantaggio dell'utente «, o, in breve,» progettato per aiutare le persone a svolgere un'attività «.

Dati e algoritmi all'interno Ghini sono stati progettati per rappresentare lo spazio fisico e la dinamica di un giardino botanico.

Fig. 3: **struttura di database** di Ghini

In the above figure, a simplified view on the database, the highlighted blocks are those relative to objects you definitely need insert in the database.

We distinguish three main sections in the database. Start reading the graph from the right hand side, with the relevant **Taxonomy** information, then step to administering your **Collection**, and finally consider the physical **Garden**.

L'elemento centrale nel punto di vista di Ghini è la accessione (`Accession`). Seguendo i suoi collegamenti agli altri oggetti di database ci permette di comprenderne meglio la struttura:

Accession links Planting to Species

> An `Accession` represents the action of receiving this specific plant material in the garden. As such, `Accession` is an abstract concept, it links physical living `Plantings` —groups of plants placed each at a `Location` in the garden— to the corresponding `Species`. It is not the same as an acquisition from a source, because in a single acquisition you can access material of more than one species. In other words: a single aquisition can embark multiple accessions. An `Accession` has zero or more `Plantings` associated to it (0..n), and it is at all times connected to exactly 1 `Species`. Each `Planting` belongs to exactly one `Accession`, each `Species` may have multiple `Accessions` relating to it.
>
> Una `Accssion` rimane nel database anche se tutti i suoi `Planting` stati rimossi, venduto, o sono morti. Identificare la `Species` di una `Accession` equivale al connettere in modo consistente tutti i suoi `Planting` alla stessa `Species`.

Accession at the base of the history of your plants

> Contatti esterni (`Contact`) o lo stesso orto botanico a traverso di propagazioni (`Propagation`) forniscono materiale vegetale per il giardino; Questa informazione è facoltativa e collezionisti più piccoli potrebbero preferire non approfittare di questa sezione del software. Una prova di `Propagation` potrebbe non riuscire, il più delle volte il risultato sarà esattamente una `Accession`, ma può anche produrre `Species` leggermente differenti, quindi il database permette di zero o più `Accession` a `Propagation` (0.. n). Anche un `Contact` può fornire zero o più `Accession` (0.. n).

Accession and Verification opinions

> Specialisti possono formulare il loro parere circa la "specie" a cui appartiene un "accessione", fornendo una "verifica", firmarlo, e affermando il livello applicabile di fiducia.

Accessing your own Propagations

Se un “accessione” è stato ottenuto nel vivaio giardino da una successo “propagazione”, i link “propagazione” “accessione” e tutti i suoi “planting” ad un unico genitore “piantatura”, il seme o il genitore vegetativo.

Anche dopo la spiegazione di cui sopra, molti nuovi utenti continuano a chiedersi perché hanno bisogno di passare attraverso questo schermo `Accession` mentre non vogliono altro se non inserire una `Plant` nella collezione, e ancora: ma da dove è uscita fuori questa `Accession`? La maggior parte delle discussioni in rete non rende il concetto di niente più chiaro. Uno dei nostri utenti ha dato un esempio che mi fa piacere includere nella documentazione di Ghini.

caso d’uso

1. All’inizio del 2007 abbiamo ottenuto cinque piantine di *Heliconia longa* (una `Specie`) da un nostro vicino (la fonte di `contatto`). Dato che era la prima acquisizione dell’anno, l’abbiamo chiamata 2007.0001, cioè abbiamo dato alle cinque piantine uno stesso codice di `accessione`, con quantità iniziale 5 e le abbiamo piantate tutte insieme in una stessa `ubicazione` come una singola `pianta`, anche questa con quantità 5.
2. Al momento della scrittura, nove anni più tardi, l”`accessione` 2007.0001 ha 6 `piante` distinte, ognuna in una diversa `ubicazione` nel nostro giardino, ottenute per via vegetativa (asessualmente) dalle 5 piante originali. Il nostro unico intervento era separare, spostare e naturalmente scrivere queste informazioni nel database. La quantità totale delle varie `piante` è ora superiore a 40.
3. Nuove `piante` ottenute per `propagazione` sessuale assistita entrano nel nostro database sotto codici di `accessione` diversi, dove il nostro giardino è la fonte di `contatto` e dove sappiamo quale delle nostre `piante` è da considerarsi la madre.

questi tre casi si traducono in diverse brevi storie d’uso:

1. attivare il menu Inserisci → accessione, verificare l’esistenza e la correttezza della specie *Heliconia longa*, specificare la quantità iniziale del’accessione “; aggiungere il suo planting nella posizione preferita.
2. modificare “ planting “ in modo da tener aggiornata la quantità di piante viventi — ripetere questa operazione quando necessario.
3. modificare “ planting “ per dividerlo in sedi separate del “ “ — questo produce un diverso “planting “ sotto la stessa “accessione “.
4. modificare “ planting “ per aggiungere un “ propagazione “ (seme).
5. modificare per aggiornare lo stato della “propagazione “ che “ pianta “.
6. attivare il menu Inserisci → accessione per associare un’accessione a un processo di “ propagazione “ successo; aggiungere la piantatura del “ “ nella posizione desiderata del “ “.

In particolare la capacità di dividere una piantatura del “ “ in diverse località diverse del “ “ e per mantenere tutte le uniformemente associato a una “specie “, o la possibilità di mantenere

informazioni su “planting “ che sono stati rimossi dalla raccolta, contribuire a giustificare la presenza del livello di astrazione di “Accessione “.

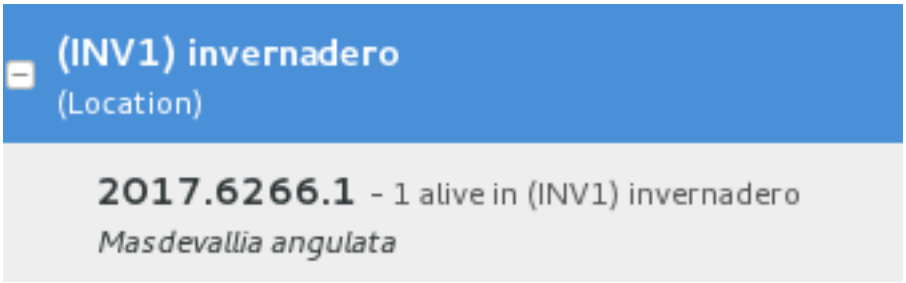
1.1.3 Hypersimplified view

People using Ghini only sporadically may prefer ignoring the database structure and look at it as two nested sequences of objects, each element of the sequence being necessary to add element at the next level.

In order to get down to an Accession, you will need four levels, as in this example:

A quite complete set of Families and Genera are inserted in your database at the moment Ghini initializes it. So all you need is adding Species and Accessions, in this order.

When placing a physical Plant (relative to an Accession) somewhere in the garden, you need to describe this «somewhere» digitally, as a Location in the garden.



1.1.4 Punti principali

non-così-breve elenco delle principali attrattive, destinata stuzzicare l’appetito.

informazioni tassonomiche

When you first start Ghini, and connect to a database, Ghini will initialize the database not only with all tables it needs to run, but it will also populate the taxon tables for ranks family and genus, using the data from the "RBG Kew's Family and Genera list from Vascular Plant Families and Genera compiled by R. K. Brummitt and published by the Royal Botanic Gardens, Kew in 1992". In 2015 we have reviewed the data regarding the Orchidaceae, using "Tropicos, botanical information system at the Missouri Botanical Garden - www.tropicos.org" as a source.

Importa dati

Ghini vi permetterà di importare i dati che hai messo in un formato intermedio json. Che cosa è importare completerà a quello che hai già nel database. Se hai bisogno di aiuto, si può chiedere qualche professionista Ghini per aiutarti a trasformare i dati in formato json intermedio di Ghini.

sinonimi

Ghini vi permetterà che si definiscono sinonimi per specie, generi e famiglie. Anche questa informazione può essere rappresentata nel formato json intermedio ed essere importata in un database esistente di Ghini.

scientifica responsabile

Ghini implementa il concetto di «accessione», intermedio tra planting fisico (o un gruppo di loro) e astratte taxon. Ogni accessione possibile associare le stesse piante di diversi taxa, se due tassonomisti non concordano sull'identificazione: ogni tassonomista possono dire la loro e non è necessario sovrascrivere il lavoro di altri. Tutte le verifiche possono essere trovate nel database, con timestamp e firma.

Aiuta identificazione off-line

Ghini permette di associare immagini alle piante fisiche, questo può aiutare a riconoscere la pianta, nel caso in cui un adesivo è perso, o identificazione tassonomica aiuto se un tassonomista non è disponibile a tutti volte.

esportazioni e rapporti

Ghini ti consente di esportare un report in qualsiasi formato testuale che avete bisogno. Utilizza un motore di template potente denominato "mako", che vi permetterà che si esportano i dati in una selezione in qualsiasi formato che avete bisogno. Una volta installato, un paio di esempi sono disponibili nella sottodirectory mako.

annotare le informazioni

È possibile associare note piante, accessioni, specie,... Note possono essere categorizzati e utilizzati nei rapporti o ricerche.

giardino o erbario

Gestione delle posizioni di pianta.

cronologia del database

Tutte le modifiche nel database viene memorizzato nel database, come log di cronologia. Tutte le modifiche sono "firmate" e timestamp. Ghini rende facile per recuperare l'elenco di tutte le modifiche nell'ultima giornata di lavoro o settimana, o in qualsiasi periodo specifico in passato.

ricerca semplice e potente

Ghini ti permette di effettuare ricerche nel database utilizzando semplici parole chiave, ad esempio: il nome della posizione o un nome del genere o si può scrivere query più complesse, che non raggiungono la complessità di SQL ma consentono un buon livello di dettaglio dei dati di localizzazione.

indipendente dal database

Ghini is not a database management system, so it does not reinvent the wheel. It works storing its data in a SQL database, and it will connect to any database management system which accepts a SQLAlchemy connector. This means any reasonably modern database system and includes MySQL, PostgreSQL, Oracle. It can also work with sqlite, which, for single user purposes is quite sufficient and efficient. If you connect Ghini to a real database system, you can consider making the database part of a LAMP system (Linux-Apache-MySQL-Php) and include your live data on your institution web site.

lingua agnostico

The program was born in English and all its technical and user documentation is first written in that language. Both technical and user documentation use `gettext`, an advanced tool for semi-automatic translation.

The program has been translated and can be used in various other languages, including Spanish (97%), French (82%), Portuguese (71%), to name some Southern American languages, as well as Ukrainian (100%) and Czech (71%).

Translation of documentation goes a bit slower, with only Ukrainian, Spanish and Italian at more than 50%.

indipendente dalla piattaforma

Installazione di Ghini su Windows è un processo semplice e lineare, ci vorranno non più di 10 minuti. Ghini nasce su Linux e installarlo su ubuntu, fedora o debian è di conseguenza ancora più facile. MacOSX essendo basato su unix, è possibile eseguire correttamente la procedura di installazione di Linux su qualsiasi computer Apple recenti, dopo alcuni passaggi di preparazione.

facilmente aggiornabile

Il processo di installazione produrrà un'installazione aggiornabile, dove l'aggiornamento avrà meno di un minuto. A seconda della quantità di feedback che riceviamo, produrremo aggiornamenti ogni pochi giorni o una volta in un po " di tempo.

unit testata

Ghini è continuamente e ampiamente testato unità, qualcosa che rende la regressione di funzionalità quasi impossibile. Ogni aggiornamento è automaticamente qualità selezionata, il servizio di Travis Continuous Integration. Integrazione di TravisCI con la piattaforma github renderà difficile per noi a rilasciare qualcosa che ha un'unità singola in mancanza di prova.

La maggior parte delle modifiche e aggiunte noi fare, venire con alcuni test di unità extra, che definisce il comportamento e farà qualsiasi cambiamento indesiderato facilmente visibile.

customizable/extensible

Ghini è estensibile tramite plugin e può essere personalizzato per soddisfare le esigenze dell'istituzione.

1.2 Missione & Visione

Qui di seguito riportiamo chi siamo, cosa pensiamo del nostro lavoro, cosa potete aspettarvi di questo progetto.

1.2.1 Chi c'è dietro Ghini

Ghini is a small set of programs, meant to let collection managers manage their collection also digitally.

Ghini was born back in 2004 as Bauble, at the Belize Botanical Garden. It was later adapted to the needs of a few more gardens. Brett Adams, the original programmer, made this software a commons, by releasing it under a GPL license.

After years of stagnation Mario Frasca revived the project, and rebranded it as Ghini in honour of Luca Ghini, founder of the first European botanic garden and herbarium. Mario Frasca

started advocating, travelling, distributing, developing, expanding, redefining, documenting it, and it is now Mario Frasca writing this, looking for users, requesting feedback.

Quindi attualmente dietro Ghini c'è molto di più che solo uno sviluppatore, c'è una piccola ma crescente comunità globale di utenti.

Translations are provided by volunteers who mostly stay behind the scenes, translating missing terms or sentences, and disappearing again.

Per rendere le cose più chiare quando si parla di Ghini, è bene indicare se si tratta di Ghini (il software), o Ghini (le persone), a meno che ovviamente non vogliamo indicare entrambe le cose.

1.2.2 Missione

Our goal as Ghini Software is to provide free software, of proven quality, and to let anybody install it if they feel like it. We also aim at facilitating access to functional knowledge, in the form of documentation or by laying the contact among users or between users and software professionals.

All our sources, software and documentation, are open and free, and we welcome and stimulate people to use and to contribute. To facilitate community forming, all our platforms can be consulted without registration. Registration is obviously required if you want to contribute.

Ghini welcomes the formation of groups of users, bundling forces to define and finance further development, and we welcome developers contributing software, from any corner in the world, and we stimulate and help them comply with the high quality requirements, before we accept the contributed code in the software sources.

1.2.3 Visione

La visione serve a indicare la strada da percorrere e proietta un'immagine futura di quello che vogliamo che la nostra organizzazione diventi, in modo realistico e attraente. Essa serve come motivazione perché visualizza la sfida e la direzione delle modifiche necessarie per crescere e prosperare.

- entro il 2020
- segno di riferimento
- comunità
- sviluppo
- integrazione con il portale web
- informazioni geografiche

CAPITOLO 2

Istallare Ghini

2.1 Istallazione

ghini.desktop è un programma multipiattaforma e funziona su macchine unix (Linux e MacOSX) ma anche su Windows.

one-liner for hurried users.

> Linux users just download and run the installation script. You may read the documentation later.
>
> Windows users in a real hurry don't the instructions and use a recent Windows installer. You do not miss any functional feature, but you have less chances to contribute to development.
>
> Mac users are never in a hurry, are they?

Il gruppo di sviluppo di Ghini è piccolo, preferiamo lavorare al migliorare il programma, o a documentarlo, piuttosto che perder tempo con pacchetti di installazione. Invece di tanto pacchetti differenti, uno per piattaforma, offriamo una stessa procedura di installazione per tutte le piattaforme. Questo non solo ci risparmia tempo, ci regala anche un po" di vantaggi, che apprezzeremo nel corso dell'uso del software.

L'installazione è basata sull'esecuzione di uno script.

- Lo script GNU/Linux si occupa di tutto, dalla soluzione delle dipendenze alla installazione per tutti gli utenti nel gruppo `ghini`.
- Lo script per Windows richiede l'installazione previa di un paio di cose.

- Su MacOSX utilizzeremo lo stesso script di GNU/Linux, però non avendo OSX un gestore di pacchetti ne installeremo uno prima di eseguire lo script.

Seguendo le seguenti istruzioni di installazione si ottiene ghini.desktop installato in un ambiente Python virtuale, tutte le dipendenze saranno installate localmente e non entreranno in conflitto con altri programmi Python che possono essere sullo stesso elaboratore.

Le dipendenze che non è possibile installare in un ambiente virtuale Python sono: Python, virtualenv, GTK+, e PyGTK. La loro installazione differisce per piattaforma.

Se dovessi in seguito decidere di rimuovere Ghini, basterà rimuovere l'ambiente virtuale, che è una directory, con tutto il suo contenuto.

2.1.1 Installare su Linux

Open a shell terminal window, and follow the following instructions.

1. Download the *devinstall.sh* script:

   devinstall.sh

2. Invoke the script from a terminal window, starting at the directory where you downloaded it, like this:

   ```
   bash ./devinstall.sh
   ```

   The script will produce quite some output, which you can safely ignore.

   ---

   global installation

   When almost ready, the installation script will ask you for your password. This lets it create a `ghini` user group, initialise it to just yourself, make the just created `ghini` script available to the whole `ghini` user group.

   If feeling paranoid, you can safely not give your password and interrupt the script there.

   Possibly the main advantage of a global installation is being able to find Ghini in the application menus of your graphic environment.

   ---

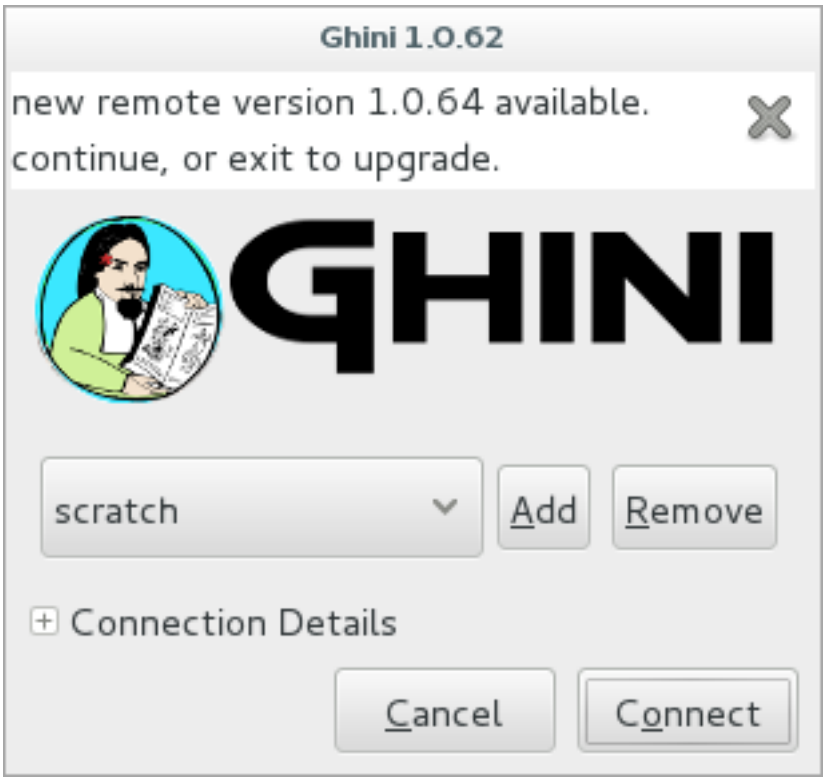
3. You can now start ghini by invoking the `ghini` script:

   ```
   ghini
   ```

   1. You use the same `ghini` script to update ghini to the latest released production patch:

      ```
      ~/bin/ghini -u
      ```

      This is what you would do when ghini shows you something like this:



2. Users of the global installation will also type `ghini` to invoke the program, but they will get to a different script, located in `/usr/local/bin`. This globally available `ghini` script cannot be used to update a ghini installation.

3. Again the same `ghini` script lets you install the optional database connectors: option `-p` is for PostgreSQL, option `-m` is for MySQL/MariaDB, but you can also install both at the same time:

```
~/bin/ghini -pm
```

   Please beware: you might need solve dependencies. How to do so, depends on which GNU/Linux flavour you are using. Check with your distribution documentation.

4. You can also use the `ghini` script to switch to a different production line. At the moment `1.0` is the stable one, but you can select `1.1` if you want to help us with its development:

```
~/bin/ghini -s 1.1
```

nota per il principiante

Per eseguire uno script, in primo luogo assicurarsi che avete segnato il nome della directory in cui avete scaricato lo script, quindi si apre una finestra di terminale e in quella finestra si digita "bash" seguita da uno spazio e il nome completo dello script incluso nome directory e colpire il tasto di invio.

nota tecnica

You can study the script to see what steps if runs for you.

In short it will install dependencies which can't be satisfied in a virtual environment, then it will create a virtual environment named `ghide`, use git to download the sources to a directory named `~/Local/github/Ghini/ghini.desktop`, and connect this git checkout to the `ghini-1.0` branch (this you can consider a production line), it then builds ghini, downloading all remaining dependencies in the virtual environment, and finally it creates the `ghini` startup script.

If you have `sudo` permissions, it will be placed in `/usr/local/bin`, otherwise in your `~/bin` folder.

Prossimo...

Connettersi ad una base dati.

2.1.2 L'installazione su MacOSX

Being macOS a unix environment, most things will work the same as on GNU/Linux (sort of).

Prima di tutto, avete bisogno di cose che sono parte integrante di un ambiente unix, ma che mancano in un mac disponibile immediatamente:

1. strumenti per gli sviluppatori: xcode. controllare la pagina di wikipedia per la versione supportata sul vostro mac.
2. Gestione pacchetti: homebrew (tigerbrew per le vecchie versioni OSX).

Installation on older macOS.

Every time we tested, we could only solve all dependencies on the two or three most recent macOS versions. In April 2015 this excluded macOS 10.6 and older. In September 2017 this excluded macOS 10.8 and older. We never had a problem with the lastest macOS.

The problem lies with homebrew and some of the packages we rely on. The message you have to fear looks like this:

```
Do not report this issue to Homebrew/brew or Homebrew/core!
```

The only solution I can offer is: please update your system.

On the bright side, if at any time in the past you did install ghini.desktop on your older and now unsupported macOS, you will always be able to update ghini.desktop to the latest version.

With the above installed, open a terminal window and run:

```
brew doctor
```

Assicurarsi di che comprendere i problemi segnala e correggerli. PyGTK dovrà xquartz e brew non risolverà automaticamente la dipendenza. installare xquartz utilizzando brew o il modo che preferisci:

```
brew install Caskroom/cask/xquartz
```

quindi installare le dipendenze rimanenti:

```
brew install git
brew install pygtk  # takes time and installs all dependencies
```

seguire tutte le istruzioni su come attivare ciò che avete installato.

In particular, make sure you read and understand all reports starting with `If you need to have this software`.

You will need at least the following four lines in your `~/.bash_profile`:

```
export LC_ALL=en_US.UTF-8
export LANG=en_US.UTF-8
export PATH="/usr/local/opt/gettext/bin:$PATH"
export PATH="/usr/local/opt/python/libexec/bin:$PATH"
```

Activate the profile by sourcing it:

```
. ~/.bash_profile
```

Prima di poter correre " devinstall.sh " come su GNU/Linux, abbiamo ancora bisogno di installare un paio di pacchetti python, a livello globale. A tale scopo:

```
sudo -H pip install virtualenv lxml
```

Il resto è proprio come su una normale macchina unix. Leggere le istruzioni di GNU/Linux sopra, li seguono, godere.

As an optional aesthetical step, consider packaging your `~/bin/ghini` script in a platypus application bundle. The `images` directory contains a 128×128 icon.

Prossimo...

Connettersi ad una base dati.

2.1.3 installazione su Windows

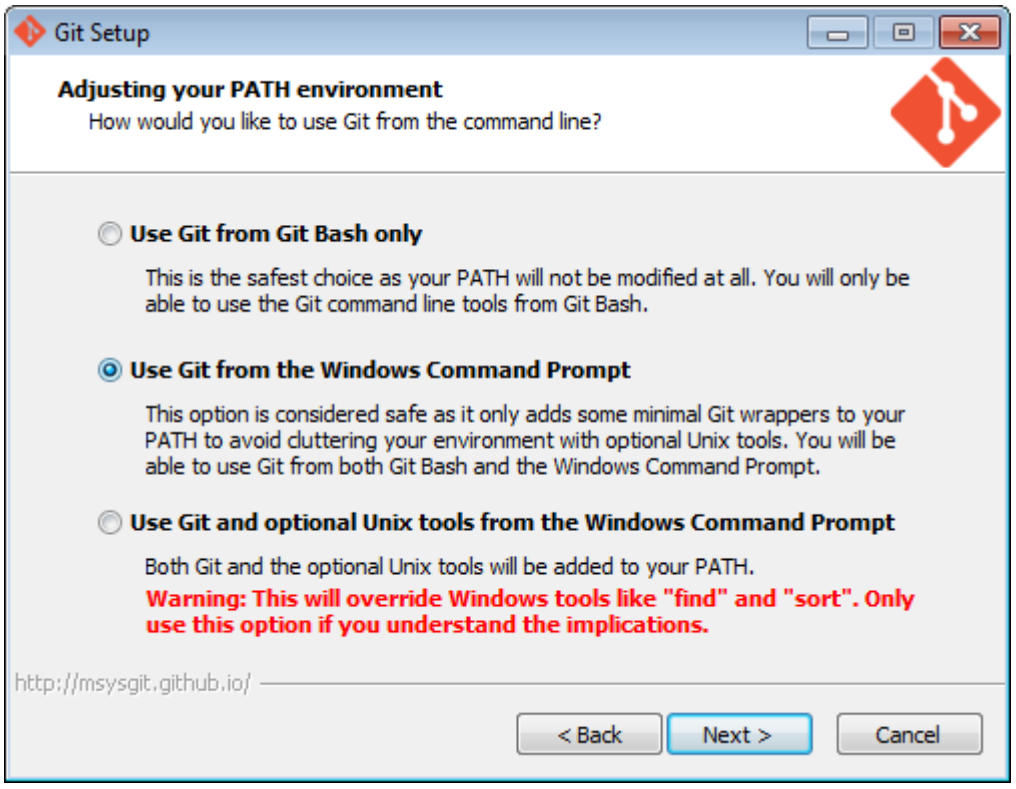
La procedura descritta qui le istruzioni su come installare Git, Gtk, Python, e il pitone database connettori. Con questo ambiente impostato correttamente, la procedura di installazione di Ghini viene eseguito come su GNU/Linux. Le fasi conclusive sono nuovamente Windows specifico.

Nota: Ghini has been tested with and is known to work on W-XP, W-7 up to W-10. Although it should work fine on other versions Windows it has not been thoroughly tested.

La procedura di installazione su Windows:

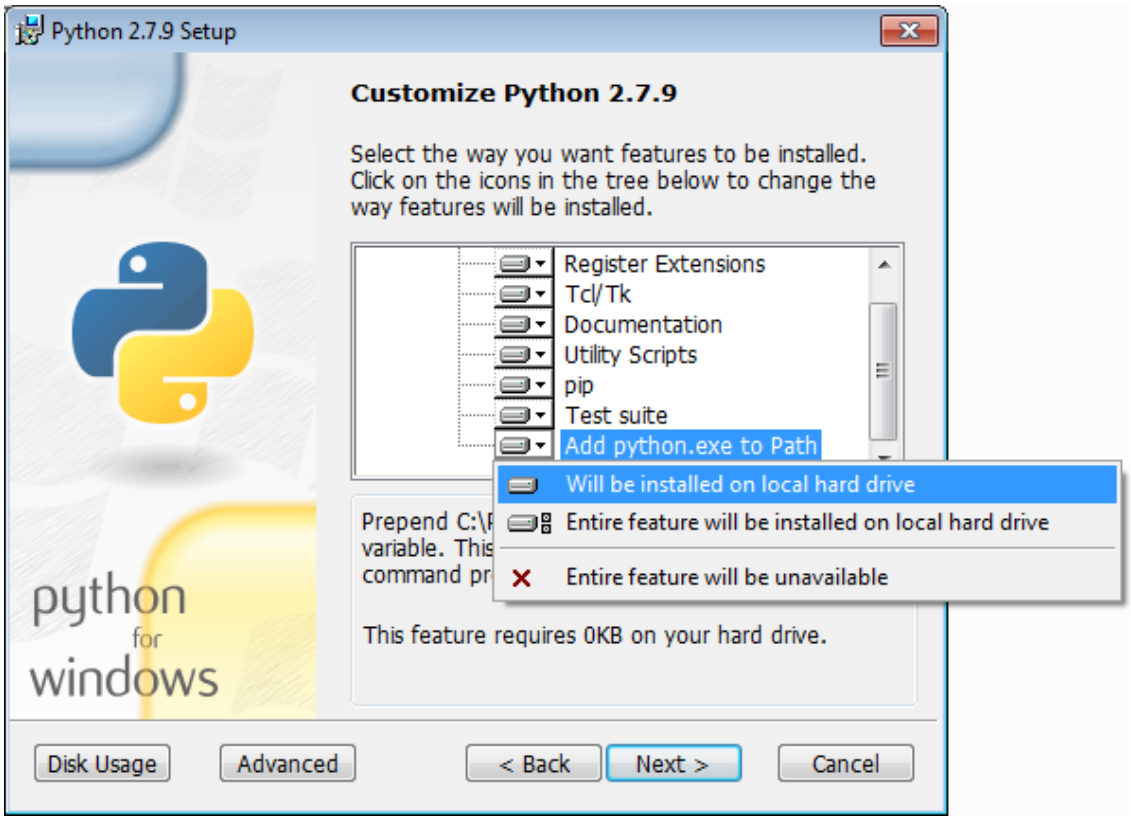
1. download and install `git` (comes with a unix-like `sh` and includes `vi`). Grab it from the Git download area.

   tutte le opzioni predefinite sono corrette, tranne che abbiamo bisogno di git per essere eseguibile dal prompt dei comandi:



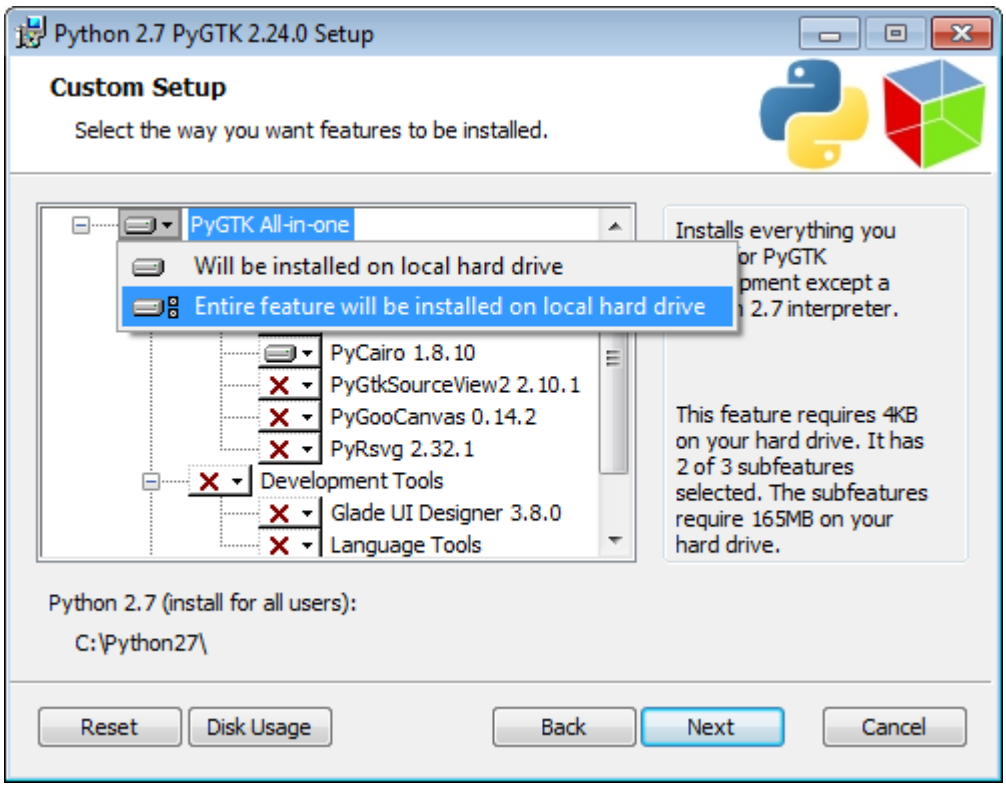
2. download and install Python 2.x (32bit). Grab it from the Python official site.

   When installing Python, do put Python in the PATH:



3. download `pygtk` from the official source. (this requires 32bit python). be sure you download the «all in one» version.

   Make a complete install, selecting everything:



4. (Possibly necessary, maybe superfluous) install lxml, you can grab this from the pypi

archives

Ricordate che avete bisogno della versione a 32 bit, per Python 2.7.

5. (definitely optional) download and install a database connector other than `sqlite3`.

   If you plan using PostgreSQL, the best Windows binary library for Python is psycopg and is Made in Italy.

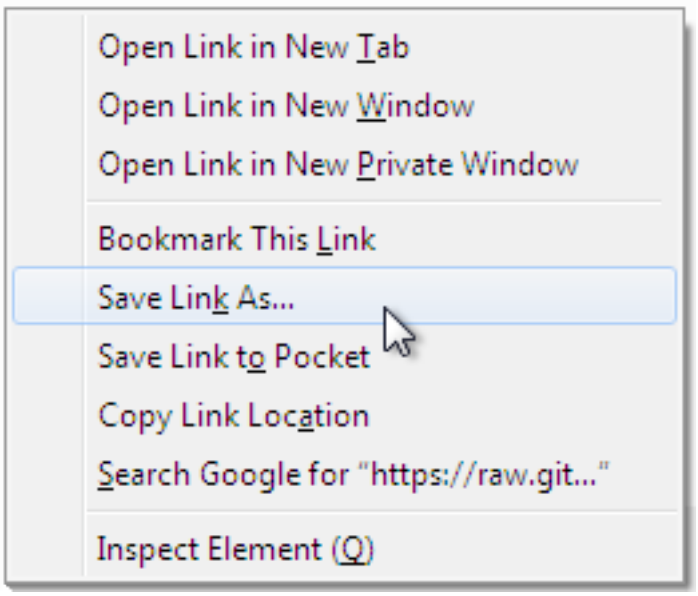
6. **RIAVVIA**

   Ehi, questo è per Windows, è necessario riavviare per rendere effettive le modifiche!

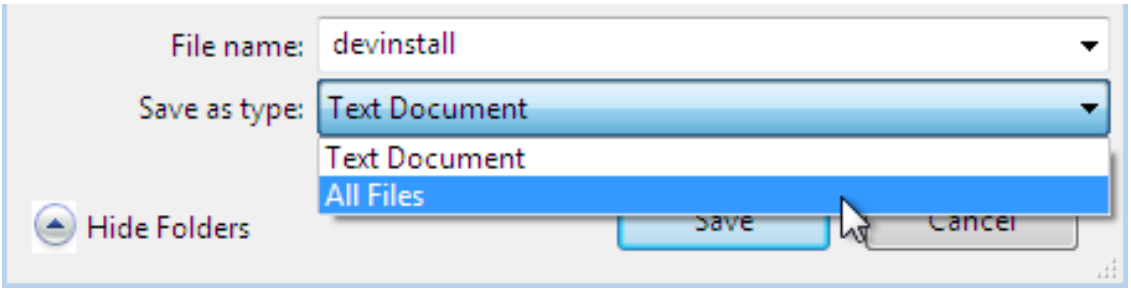
7. We're done with the dependecies, now we can download and run the batch file:

   devinstall.bat

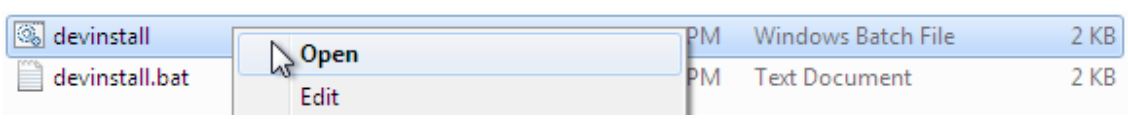
   Please don't just follow the above link. Instead: right click, save link as...



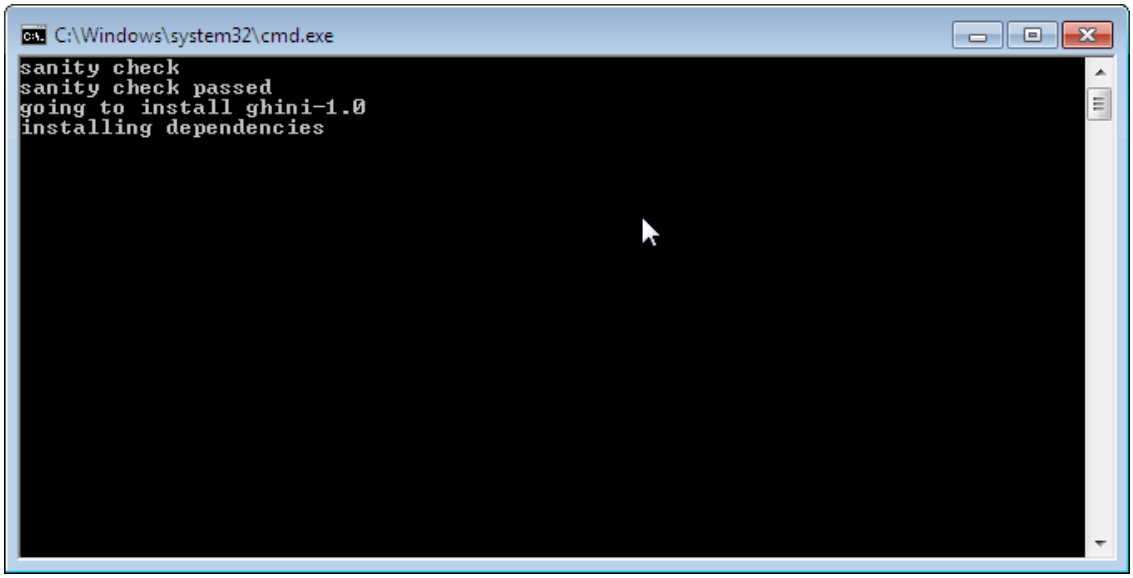
Also make sure you don't let Windows convert the script to a text document.



Now **Open** the script to run it. Please note: in the below image, we have saved the file twice, once letting Windows convert it to a text document, and again as a Windows Batch File. Opening the batch file will run the script. Opening the text document will show you the code of the batch file, which isn't going to lead us anywhere.



If you installed everything as described here, the first thing you should see when you start the installation script is a window like this, and your computer will be busy during a couple of minutes, showing you what it is doing.



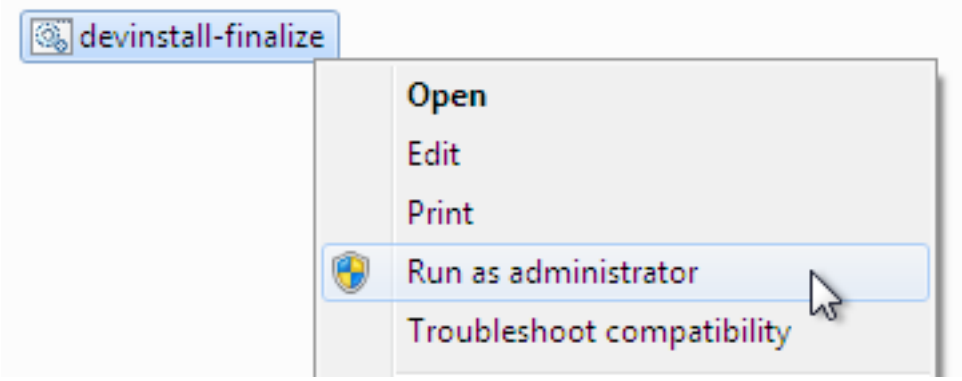
Running `devinstall.bat` will pull the `ghini.desktop` repository from github to your home directory, under `Local\github\Ghini`, checkout the `ghini-1.0` production line, create a virtual environment and install ghini into it.

You can also run `devinstall.bat` passing it as argument the numerical part of the production line you want to follow.

This is the last installation step that depends, heavily, on a working internet connection.

The operation can take several minutes to complete, depending on the speed of your internet connection.

8. l'ultimo passo di installazione crea il gruppo Ghini ed i collegamenti nel Menu Start di Windows, per tutti gli utenti. A tale scopo, è necessario eseguire uno script con diritti amministrativi. Lo script si chiama " devinstall-finalize.bat ", è proprio nella tua cartella HOME ed è stato creato nel passaggio precedente.



Right-click on it, select run as administrator, confirm you want it to make changes to your computer. These changes are in the Start Menu only: create the Ghini group, place the Ghini shortcut.

9. download the batch file, it will help you staying up-to-date:

   ghini-update.bat

   If you are on a recent Ghini installation, each time you start the program, Ghini will check on the development site and alert you of any newer ghini release within your chosen production line.

   Any time you want to update your installation, just run the `ghini-update.bat` script, it will hardly take one minute.

   How to save a batch file, and how to run it: check the the quite detailed instructions given for `devinstall.bat`.

If you need to generate PDF reports, you can use the XLS based report generator and you will need to download and install Apache FOP. After extracting the FOP archive you will need to include the directory you extracted to in your PATH.

If you choose for PostScript reports, you can use the Mako based report generator and there are no further dependencies.

Prossimo...

Connettersi ad una base dati.

2.1.4 Installing on Android

`ghini.desktop` is a desktop program, obviously you don't install it on a handheld device, but we do offer the option, for your Android phone or tablet, to install `ghini.pocket`.

`ghini.pocket` is a small data viewer, it comes handy if you want to have a quick idea of a plant species, its source, and date it entered the garden, just by scanning a plant label.

Installation is as easy as it can be: just look for it in Google Play, and install it.

Export the data from `ghini.desktop` to pocket format, copy it to your device, enjoy.

CAPITOLO 3

User's Guide

3.1 Configurazione iniziale

Dopo una corretta installazione, organizzazioni più complesse vorranno configurare il loro database e configurare Ghini secondo la loro configurazione del database. Questa pagina si concentra su questo compito. Se non hai idea di cosa si tratti, meglio continuare la lettura con la parte relativa a SQLite.

3.1.1 E se fosse SQLite?

Questa è la prima volta che si utilizza Ghini, sono stai andando a lavorare in un ambiente autonomo, avete non la più pallida idea come gestire un sistema di gestione di database? Se avete risposto sì a uno qualsiasi dei precedenti, è probabilmente meglio restare con SQLite, il database facile, veloce, senza doveri di amministrazione, tutto basato su file, il database più diffuso al mondo.

Con SQLite non c'è bisogno di preparare altro e si può proseguire con *connecting*.

D'altra parte, se si desidera collegare più di una workstation ghini allo stesso database o se si desidera rendere disponibili i dati per altri client, come potrebbe essere un server web in una configurazione LAMP, è consigliabile mantenere il database in un database management system come PostgreSQL o MySQL/MariaDB, entrambi supportati da Ghini.

When connecting to a database server as one of the above, you have to manually do the following: Create at least one user; Create your database; Give at least one user full permissions on your database; If you plan having more database users: Give one of your users the `CREATEROLE` privilege; Consider the user with the `CREATEROLE` privilege as a super-user, not meant to handle data directly; Keep your super-user credentials in a very safe place.

When this is done, Ghini will be able to proceed, creating the tables and importing the default data set. The process is database-dependent and it falls beyond the scope of this manual.

Se già vi tremano le ginocchia, non c'è bisogno di preoccuparsi, basta restare con SQLite, senza perdere in funzionalità né prestazioni.

Some more hints if you need PostgreSQL

Start simple, don't do all at the same time. Review the online manual, or download and study the offline version.

As said above, create a database, a user, make this user the owner of the database, decide whether you're going to need multiple users, and preferably reserve a user for database and normal user creation. This super-user should be your only user with `CREATEROLE` privilege.

All normal users will need all privileges on all tables and sequences, something you can do from the *Tools→Users* menu. If you have any difficulty, please open an issue about it.

Connect using the `psql` interactive terminal. Create a `~/.pgpass` file (read more about it in the manual), tweak your `pg_hba.conf` and `postgresql.conf` files, until you can connect using the command:

```
psql <mydb> --username <myuser> --no-password --host
↪<mydbhost>
```

With the above setup, connecting from ghini will be an obvious task.

3.1.2 Connettersi ad una base dati

All'avviare Ghini, la prima cosa che appare è la finestra per la scelta della connessione.

Abbastanza naturalmente, la prima volta che si avvia Ghini non ci saranno connessioni a disposizione. Ghini mostra una finestra di avviso.

Questo avviso vi mostrerà alla prima attivazione e anche in futuro se il tuo elenco di connessioni diventa vuota. Come l'avviso stesso dice: fare clic su **aggiungere** per creare la prima connessione.

Semplicemente: inserire un nome per la connessione, una cose semplice e significativa che si associ naturalmente alla collezione che si vuol inserire (che so, «alberi monumentali»), e scegliere **OK**. Ghini torna allo schermo precedente, ma il nome della connessione sarà selezionato e la sezione Dettagli della Connessione sarà visibile.

specificare i dettagli della connessione

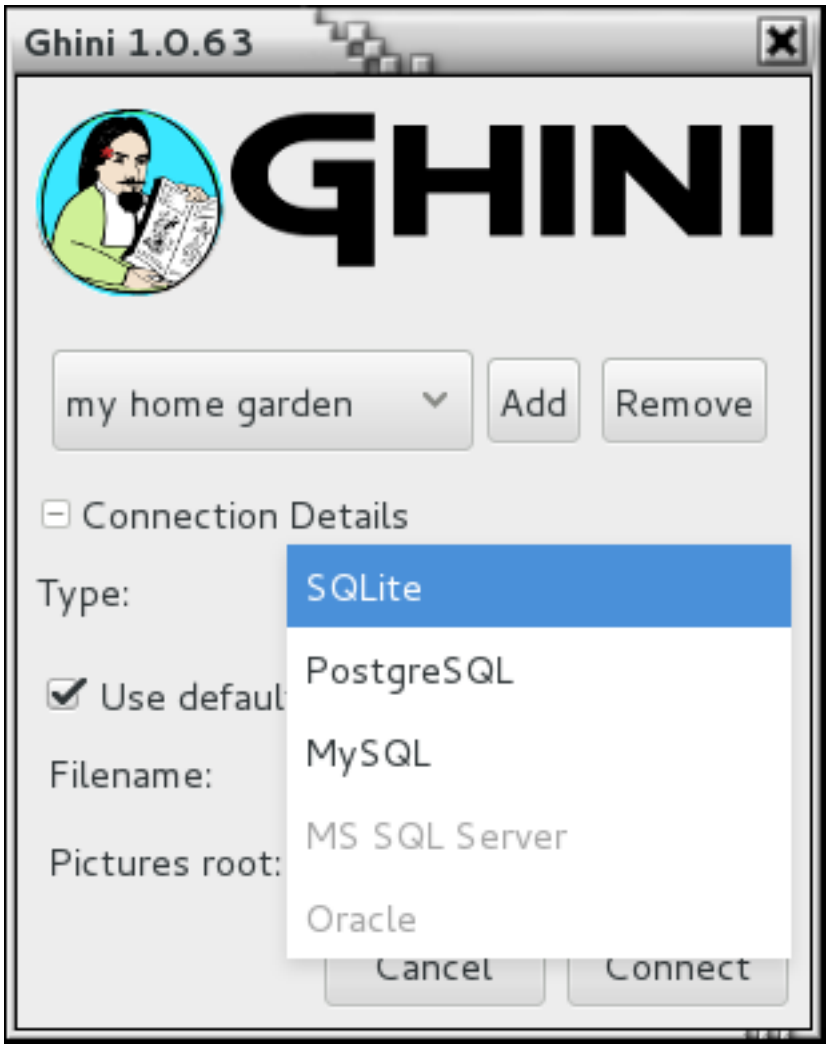
Se non si conosce cosa fare qui, Ghini vi aiuterà a stare al sicuro. Attivare il **utilizzare la casella di controllo posizioni** di predefinito e creare il tuo primo collegamento facendo clic su **Connect**.

Si può tranquillamente ignorare il resto di questa sezione per il momento e continua a leggere la sezione seguente.

mettere a punto i dettagli di connessione

Per impostazione predefinita Ghini utilizza SQLite database basato su file. Durante il processo di installazione si aveva la scelta (e avete ancora dopo l'installazione), per aggiungere connettori di database diverso da quello predefinito SQLite.

In questo esempio, Ghini può connettersi a SQLite, PostgreSQL e MySQL, ma nessun connettore è disponibile per MS SQL Server o Oracle.



Se si utilizza SQLite, è davvero necessario specificare solo il nome di connessione. Se permetti che Ghini utilizzi il nome file predefinito, Ghini crea un file di database con lo stesso nome come la connessione ed estensione `.db` e una cartella di immagini ancora con lo stesso nome ma senza estensione, entrambe in `~/.bauble` su Linux/MacOSX o in `AppData\Roaming\Bauble` su Windows.

Ancora con SQLite, si potrebbe avere ricevuto o scaricato un database bauble, e si desidera connettersi ad esso. In questo caso non lasciate Ghini utilizzare il nome file predefinito, ma navighi nel vostro computer nel percorso dove è stato salvato il Ghini SQLite database file.

Se si utilizza un connettore diverso database, finestra di dialogo avrà un aspetto diverso e vi offrirà l'opzione di mettere a punto tutti i parametri necessari per connettersi al database di vostra scelta.

If you are connecting to an existing database you can continue to *Inserire e modificare dati* and subsequently searching-in-ghini, otherwise read on to the following section on initializing a database for Ghini.

Se si prevede di associare immagini alle piante, specificare anche il * cartella radice * immagini. Il significato di questo è spiegato in dettaglio a: ref: "pianta-immagini" in: ref: "editing-e-inserimento-dati".

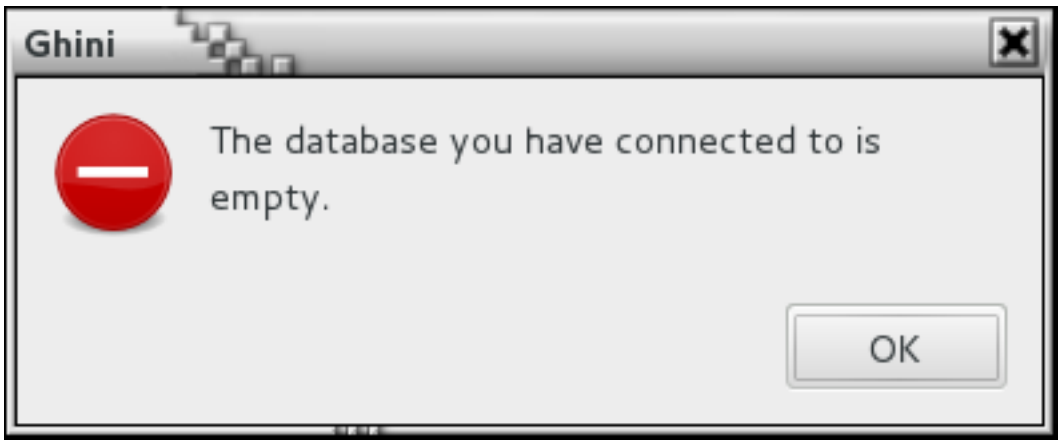
A sample SQLite database

Indeed we have a sample database, from our pilot garden «El Cuchubo», in Mompox, Colombia. We have a zipped sample database for ghini-1.0.

Download and unzip it to the location of your choice, then start Ghini, create a connection named possibly `cuchubo`, or `sample`, and edit the Connection Details. Keep the connection type at the default SQLite, but instead of using the default locations, make sure that Filename points to your unpacked `cuchubo.db` file.

3.1.3 Inizializzare la base dati

La prima volta che si apre una connessione ad una base dati che Ghini non ha mai utilizzato prima, Ghini mostrerà l'avviso:



seguito a ruota da una domanda:

Fare attenzione quando si specificano manualmente i parametri di connessione: i valori immessi possono riferirsi a un database esistente, non destinato all'uso con Ghini. Permettendo a Ghini di inizializzare il database, questo sarà svuotato e tutto il suo contenuto verrà perso.

Se sei sicuro di voler creare un database con questi parametri seleziona «Sì». Ghini inizierà quindi col creare le tabelle del database e procederà con l'importazione di dati predefiniti. Questo può richiedere un minuto o due: prego pazientare mentre tutti i dati predefiniti vengono importati.

Una volta creata, configurata, inizializzata la base dati, siamo pronti per iniziare a editing-and-inserting-data'ed in seguito :ref:'searching-in-bauble.

3.2 Ricerca in Ghini

Ricerca permette di visualizzare, esplorare e creare rapporti dai dati. È possibile eseguire ricerche inserendo le query nella voce di ricerca principale o utilizzando il generatore di Query per creare le query per voi. I risultati delle ricerche di Ghini sono elencati nella finestra principale.

3.2.1 Strategie di ricerca

Ghini offre quattro strategie di ricerca distinte:

- per valore — in tutti i domini;
- per espressione — in alcuni campi impliciti in un dominio esplicito;
- per query — in un dominio;
- di nome binomiale — cerca solo nel dominio di `Species`.

Tutte le strategie di ricerca —ad eccezione di 'ricerca nome binomiale'— non fanno distinzione tra maiuscole e minuscole.

Ricerca per valore

Ricerca per valore è il modo più semplice per la ricerca. Immettere una o più stringhe e vedere che cosa corrisponde. Il risultato include oggetti di qualsiasi tipo (dominio), dove uno o più dei relativi campi contengono uno o più delle stringhe di ricerca.

Non si specifica il dominio di ricerca, tutti sono inclusi, né è necessario indicare quali campi si desidera far corrispondere, questo è implicito nel dominio di ricerca.

Nella tabella seguente consente di comprendere i risultati e vi guida nel formulare le ricerche.

Panoramica sui domini di ricerca		
nome ed abbreviazioni	campo	tipo di risultato
family, fam	epithet (family)	`Family`
genus, gen	epithet (genus)	`Genus`
species, sp	epithet (sp) ×	`Species`
vernacular, common, vern	name	`Species`
geography, geo	name	`Geography`
accession, acc	code	`Accession`
planting, plant	code ×	`Plant`
location, loc	code, name	`Location`
contact, person, org, source	name	`Contact`
collection, col, coll	locale	`Collection`
tag, tags	name	`Tag`

Esempi di ricerca per valore sarebbe: Maxillaria, Acanth, 2008.1234, 2003.2.1, indica.

A meno di non includere fra virgolette, gli spazi separano le stringhe di ricerca. Per esempio se si cerca `Block 10`, Ghini cerca le due stringhe `blocco` e `10` e restituisce tutti i risultati che corrispondono a una di queste due voci. Se si desidera cercare Block 10 come una stringa intera basterà usare le virgolette, così: `"Block 10"`.

× Chiavi primarie composte

Un epiteto di **specie** significa poco senza il genere corrispondente, allo stesso modo un codice di **planting** è univoco solo nell'ambito dell'accessione a cui appartiene. Nella terminologia della teoria dei database, epiteto e codice non sono sufficienti per formare un **chiave primaria** per rispettivamente specie e le planting. Questi domini necessitano di una **chiave primaria composta**.

Ricerca per valore consente di cercare **planting** dal loro codice planting completo, che include il codice di accessione. Presi insieme, il codice di accessione e la piantatura del codice forniscono uns **chiave primaria composito** per gli planting. Per **specie**, abbiamo introdotto la ricerca binomiale, descritta di seguito.

Ricerca per espressione

Ricerca per espressione vi dà un po" più controllo su ciò che stai cercando. Permette di restringere la ricerca ad un dominio specifico, il software definisce i campi per la ricerca all'interno del dominio indicato.

Un'espressione è costruita come `<domain> <operator> <value>`. Ad esempio la ricerca: `gen = Maxillaria` restituisce tutti i generi che corrispondono al nome Maxillaria. In questo caso il dominio è `gen`, l'operatore è = e il valore è `Maxillaria`.

La tabella *Panoramica sui domini di ricerca* poco sopra ti indica i nomi dei domini di ricerca, e, per ogni dominio di ricerca, quali campi vengono inclusi implicitamente nelle ricerche.

La stringa di ricerca `loc like blocco%` restituisce tutte le collocazioni il cui nome o codice inizia con «blocco». In questo caso il dominio è `loc` (un'abbreviazione per `location`), l'operatore è `like` (questo viene dall'SQL ed aggiunge un fattore di approssimazione nella ricerca), il valore è `blocco%`, i campi di ricerca sono implicitamente `nome` e `codice`. Il segno di percentuale viene utilizzato come jolly, pertanto se si cerca `max%` ricercherà tutti i valori che iniziano con «max». Se si cerca `%10` ricerca per tutti i valori che terminano in `10`. La stringa `%ck%10` indica la ricerca di tutto ciò che contenga `ck` e termini in `10`.

Quando una query richiede molto tempo per completare

Si dà una query, ci vuole davvero tempo per calcolare il risultato e questo contiene una quantità irragionevolmente di voci. Questo accade quando si intende utilizzare una strategia, ma le stringhe inserite non formano un'espressione valida. In questo caso Ghini ricorre al caso *ricerca per valore*. Ad esempio, la stringa di ricerca `gen lik maxillaria` cercherà le stringhe `gen`, `lik` e `maxillaria`, restituendo tutto ciò che soddisfi almeno uno dei tre criteri.

Ricerca binomiale

È inoltre possibile eseguire una ricerca nel database, se si conosce la specie, semplicemente inserendo alcune lettere iniziali di epiteti di genere e la specie nel motore di ricerca, in maiuscolo correttamente, vale a dire: **genere epiteto** con una lettere maiuscole iniziali, **specie epiteto** tutte le lettere minuscole.

In questo modo è possibile eseguire la ricerca `So ha`.

Queste sarebbero le iniziali per il Solanum hayesii, o Solanum havanense.

Ricerca nome binomiale viene a compensare la limitata utilità della succitata ricerca per espressione quando si tenta di cercare una specie.

È la lettera maiuscola corretta **Xxxx xxxx** che informa il software della vostra intenzione di eseguire una ricerca binomiale. Seconda ipotesi del software sarà una ricerca per valore, che possibilmente si tradurrà in molti più risultati del previsto.

La richiesta affatto simile `so ha` restituisce, in una nuova installazione, oltre 3000 oggetti, a partire da famiglia «Acalyp(**ha**)ceae», per finire con la geografia «Western (**So**)uth America».

Ricerca per Query

Le query consentono il massimo controllo su ricerca. Con le query è possibile cercare attraverso relazioni, colonne specifiche, combinare i criteri di ricerca utilizzando gli operatori booleani come `and`, `or`, `not` (e loro rispettiva abbreviazioni `&&`, `||`, `!`).

Se volete maggiori informazioni, o volontariare per documentare questo più a fondo, si prega di contattare gli autori. Nel frattempo si può iniziare a familiarizzare con la struttura di base del database di Ghini.

Fig. 1: **struttura di database** di Ghini

Alcuni esempi:

- piante della famiglia delle Fabaceae in posizione Blocco 10:

```
plant WHERE accession.species.genus.family.epithet=Fabaceae AND␣
↪location.description="Block 10"
```

- posizioni che non contengono piante:

```
location WHERE plants = Empty
```

- accessioni associate ad una specie dal nome binomiale noto (es.: Mangifera indica):

```
accession WHERE species.genus.epithet=Mangifera AND species.
↪epithet=indica
```

- accessioni propagate nell'anno 2016:

```
accession WHERE plants.propagations._created BETWEEN␣
↪|datetime|2016,1,1| AND |datetime|2017,1,1|
```

- accessioni modificate negli ultimi tre giorni:

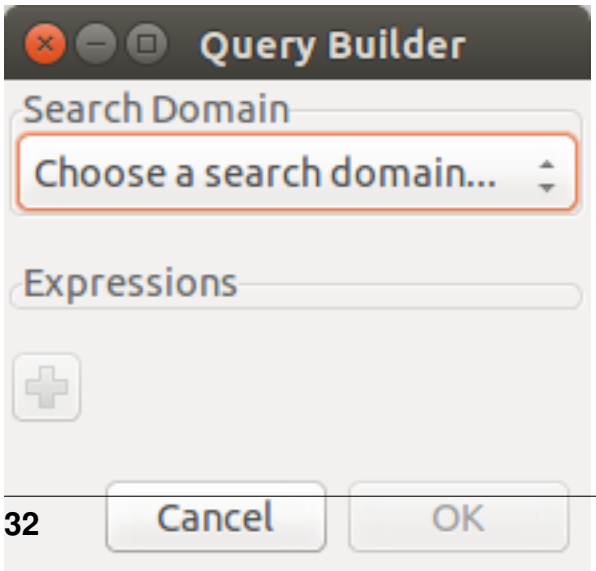
```
accession WHERE _last_updated>|datetime|-3|
```

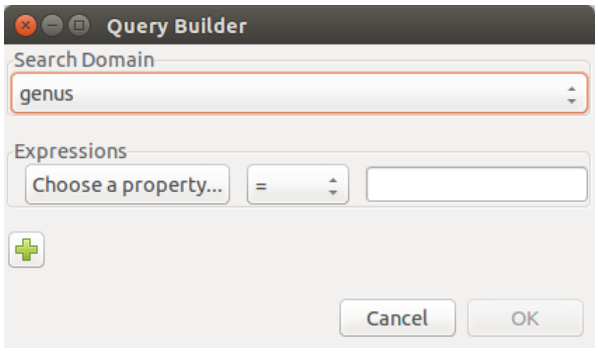
Ricerca per query richiede qualche conoscenza della sintassi ed un'idea della estesa struttura del database Ghini. Entrambe si acquisiscono con la pratica e magari con l'aiuto del Generatore di Query.

3.2.2 Generatore di Query

Ghini offre un generatore di Query, una interfaccia grafica che aiuta a costruire query di ricerca anche complesse. Per aprire il Generatore di Query fare clic sull'icona  a sinistra della voce di ricerca o selezionare *Strumenti→Query Builder* dal menu.

Una finestra apparirà, che vi guiderà attraverso tutti i passaggi necessari per creare una query corretta che sarà compresa dalla strategia di Ricerca per Query di Ghini.





Prima di tutto si indica il dominio di ricerca, questo vi permetterà il generatore di Query

completa la sua interfaccia grafica, quindi si aggiunge il numero di clausole logico necessario, collegandoli con operatore binario `and` o `or`.

Ogni clausola è formata da tre parti: una proprietà che può essere raggiunto dal dominio di ricerca iniziale, un operatore di confronto che si seleziona dall'elenco a discesa, un valore che è possibile digitare o selezionare dall'elenco di valori validi per il campo.

Aggiungere altrettante proprietà di ricerca quante necessarie, facendo clic sul segno più. Selezionare `and` / `or` al lato dal nome della proprietà, per scegliere come combinare le clausole nella query.

Quando si è finito di costruire la query fare clic su OK per eseguire la ricerca.

A questo punto il generatore di Query scrive la query nella voce Cerca e lo esegue. Adesso si può editare la stringa come fosse stata inserita manualmente. Si noti come i valori alla sinistra vengono interpretati dal generatore di query e racchiusi tra virgolette singole se riconosciuto come stringhe, o lasciati senza virgolette se sono numeri o i due riservati parole `None` e `Empty`. Si può modificare la query e inserire virgolette se necessario, ad esempio se avete bisogno di cercare letteralmente la stringa `Empty`.

`None` è il valore di un campo vuoto. Non è la stessa che la stringa di lunghezza zero `''` né il numero `0` né il booleano `False` né l'insieme `Empty`: `None` indica che al campo non è associato alcun valore.

`Empty` is the empty set. Being it a set, it can be matched against sets (eg: plants of an accession, or accessions of a species), not against elements (eg: quantity of a plant or description of a location). However, the Query Builder does not let you choose a left hand side value stopping at a set, it expects you to select a field. Choose just any field: at the moment of producing the query, when the Query Builder meets a clause with right hand side value the literal string `Empty`, it will drop the field name and let you compare the set on the left with `Empty` on the right.

We have no literals `False` and `True`. These are typed values, and the Query Builder does not know how to produce them. Instead of `False` type `0`, and instead of `True` type `1`.

3.2.3 Grammatica delle Query

Per coloro che non temono un po" di precisione formale, il codice BNF seguente vi dà un'idea piuttosto precisa della grammatica implementata dalla strategia di Ricerca per Query. Alcune categorie grammaticali sono informalmente definite; alcune mancano e sono lasciate alla vostra fertile immaginazione; i valori letterali sono inclusi tra virgolette singole; la grammatica non fa differenza tra maiuscole e minuscole, se non diversamente indicato:

```
query ::= domain 'WHERE' expression

domain ::= #( one of our search domains )
expression ::= signed_clause
             | signed_clause 'AND' expression
             | signed_clause 'OR' expression
             ;
signed_clause ::= clause
                | 'NOT' clause  #( not available in Query Builder)
                ;
clause ::= field_name binop value  #( available in Query Builder)
         | field_name set_binop value_list
         | aggregated binop value
         | field_name 'BETWEEN' value 'AND' value
         | '(' expression ')'
         ;
field_name ::= #( path to reach a database field or connected table␣
↪)
aggregated ::= aggregating_func '(' field_name ')'
aggregating_func ::= 'SUM'
                   | 'MIN'
                   | 'MAX'
                   | 'COUNT'
                   ;
value ::= typed_value
        | numeric_value
        | none_token
        | empty_token
        | string_value
        ;
typed_value ::= '|' type_name '|' value_list '|'
numeric_value ::== #( just a number )
none_token ::= 'None'    #( case sensitive )
empty_token ::= 'Empty'  #( case sensitive )
string_value = quoted_string | unquoted_string

type_name ::= 'datetime' | 'bool' ;  #( only ones for the time␣
↪being )
quoted_string ::= '"' unquoted_string '"'
unquoted_string ::=  #( alphanumeric and more )

value_list ::= value ',' value_list
             | value
             ;
binop ::= '='
        | '=='
        | '!='
        | '<>'
        | '<'
        | '<='
```

(continues on next page)

(continua dalla pagina precedente)

```
        | '>'
        | '>='
        | 'LIKE'
        | 'CONTAINS'
        ;
set_binop ::= 'IN'
```

Si prega di essere consapevoli del fatto che il linguaggio di Query di Ghini è più complesso di quello descritto dal Generatore di Query: le query costruibili con il Generatore di Query costituiscono un sottoinsieme proprio delle query riconosciute dal software:

```
query ::= domain 'WHERE' expression

domain ::= #( one of our search domains )
expression ::= clause
             | clause 'AND' expression
             | clause 'OR' expression
             ;
clause ::= field_name binop value
         ;
field_name ::= #( path to reach a database field or connected table␣
↪)
value ::= numeric_value
        | string_value
        ;
numeric_value ::== #( just a number )
string_value = quoted_string | unquoted_string ;

quoted_string ::= '"' unquoted_string '"'
unquoted_string ::=  #( alphanumeric and more )

binop ::= '='
        | '=='
        | '!='
        | '<>'
        | '<'
        | '<='
        | '>'
        | '>='
        | 'LIKE'
        | 'CONTAINS'
        ;
```

3.3 Inserire e modificare dati

Fondamentalmente, la maniera per aggiungere o modificare informazione con Ghini, è utilizzando gli editori. Per ciascun tipo di informazione, Ghini ha un editore, così per esempio c'è

un editore per le famiglie, per i generi, per le accessioni, eccetera.

Per creare un nuovo elemento nella base dati, attivare il menu *Insert*, e scegliere il tipo di oggetto da inserire. Questo apre un nuovo editore, vuoto, per il tipo scelto.

Per modificare un registro già presente nella base dati e visualizzato nella finestra principale di Ghini, fare click destro sulla sua riga e scegliere *Edit* dal menu che emerge. Questo apre un editore relativo al registro selezionato, modificando i dati nell'editore è possibile aggiornare la base dati.

La maggior parte dei tipi di dati hanno registri che ne dipendono.

3.3.1 Note

Quasi tutti gli editori in Ghini hanno una vista *Note*, che dovrebbe funzionare ugualmente per qualsivoglia editore attivato.

Un indirizzo web inserito a sé stante in una nota, viene trattato in modo speciale da Ghini: quando l'oggetto è selezionato nel pannello risultati, l'indirizzo web verrà mostrato nel contenitore *Links».

È possibile sfogliare le note per un elemento nel database utilizzando la casella note nella parte inferiore dello schermo. La casella note è desensibilizzata se l'elemento selezionato non dispone di eventuali note.

3.3.2 Famiglia

L'editore per le famiglie permette aggiungere o cambiare una famiglia botanica.

Il campo *famiglia* consente di modificare l'epiteto della famiglia. Il campo *famiglia* è obbligatorio.

Il * campo di qualificatore consente di modificare il qualificatore di famiglia. Il valore può essere * sensu lato *, * sensu stricto *, o niente.

Synonyms allow you to add other families that are synonyms with the family you are currently editing. To add a new synonyms type in a family name in the entry. You must select a family name from the list of completions. Once you have selected a family name that you want to add as a synonym click on the Add button next to the synonym list and the software adds the selected synonym to the list. To remove a synonym, select the synonym from the list and click on the Remove button.

Per annullare le modifiche, fare click sul pulsante *Cancel/Annulla*.

Per memorizzare le modifiche effettuate, fare click su *OK*.

Per memorizzare le modifiche e per aggiungere un genere alla famiglia aggiunta o modificata, fare click su *Add Genera*.

Per aggiungere un'altra famiglia dopo aver memorizzato le modifiche, fare click su *Next*. Questo salva la famiglia correntemente aperta ed inizia la modifica della prossima.

3.3.3 Genere

Il genere editor consente di aggiungere o modificare un genere botanico.

Il campo *Famiglia* nell'editore di generi permette di selezionare la famiglia a cui appartiene il genere. La famiglia dev'essere già presente nella base dati per poter essere selezionata. Iniziare a scrivere il nome, Ghini mostrerà quelli che coincidono.

Il campo *Epiteto* permette di specificare l'epiteto di questo genere.

Il campo *Autore* permette di specificare il nome o l'abbreviazione dell'autore o gli autori della pubblicazione del genere.

Synonyms allow you to add other genera that are synonyms with the genus you are currently editing. To add a new synonyms type in a genus name in the entry. You must select a genus name from the list of completions. Once you have selected a genus name that you want to add as a synonym click on the Add button next to the synonym list and it will add the selected synonym to the list. To remove a synonym select the synonym from the list and click on the Remove button.

Per annullare le modifiche, fare click sul pulsante *Cancel/Annulla*.

Per salvare il genere sta lavorando, quindi fare clic su * OK *.

Per salvare il genere sta lavorando e aggiungere una specie ad esso, quindi fare clic sul * specie * Aggiungi pulsante.

Per aggiungere un altro genere, quando si è finito di modificare la corrente di uno scatto sulla * avanti * pulsante sul fondo. Questo salverà il genere corrente e aprire un nuovo editor di genere vuoto.

3.3.4 Specie / Tassone

Per ragioni storiche, chiamati una `species`, ma da questo si intende un `taxon` di rango specie o inferiore. Esso rappresenta un nome univoco nel database. L'editor di specie consente di costruire il nome così come associare metadati con il taxon come la sua distribuzione, sinonimi e altre informazioni.

La sezione *informazione infraspecifica* nell'editor di specie consente di specificare il `taxon` oltre al rango di specie.

Per annullare le modifiche, fare click sul pulsante *Cancel/Annulla*.

Per salvare la specie stanno lavorando, quindi fare clic su * OK *.

Per salvare la specie stanno lavorando e aggiunta un'accessione ad esso, quindi fare clic sul * accessione * Aggiungi pulsante.

Per aggiungere un'altra specie quando si è finito di modificare la corrente di uno scatto sulla * avanti * pulsante sul fondo. Questo salverà la specie attuale e aprire un nuovo editor di specie in bianco.

3.3.5 Accessioni

L'editore di accessioni ci permette di aggiungere accessioni relative ad una specie. In Ghini una accessione rappresenta una pianta, un gruppo di piante, tutte della stessa specie, tutte propagate allo stesso modo e con lo stesso trattamento, tutte ricevute allo stesso momento dalla stessa fonte.

Scegli il nome della specie, o aggiungine una se manca quella che serve.

Si può segnalare l'incertezza nell'identificazione con l'aggiunta di un qualificatore di identificazione, al rango corrispondente, così è possibile ad esempio avere una pianta inizialmente identificata come *Iris* cf *florentina* selezionando *Iris florentina* come specie, e porre come qualificatore di identificazione "cf", essendo la "specie" il rango così qualificato.

Scrivi l'identificativo dell'accessione, e preferibilmente anche la quantità ricevuta.

Fonte dell'Accessione

La fonte delle accessioni consente di aggiungere ulteriori informazioni sulla provenienza della accessione. Selezionare un contatto dall'elenco a tendina, o scegliere «Propagazione del giardino», primo elemento predefinito nell'elenco dei contatti.

Una propagazione del giardino è il risultato di una propagazione andata a buon fine.

Quando si accessiona materiale di una propagazione di giardino, lascia inizialmente vuota la prima scheda (generale) e inizia dalla seconda scheda (fonte). Seleziona come contatto «Propagazione del Giardino», indicare quale pianta è la pianta madre e scegli tra le propagazioni ancora non completamente accessionate quella che si desidera aggiungere come accessione nel database.

Una volta che si seleziona una propagazione, il software riempie alcuni campi nella scheda generale, che è ora bene esaminare. Il Taxon (forse sei riuscito a ottenere qualcosa di leggermente diverso da quello della pianta madre). La quantità iniziale (in caso che non tutte le piante vanno nella stessa accessione). Il tipo di materiale, derivato dal tipo di propagazione.

3.3.6 Pianta

Una `Plant` nella base dati Ghini rappresenta un individuo nella collezione fisica. Una pianta appartiene ad un'accessione, ed ha una sua collocazione.

Creare più piante allo stesso tempo

È possibile creare più di una pianta in una sola volta, specificando un intervallo come codice di pianta. Questa opzione è disponibile solo al momento di inserire una nuova pianta, non quando se ne sta modificando una già presente nella base dati.

Per esempio, l'intervallo 3-5 creerà tre piante, con codici rispettivamente 3, 4 e 5. L'intervallo 1,4-7,25 crea sei piante con codici rispettivamente 1, 4, 5, 6, 7 e 25.

Quando si immette l'intervallo nella voce del codice planting la voce diventa blu per indicare che si stanno ora creando planting multipli. Tutti i campi che sono impostati in questa modalità verranno copiati tutte le piante che vengono create.

Fotografie

Come quasi tutti gli oggetti in un database Ghini possono avere associate *note*, allo stesso modo Piante e Specie possono vedersi associate *immagini*: accanto alla scheda per le note, l'editor delle piante e quello delle specie contengono una scheda aggiuntiva chiamata «Pictures». È possibile associare ad una pianta o ad una specie tante immagini quanto sia necessario.

Quando si associa un'immagine ad un oggetto, il file viene copiato nella cartella immagini ed una miniatura (500x500) è generata e copiata nella cartella `thumbnails` all'interno della cartella immagini.

Con Ghini-1.0.62, le immagini non restano nel database. Per garantire che le immagini siano disponibili su tutti i terminali dove s'è installato e configurato Ghini, è possibile utilizzare una cartella di rete, o un servizio di condivisione di file, come Tresorit o Dropbox.

Ricordo di aver configurato la cartella radice di immagini quando specificato i dettagli della connessione di database. Ancora una volta, è necessario assicurarsi che la cartella principale di immagini è condivisa con il vostro servizio di scelta di condivisione di file.

When a Plant or a Species in the current selection is highlighted, its pictures are displayed in the pictures pane, the pane left of the information pane. When an Accession in the selection is highlighted, any picture associated to the plants in the highlighted accession are displayed in the pictures pane.

In Ghini-1.0, pictures are special notes, with category «<picture>», and text the path to the file, relative to the pictures root folder. In the Notes tab, Picture notes will show as normal notes, and you can edit them without limitations.

A Plant is a physical object, so you associate to it pictures taken of that individual plant, taken at any relevant development stage of the plant, possibly helping its identification.

Species are abstract objects, so you would associate to it pictures showing the characteristic elements of the species, so it makes sense to associate a flora illustration to it. You can also do that by reference: go to the Notes tab, add a note and specify as category «<picture>», then in the text field you type the URL for the illustration of your choice.

3.3.7 Ubicazioni

Editore delle Collocazioni

zona di pericolo

L'editor di percorso contiene una sezione inizialmente nascosta denominata * pericolo zona *. I widget contenuti in questa sezione consentono di unire la posizione corrente in un'altra

posizione, consentendo all'utente di correggere errori di ortografia o implementare le modifiche dei criteri.

3.4 Si occupano di propagazioni

Ghini offre la possibilità di associare prove di propagazioni di piante e di documentare i loro trattamenti e risultati. Trattamenti sono parte integrante della descrizione di una prova di propagazione. Se una prova di propagazione è successo, Ghini consente di associarla a una nuova accessione. È possibile associare solo una accessione ad una prova di propagazione.

Qui descriviamo come si utilizza questa parte dell'interfaccia.

3.4.1 Creazione di una propagazione

Una propagazione (trial) è ottenuta da una pianta. Ghini si riflette nella sua interfaccia: selezionare una pianta, aprire l'Editor di pianta su di esso, attivare la scheda di propagazione, fare clic su Aggiungi.

Quando si fare quanto sopra, si ottiene una finestra di Editor di propagazione. Ghini non considera prove di propagazione come entità indipendenti. Di conseguenza, Ghini considera l'Editor di propagazione come una finestra di editor speciale, che si può raggiungere solo dall'Editor di pianta.

Per una propagazione di nuova, si seleziona il tipo di propagazione (che diventa una proprietà immutabile della propagazione) quindi inseriscono i dati che lo descrive.

Sarete in grado di modificare i dati di propagazione tramite lo stesso percorso: selezionare una pianta, aprire l'Editor di pianta, identificare la propagazione si desidera modificare, fare clic sul pulsante modifica corrispondente. Sarete in grado di modificare tutte le proprietà di una prova, tranne il tipo di propagazione esistente.

Nel caso di una prova di propagazione del seme, si dispone di un genitore di polline e un genitore del seme. È sempre necessario associare il processo di propagazione al padre di seme.

Nota: In Ghini-1.0 è possibile specificare la pianta del genitore di polline nella «Notes» di campo, mentre Ghini-1.1 ha un campo (relazione) per esso. Secondo ITF2, potrebbero esserci casi nelle prove di propagazione di seme dove non è conosciuto che pianta svolge quale ruolo. Ancora una volta, in Ghini-1.0 è necessario utilizzare una nota per indicare che se questo è il caso, Ghini-1.1 presenta un campo (boolean) che indica se questo è il caso.

3.4.2 Utilizzando una propagazione

Una prova di propagazione può essere successo e provocare una nuova accessione.

Ghini ti aiuta a riflettere questo nel database: creare una nuova accessione, immediatamente passare alla scheda origine e selezionare «Garden Propagation» nel campo contatto (devo ammettere che un po " fuorviante).

Iniziare a digitare il numero di piante e verrà visualizzato un elenco di piante corrispondenti con prove di propagazione per selezionare da.

Selezionare l'planting e l'elenco delle prove di propagazione raggiunto e inutilizzato apparirà nella parte inferiore della finestra.

Selezionare una versione di prova di propagazione ancora inutilizzato dall'elenco e fare clic su Ok per completare l'operazione.

Utilizzando i dati dalla prova di propagazione, Ghini completa alcuni campi nella scheda Generale: nome del Taxon, tipo di materiale e possibilmente di provenienza. Sarete in grado di modificare questi campi, ma siete pregati di notare che il software non impedirà l'introduzione concettuale incoerenze nel database.

È possibile associare una prova di propagazione sola accessione.

3.5 Etichette

Tagging is an easy way to give context to an object or create a collection of object that you want to recall later.

The power in this tagging action is that you can share this selection with colleagues, who can act on it, without the need to redo all your collecting work.

For example if you need to print accession labels of otherwise unrelated plants, you can group the objects by tagging them with the string «relabel». You or one of your colleagues can then select «relabel» from the tags menu, the search view will show all the objects you tagged, and performing a report will act on the tagged objects.

Tagging acts on the active selection, that is the items in the search results which you have selected.

Please remember: you can select all result rows by pressing `Ctrl-A`, you can deselect everything by pressing `Ctrl-Shift-A`, you can toggle tagging of a single row by `Ctrl-Mouse click` on it.

Once you have an active selection, tagging can be done in two ways:

3.5.1 dialog box tagging

Press `Ctrl-T` or select *Tag→Tag Selection* from the menu, this activates a window where you can create new tags and apply any existing tag to the selection.

The tag window is composed of three parts:

1. The upper part mentions the list of objects in your active selection. This is the list of object of which you are editing the tags;

2. The middle part has a list of all available tags, with a checkbox that you can activate for applying the tag to or removing the tag from the selection;
3. The lower part only holds a link to new tag creation, and the Ok button for closing the dialog box.

If, when opening the tag dialog box, the active selection holds multiple items, then only the tags that are common to all the selected items will have a check next to it. Tags that only apply to a proper subset of the active selection will show with an "undecided" status. Tags that don't apply to any object in the active selection will show blank.

The most recently created tag, or the most recently selected tag becomes the active tag, and it shows with a check next to it in the tags menu.

3.5.2 windowless tagging

Once you have an active tag, pressing `Ctrl-Y` applies the active tag to all objects in the active selection. `Ctrl-Shift-Y` removes the active tag from all objects in the active selection.

3.6 Generare relazioni

A database without exporting facilities is of little use. Ghini lets you export your data in table format (open them in your spreadsheet editor of choice), as labels (to be printed or engraved), as html pages or pdf or postscript documents.

3.6.1 The Report Tool

You activate the Report Tool from the main menu: *Tools→Report*. The Report Tools acts on a selection, so first select something, then start the Report Tool.

Report on the whole collection.

To produce a report on your whole plant collection, a shortcut would be from the home screen, to click on the `Families: in use` cell.

If your focus is more on the garden location than on taxonomy and accessions, you would click on the `Locations: total` cell.

Reports are produced by a report engine, making use of a report template. Ghini relies upon two different report engines (Mako & XSL), and offers several report templates, meant as usable examples.

Choose the report you need, specify parameters if required, and produce the report. Ghini will open the report in the associated application.

Configuring report templates, that's a task for who installs and configures ghini at your institution. Basically, you create a template name, indicating the report engine and specifying the

template. Configured templates are static, once configured you are not expected to alter them. Only the special `**scratch**` template can be modified on the fly.

The remainder of this page provides technical information and links regarding the formatter engines, and gives hints on writing report templates. Writing templates comes very close to writing a computer program, and that's beyond the scope of this manual, but we have hints that will definitely be useful to the interested reader.

3.6.2 Utilizzando il formattatore di Report di Mako

Il formattatore di relazione Mako utilizza il linguaggio del modello di Mako per la generazione di report. Ulteriori informazioni su Mako e il suo linguaggio si trovano sul sito makotemplates.org.

Il sistema di template Mako dovrebbe già essere installato sul tuo computer se è installato Ghini.

Creazione di report con Mako è simile nel modo in cui si creerà una pagina web da un modello. È molto più semplice di Formatter(see below) il XSL e dovrebbe essere relativamente facile creare un modello per chiunque con un po " ma di esperienza di programmazione.

The template generator will use the same file extension as the template which should indicate the type of output the template with create. For example, to generate an HTML page from your template you should name the template something like *report.html*. If the template will generate a comma separated value file you should name the template *report.csv*.

Il modello riceverà una variabile denominata "valori" che conterranno l'elenco di valori nella ricerca attuale.

Il tipo di ogni valore in "valori" sarà lo stesso come il dominio di ricerca utilizzato nella query di ricerca. Per ulteriori informazioni su domini di ricerca vedere: ref: domini di ricerca.

Se la query non dispone di un dominio di ricerca quindi tutti i valori possono essere di diverso tipo e il modello di Mako dovrebbe preparare per gestire loro.

3.6.3 Utilizzando il formattatore di Report XSL

Il formattatore di relazione XSL richiede un XSL per renderer PDF per convertire i dati in un file PDF. Apache FOP è è un libero e open-source XSL-> renderer PDF ed è consigliato.

Se si utilizza Linux, Apache FOP dovrebbe essere installabile utilizzando il vostro gestore di pacchetti. Su Debian/Ubuntu è installabile come "fop " in Synaptic o utilizzando il comando seguente:

```
apt-get install fop
```

L'installazione di Apache FOP su Windows

Ci sono due modi per installare FOP su Windows. Il più semplice è scaricando il precompilato **'ApacheFOP-0.95-1-setup.exe**

<http://code.google.com/p/apache-fop-installer/downloads/detail?name=ApacheFOP-0.95-1-setup.exe&can=2&q=#makechanges >‘_.

In alternativa è possibile scaricare l’archivio. Dopo l’estrazione dell’archivio è necessario aggiungere al PATH la directory dove è stato estratto l’archivio.

3.7 Importare ed esportare dati

Anche se Ghini può essere ampliato tramite plugin, così da poter supportare formati alternativi per importare ed esportare dati, per impostazione predefinita può solo importare ed esportare file di valori separati da virgola o CSV.

C’è qualche supporto per l’esportazione dell’accesso per dati biologici collezioni è limitato.

C’è anche un supporto limitato per l’esportazione in formato XML che più o meno riflette esattamente le tabelle e la riga del database.

Esportazione di ABCD e XML non sono coperti qui.

Avvertimento: Importazione di file molto probabilmente distruggerà tutti i dati che avete nel database in modo assicurarsi che avete eseguito il backup dei dati.

3.7.1 Importazione da

In generale è meglio solo importare i file CSV in Ghini che sono stati precedentemente esportati da Ghini. È possibile importare qualsiasi file CSV ma che è più avanzato che coprirà questo doc.

Per importare file CSV in Ghini selezionare *Strumenti→ esportazione→ valori separati da virgola* dal menu.

Dopo facendo clic su OK nella finestra di dialogo che chiede se sei sicuro di che sapere quello che stai facendo si aprirà una finestra di selezione file. Nella finestra selezione file selezionare i file che si desidera importare.

3.7.2 Esportazione in CSV

Per esportare i dati di Ghini in formato CSV selezionare *Strumenti→ esportazione→ valori separati da virgola* dal menu.

Questo strumento vi chiederà di selezionare una directory per esportare i dati in CSV. Tutte le tabelle in Ghini verranno esportate ai file in tablename.txt il formato dove tablename è il nome della tabella dove è stati esportati i dati da.

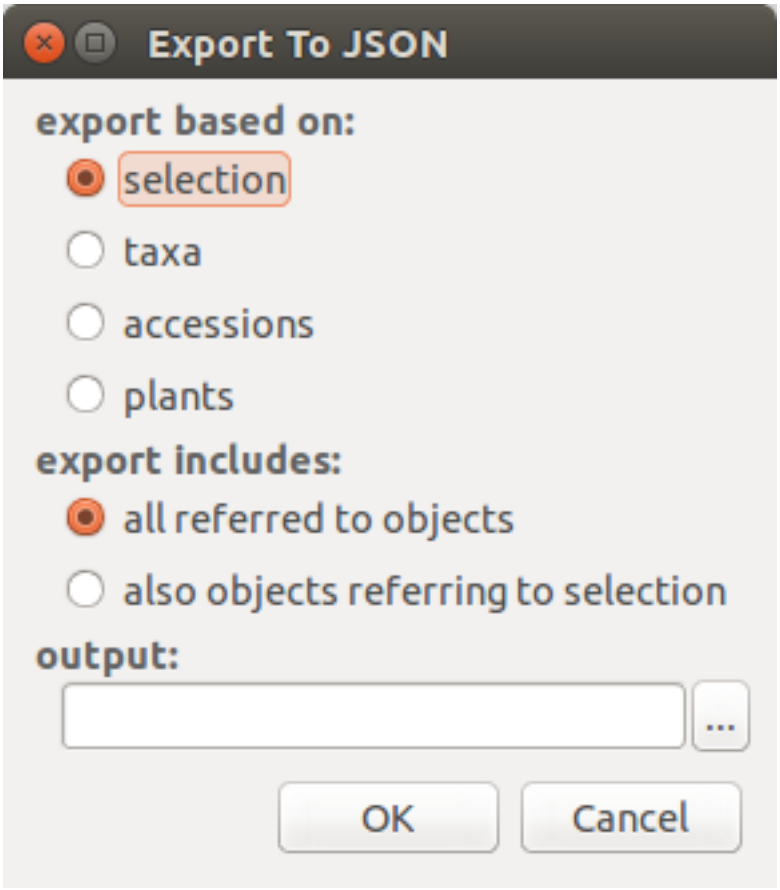
3.7.3 Importare da JSON

Questo è * il * modo per importare dati in un database esistente, senza distruggere il contenuto precedente. Un tipico esempio di questa funzionalità potrebbe essere importazione tua collezione digitale in un database di Ghini fresco, appena inizializzato. Conversione di un database in formato di interscambio di json Bauble esula dall'ambito del presente manuale, si prega contattare uno degli autori se avete bisogno di ulteriore aiuto.

Utilizzando il formato di scambio JSON, è possibile esportare dati da una versione di Ghini ed importarli in una versione differente.

3.7.4 Esportare verso JSON

Questo è ancora "lavori in corso".



Quando si attiva questo strumento di esportazione, si è data la scelta di specificare cosa esportare. È possibile utilizzare la selezione corrente per limitare l'intervallo di esportazione, oppure è possibile avviare il contenuto completo di un dominio, a scelta tra specie, accessione, pianta.

Esportazione di * specie * solo esporterà le informazioni tassonomiche complete nel database. * Accessione * esporterà tutte le vostra accessioni oltre a tutte le informazioni tassonomiche si riferisce a: unreferred a taxa non verrà esportati. * Planting * esporterà tutti i viventi, piante (alcuni accessione non può essere incluso), tutti si riferiscono a posizioni e taxa.

3.7.5 Importare da una base dati generica

Questa funzionalità è oggetto della issue #127, per la quale non abbiamo ancora una soluzione generica.

Dovendo importare dati da una tabella, da un foglio di calcolo, o da un database non compatibile con ghini, contattare un programmatore.

3.7.6 Importare una collezione di immagini

Una particolare forma di base dati botanica è una collezione di fotografie, ciascuna chiaramente associata ad una specifica pianta.

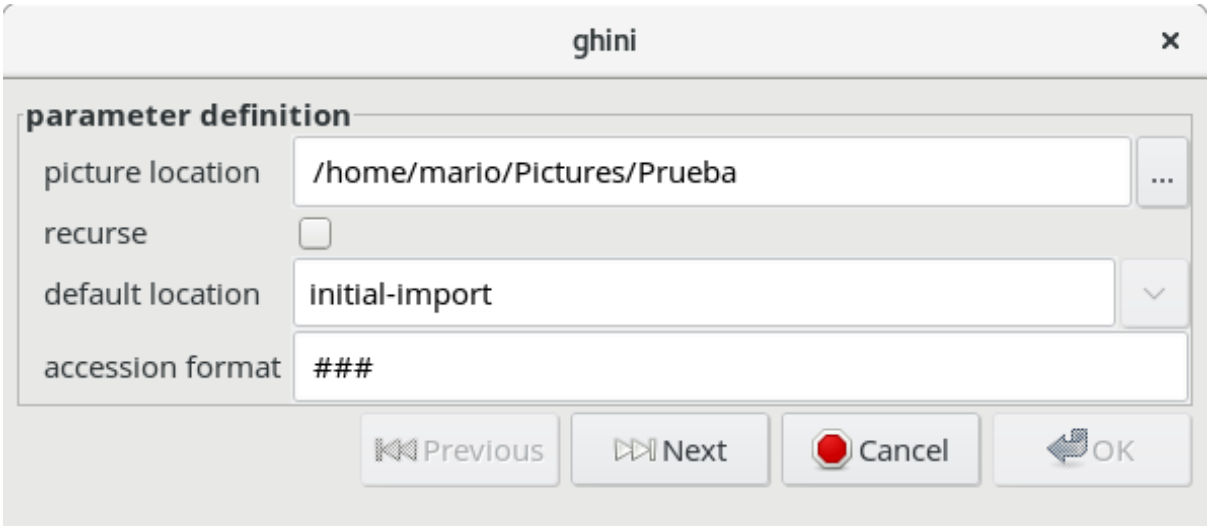
Anche senza utilizzare un software disegnato per collezioni di foto, è possibile associare foto ad accessioni attenendosi ad una stessa regola nel dare nome ai file delle foto.

Per esempio, `2018.0020.1 (4) Epidendrum.jpg` potrebbe essere il nome della quarta fotografia associata alla pianta numero 1 della ventesima accessione dell'anno 2018, identificato a rango genus come Epidendrum.

La funzione *Tools→Import→Pictures* qui descritta permette di utilizzare una collezione di foto per inizializzare una base dati ghini, o per aggiungere dati periodicamente.

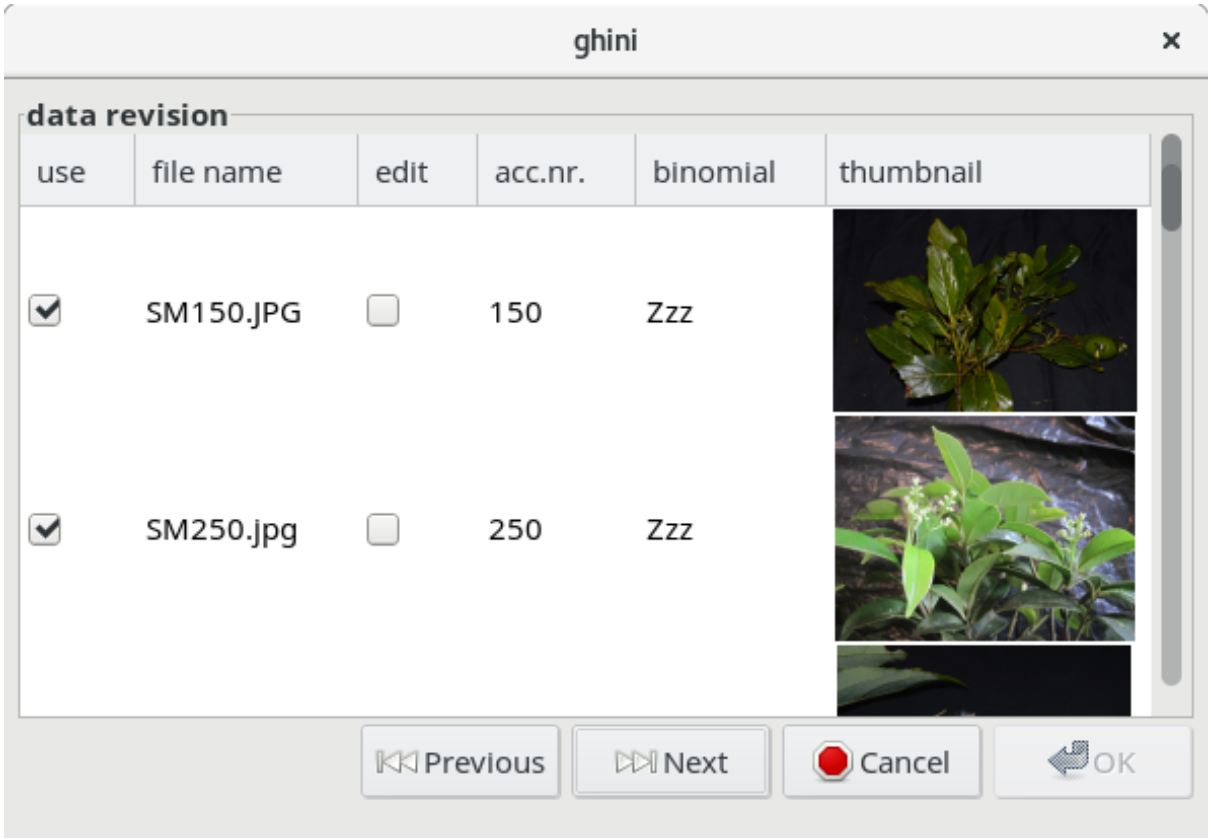
Use *Tools→Import→Pictures* to activate this import tool. Import goes in several steps: parameter definition; data revision and confirmation; the import step proper; finally review the import log. At the first two steps you can confirm the data and go to the next step by clicking on the `next` button, or you can go back to the previous step by clicking on the `prev` button. Once the import is done and you're reviewing the log, you can only either confirm —or abort— the whole transaction.

Nella fase di definizione parametri puoi: selezionare la directory da cui importare le foto; indicare se l'importazione va effettuata in modo ricursivo; selezionare o creare l'ubicazione da usare per le nuove piante; descrivere la regola seguita nel dare nome ai file delle foto.

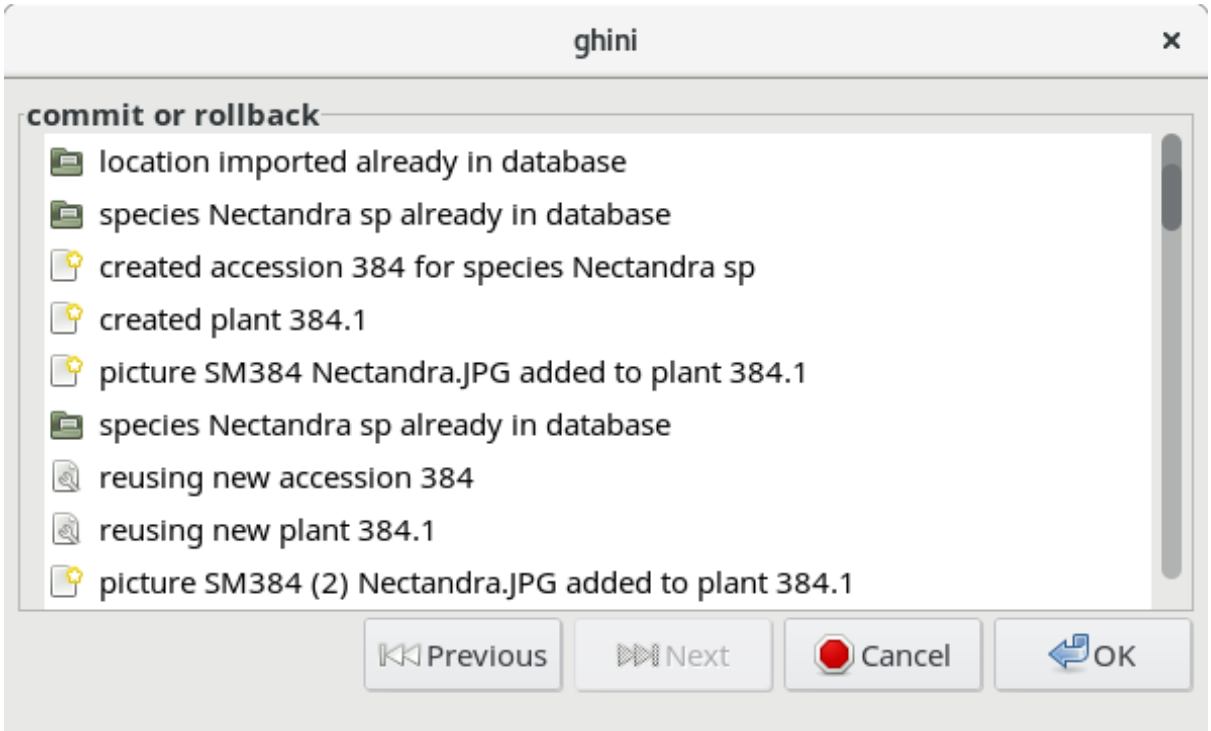


La fase di revisione consiste nel visionare una tabella con tante righe quante le foto da importare. Ogni riga contiene tanta informazione quanto il tool avrà potuto estrarre dal nome di

ciascuna foto. Si può accettare ciascun campo così come definito dal tool, e si può indicare se ciascuna riga va importata o meno.



la fase finale di «commit or rollback» consiste nel visionare il rapporto dell'importazione e decidere se mantenere i cambi (commit), o rifiutarli in blocck (rollback).



When the Picture Collection importer creates or updates objects, it also sets a Note that you

can use for selecting the objects involved in the import, and for reviewing if needed.

3.8 Gestione Utenti

Nota: Il plugin di utenti Ghini disponibile solo su database PostgreSQL basata.

The Ghini Users Plugin will allow you to create and manage the permissions of users for your Ghini database.

You must log in to your database as a user with `CREATEROLE` privilege in order to manage other users.

3.8.1 Creare Utenti

Crea nuovo utente...

3.8.2 Permessi

Ghini consente lettura, scrittura ed esecuzione autorizzazioni.

CAPITOLO 4

Cookbook

4.1 Contributed recipes collection

This page presents lists of use cases. If you're looking for straight, practical information, you are at the right place. If you prefer a thorough presentation of the software and database structure, check the section software for botanical gardens

All material here has been contributed by gardens using the software and sharing their experiences back to the user community.

The authors of the software wish to thank all dearly.

4.1.1 Orto botanico di Quito

Presso il JBQ, orto botanico di Quito, abbiamo adottato il software Ghini in aprile 2015. Da quel momento, abbiamo accumulato esperienza con il programma, e siamo noi stessi nella necessità di documentare, al fine di assicurare la conoscenza all'istituzione. Siamo felici di condividere.

Dettagli tecnici

- Lavoriamo su GNU/Linux, una piattaforma che molti utenti non dominano, e il nostro database è all'interno di un sistema di gestione di database. Ciò implica passaggi che non sono evidenti all'utente finale casuale.

Come avviare un programma

per avviare un programma dato il suo nome, premi il tasto  accanto ad Alt, o fare clic su  , quindi iniziare a digitare il nome del programma, nel nostro caso «Ghini» o semplicemente cliccare sul simbolo del programma  , che appare al margine sinistro dello schermo.

Server di database

Abbiamo scelto un server centralizzato di database PostgreSQL. In questo modo siamo protetti da conflitti fra modifiche simultanee, e tutte le modifiche sono disponibili simultaneamente su tutti i client ghini. Abbiamo però bisogno di esternalizzare la gestione del server di database.

Creare un nuovo utente

Ghini tiene traccia dell'utente che esegue qualsiasi modifica al database. Nel nostro giardino, a parte gli utenti stabili, abbiamo spesso utenti temporanei abilitati alla scrittura nel database. Abbiamo deciso lasciarci aiutare da Ghini a tenere traccia degli eventi del database.

Since we work using PostgreSQL, the users that Ghini stores in the database history are the database users, not the system users.

Each user knows their own password, and only knows that one. Our super-user, responsible for the database content, also has the `bauble` fictional user password, which we only only use to create other users.

We do not use account names like `voluntario`, because such accounts do not help us associate the name to the person.

— Creare un nuovo utente di sistema

Adding a system user is not strictly necessary, as ghini does not use it in the logs, however, adding a system user allows for separation of preferences, configured connections, search history. On some of our systems we have a single shared account with several configured connections, on other systems we have one account per user.

On systems with one account per user, our users have a single configured connection, and we hold the database password in the `/home/<account>/.pgpass` file. This file is only readable for the `<account>` owner.

On systems with a shared account, the user must select their own connection and type the corresponding password.

These are the steps to add system users:

```
sudo -k; sudo adduser test
sudo adduser test adm; sudo adduser test sudo
sudo adduser test sambashare; sudo adduser test ghini
```

— aggiungere un utente del database

Ghini has a very minimal interface to user management, it only works with postgresql and it very much lacks maintainance. We have opened issues that will allow us use it, for the time being we use the `create-role.sh` script:

```
#!/bin/bash
USER=$1
PASSWD=$2
shift 2
cat <<EOF | psql bauble -U bauble $@
create role $USER with login password '$PASSWD';
alter role $USER with login password '$PASSWD';
grant all privileges on all tables in schema public to
 ↪$USER;
grant all privileges on all sequences in schema public␣
 ↪to $USER;
grant all privileges on all functions in schema public␣
 ↪to $USER;
EOF
```

The redundant `alter role` following the `create role` lets us apply the same script also for correcting existing accounts.

Our ghini database is called `bauble`, and `bauble` is also the name of our database super user, the only user with `CREATEROLE` privilege.

For example, the following invocation would create the user `willem` with password `orange`, on the `bauble` database hosted at 192.168.5.6:

```
./create-role.sh willem orange -h 192.168.5.6
```

- Capire quando aggiornare

Aggiornare il sistema

Ubuntu updates are a lot lighter and easier than with Windows. So whenever the system suggests an update, we let it do that. Generally, there's no need to wait during the update nor to reboot after it's done.

Aggiornare Ghini

La prima finestra presentata da Ghini assomiglia a questo. Normalmente non devi fare nulla in questa finestra, solo premere invio ed entrare nella schermata principale del programma.

Occasionalmente, nella parte superiore dello schermo apparirà un testo, che t'informa che una versione aggiornata è disponibile on-line.

La procedura di aggiornamento è semplice, e dipende dal sistema operativo in uso, non stiamo a spiegarlo di nuovo.

È generalmente una buona idea aggiornare il software. In caso di dubbio, contattare l'autore, o scrivere al gruppo.

- la schermata iniziale di ghini

Schermo completo

Al momento di scrivere, la nostra schermata iniziale si presentava così:

Oltre al menù principale dell'applicazione, Ghini offre tre sezioni di interfaccia speciale con informazioni e strumenti per esplorare il database.

Panoramica numerica

La tabella nella parte destra dello schermo presenta un riepilogo di tutte le piange nel database. Ogni voce in grassetto è un link alla query che seleziona gli oggetti corrispondenti.

	total	in use	unused
Families:	511	7	504
Genera:	25394	158	25236
Species:	637	623	14
Accessions:	7722	7675	47
Plants:	7676	7676	0
Locations:	170	163	7

Query memorizzate

La metà inferiore del lato destro contiene un set di query memorizzate. Mentre è possibile modificarle a proprio piacimento, suggeriamo includere la selezione di accessioni non identificate al rango specie. E uno per la cronologia del database.

Pulsanti di azione e di query

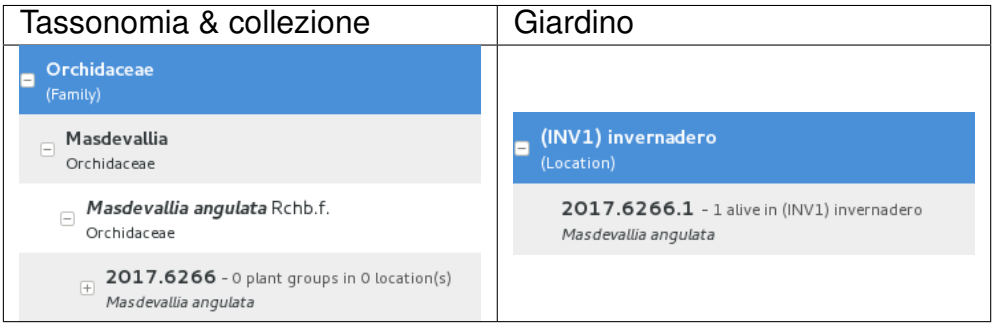
Nella parte superiore della schermata sta il campo in cui immettere le ricerche.

- Con il pulsante  , una casa, è possibile tornare dalle tue ricerche alla schermata principale.
- Con il pulsante  , una freccia, è possibile tornare alla ultima ricerca.
- Con il pulsante  , un ingranaggio, è possibile avviare il «Generatore di Query», che ti aiuta a comporre ricerche complesse in modo semplice, grafico.

- Abbiamo spesso volontari che lavorano presso il giardino per un tempo molto breve. Pensando a loro abbiamo sviluppato una vista ultrasemplificata della struttura del database di ghini.

Dettagli

Le due figure qui mostrano tutto ciò che i nostri collaboratori temporanei devono sapere.

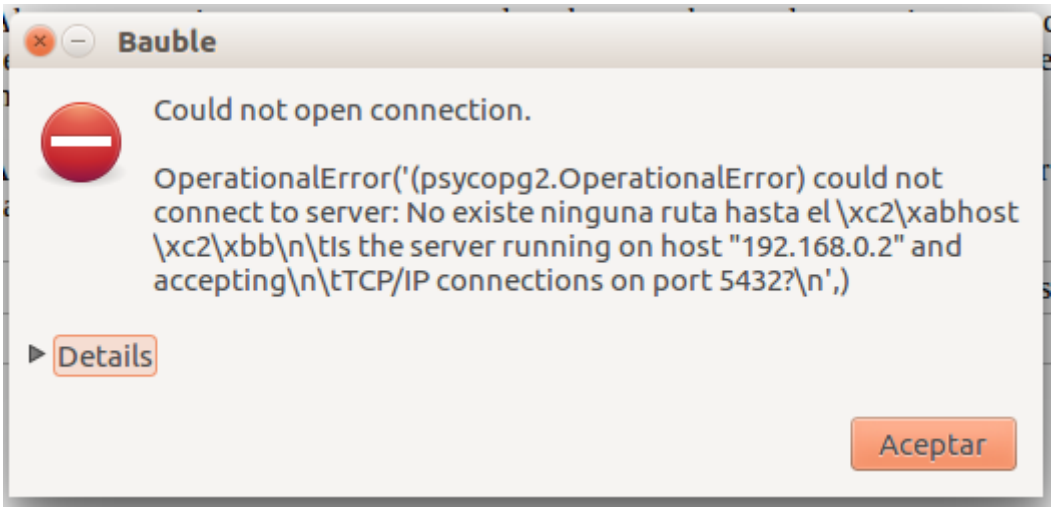


- A volte, il programma dà messaggi di errore. 

, riprova, o informa gli sviluppatori.

Problemi di rete

Per poter funzionare, il programma ha bisogno di una connessione di rete stabile al server di database. Può succedere: si avvia il programma, e non è possibile connettersi al nostro server di database. Si ottiene un messaggio di errore abbastanza chiaro ma molto mal composto.

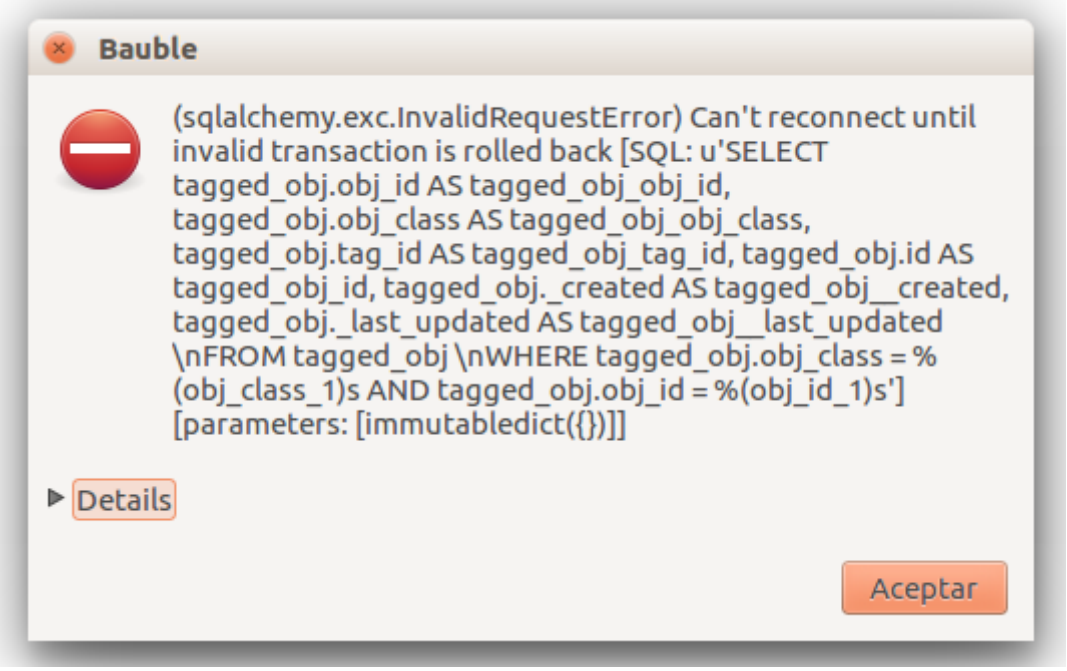


Basta ignorarlo e riprovare.

La ricerca ha esito negativo con errore

A volte e senza alcuna causa apparente, una ricerca non è eseguita correttamente e appare una finestra con un messaggio di errore. In questo caso prova a eseguire nuovamente la stessa ricerca.

Un esempio di un tale messaggio di errore:



La ricerca non restituisce qualcosa che ho appena inserito

Codici di accessione che iniziano con zero e composti di sole cifre, come ad esempio `016489` sono considerati dal software come numeri, quindi se non si racchiude la stringa di ricerca fra virgolette, tutti gli 0 iniziali verranno eliminati e il valore non verrà trovato.

Riprovare, ma racchiudere la stringa di ricerca tra virgolette singole o doppie.

Numero sull'etichetta	ricerca corrispondente
16489	"016489"

Please note: when you look for a Plant code, not an Accession, the leading zero becomes optional, so in the above example it's maybe easier to type `16489.1`.

- A serious situation happened once, and we absolutely want to prevent it from happening again: a user deleted a genus, with everything that was below it, species and accessions, and synonyms.

Solving it with user permissions

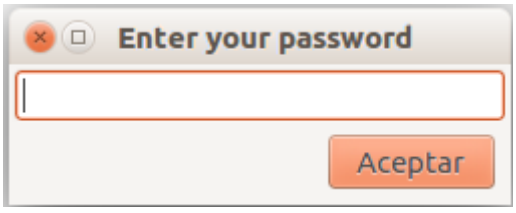
We propose to have different connection profiles, associated to different database users, each user with all needed permissions.

Full permission (BD-JBQ) Only qualified personnel get this kind of access.

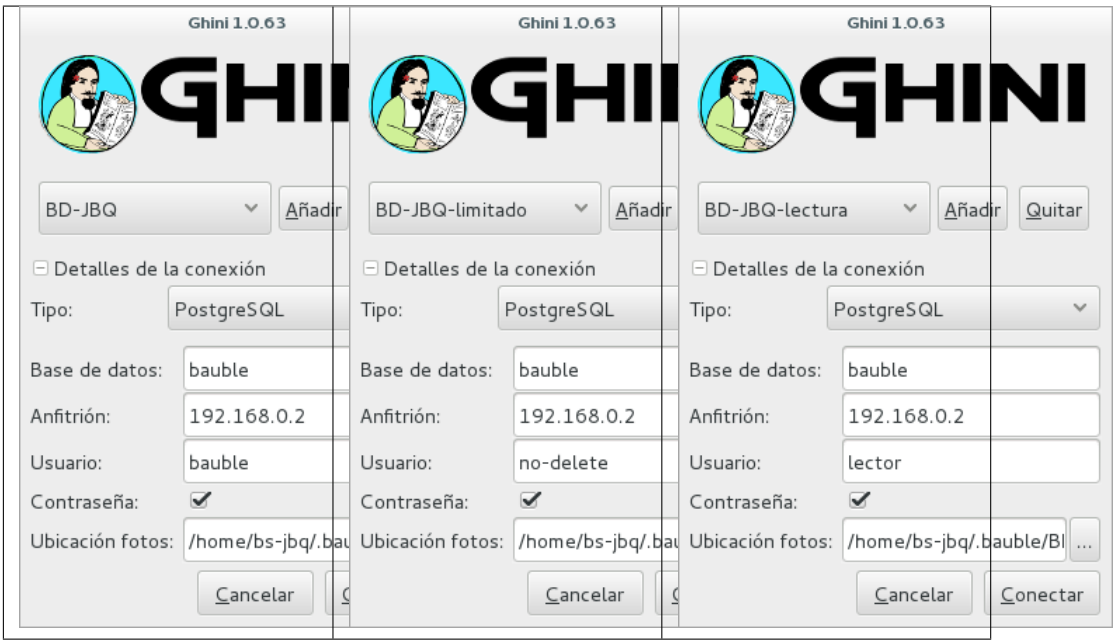
Insert and update (BD-JBQ-limitado) We use this one for those users who come help us for a limited time, and who did not get a complete introduction to database concepts. It is meant to prevent costly mistakes.

Read only (BD-JBQ-lectura) it can be shared with anyone visiting the garden

You select the connection at start-up, and the software asks you for the password corresponding to the connection you selected.



If you want to review the details of the connection, click on the next to "Connection Details", it will change to , and the connection window will be displayed as one of the following:



As you can see, we are connecting to the same database server, each connection uses the same database on the server, but with different user.

Thinking further about it

On the other hand, we are questioning if it is at all appropriate, letting any user delete something at such high level as a family, or a genus, or, for that matters, of anything connected to accessions in the collection.

The ghini way to question the software features, is by opening a corresponding issue.

- When contacting the developers, they will definitely ask for technical information, or at least to see a screen-shot. Help them help you.

Taking a screen-shot

On Linux there are three ways to create a screen-shot, all involve hitting the "PrtSc" key. The most practical one is possibly hitting the "PrtSc" key in combination with Ctrl and Shift. This will start an interactive screen copy tool. You select a rectangle and the area is copied in the clipboard. Paste it in the email you're writing, or in the chat line where the developers are trying to help you.

Dove stanno i log

Ghini continuously saves a very informative log file, in the `~/.bauble/bauble.log` file. Don't bother opening it, just send it over. It contains loads of technical information.

Continuous unmanned alerting

An other option is to activate the sentry handler. It will notify our sentry server of any serious situations in the software. If you registered, the developers will know how to contact you if necessary.

To the healthy paranoid: we're not monitoring what you're doing, we're monitoring how our software works. You can always opt out.

You activate the Sentry handler in the `:prefs` page: look for the row with name `bauble.use_sentry_handler`, if the value is not what you wish, double click on the line and it will change to the other value.

Taxonomy

- Introduzione

Orchidaceae taxonomic complexity

At the JBQ, we work most of all with orchids, family Orchidaceae, one of the largest plant families, with no less than 850 genera, organized —according to Dressler— in approximately 70 subtribes, 22 tribes, 5 subfamilies. How we represent this information is not obvious and needs be explained.

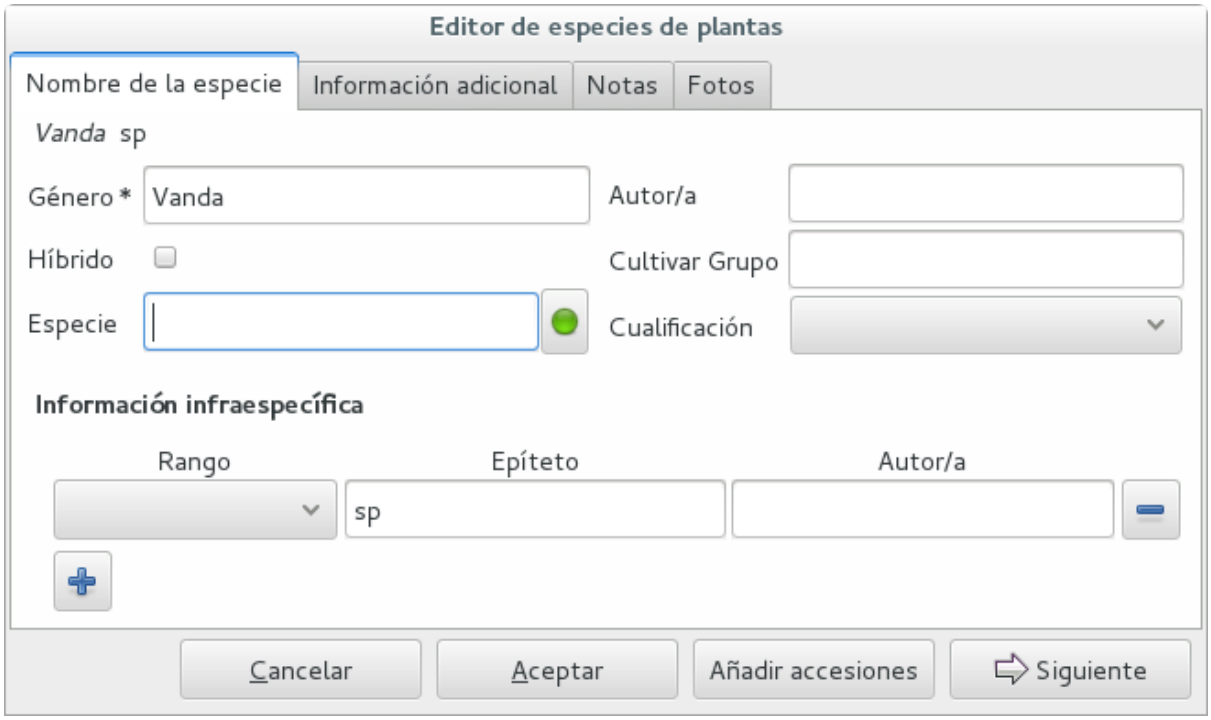
The taxonomy of the Orchidaceae family is continuously being reviewed. Genera get added, refused, reorganized, recognized as synonyms, some taxonomists prefer grouping species or genera in a new way, others split them again and differently, botanists of different nationalities may have different views on the matter. All this sounds very complex and specialistic, but it's part of our daily routine, and it can all be stored in our Ghini database.

- Identifying at rank Genus, or Family

At rank genus

Ghini-1.0 prescribes that an accession is identified at rank species, in all cases. The current maintainer acknowledges that this is a mistake, coming from the early Bauble days, and which Ghini-1.0 has in common with other botanic software. Until this is fixed, we rely on established practices.

If an accession is identified at rank genus, we add a fictive species in that genus, we don't specify its species epithet (we don't know that) and we add an unranked epithet in the infraspecific information section, like this:



When displayed in a search result, it shows like this:

At rank family

If an accession is only identified at rank family, we need a fictive genus, to which we can add the fictive species. Since our garden is primarily focusing on Orchidaceae, we use the very short name **Zzz** for the fictive genus within the family, like this:

The current maintainer suggests to use the prefix **Zzz-** and behind the prefix to write the family name, possibly removing the trailing **e**. Removal of the trailing **e** is useful in order not to get results that include genus names when you as for stuff ending in **aceae**.

Apart from the aforementioned **Zzz** genus in the Orchidaceae family, we follow this suggested practice, so for example our collection would include *Zzz-cactacea* or *Zzz-bromeliacea*.

Remember: our **Zzz** genus is a fictive genus in the **Orchidaceae** family, do not use it as unspecified genus in other families.

- Identifying at a rank that is not allowed by the software (eg: Subtribe, or Subfamily)

At rank subtribe

We sometimes can't identify a taxon at rank genus, but we do manage to be more precise than just «it's an orchid». Quite often we are able to indicate the subtribe, this is useful when you want to produce hybrids.

The software does not let us store ranks which are intermediate between family and genus, so we need to invent something, and this is what we do:

We insert a fictive genus, naming it as the subtribe, prefixing it with "Zzx-", like in this example:

This Zzx-Laeliinae is some genus in the Laeliinae subtribe.

In order to be able to select genera by subtribe, we also add a note to the Zzx-Laeliinae fictive genus as well as for all real genera in that subtribe, note category subtribus, note value the subtribe name.

This allows for queries like:

```
genus where notes.note=Laeliinae
```

We are very much looking forward to seeing that issue-9 solved!

At rank subfamily, tribe

Just as we reserved the prefix Zzx- for subtribe, we reserve the prefixes Zzy- for tribe, Zzw- for subfamily.

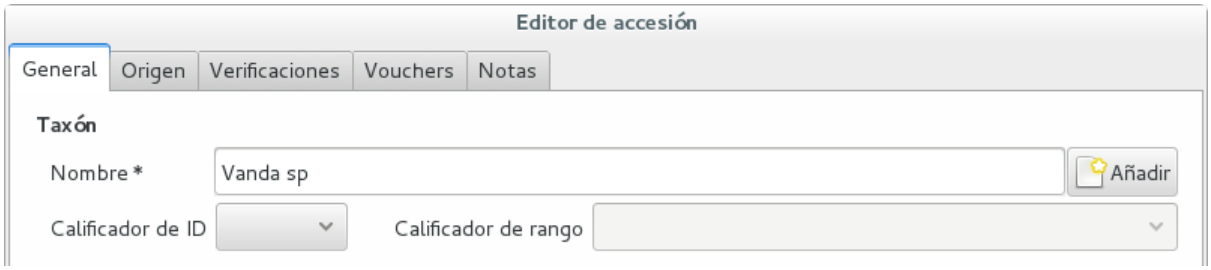
In particular, the subfamily information is relevant, because there are subfamilies within the Orchidaceae family which are not further separated.

- Editing the Accession identification - the Species details

Placeholder species for individual accessions

Scenario one describes the identification of a single accession, which had been associated to a «generic», placeholder species, something like "*Zzz* sp" or "*Vanda* sp";

In this case, when the plant species becomes known, we change the association in the accession, selecting a different species.

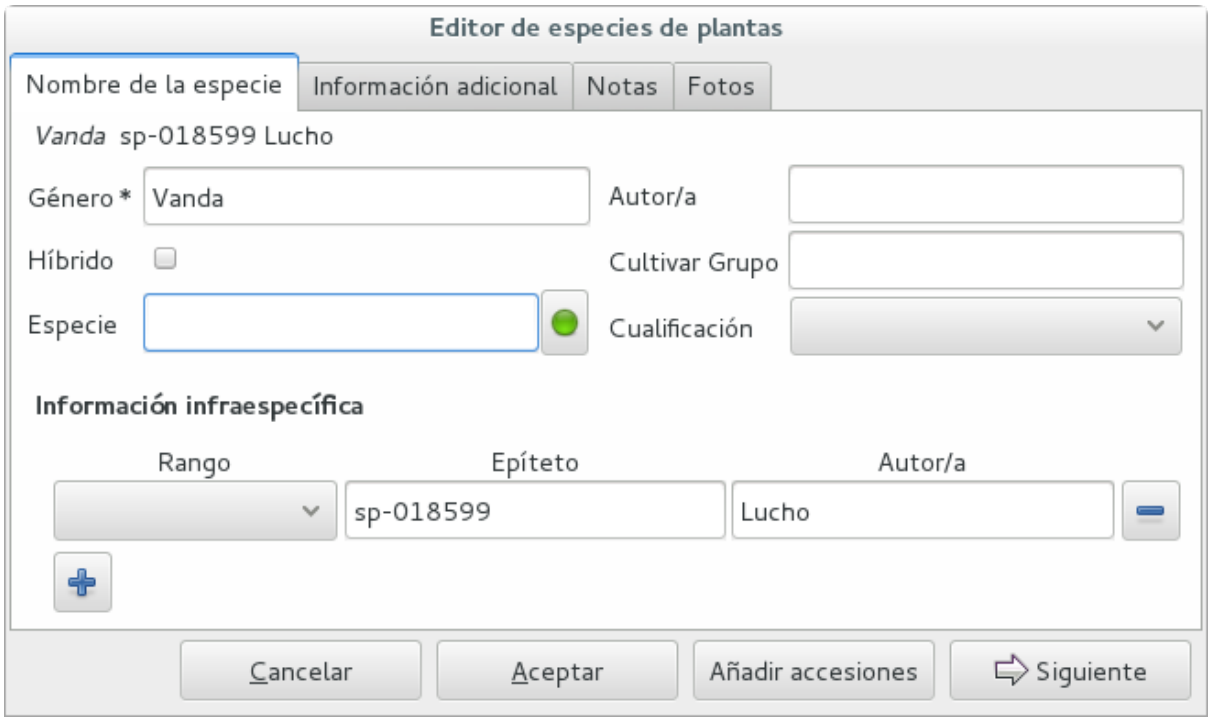


We do not edit the species, because there might be totally unrelated accessions connected to the same placeholder species.

Unknown species for multiple accessions

A different case is when we have a whole batch of accessions, all obviously the same species, but we haven't been able to identify it. In this case, we associate the accessions with an incompletely specified species, something like "*Zzz* sp-59", preferably adding the taxonomist's name, who made the association.

A species like "*Vanda* sp-018599" is not a placeholder species, it is a very concrete species, which we haven't yet identified.



In this case, when the species gets identified (and it could even be a species nova), we directly edit the species, so all accessions that refer to it get the change.

- A new plants is relative to a species not yet in our collection.

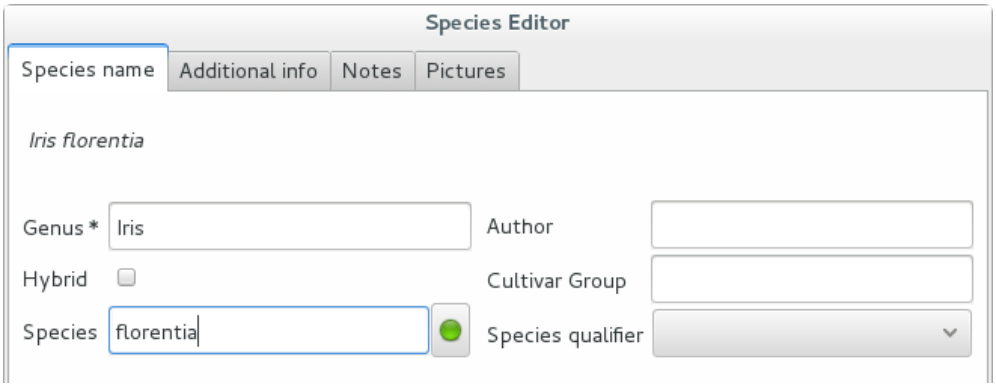
Last minute species

We start this from the Accession window and it's very simple, just click on the **+** next to the species name, we get into the Species window.

- Adding a species and using online taxonomic services

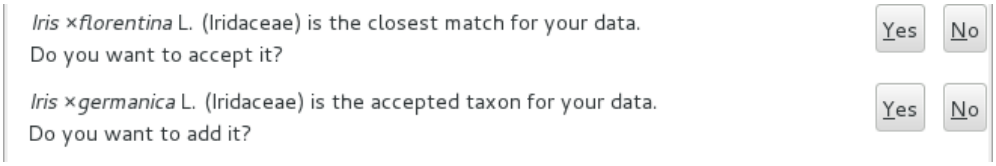
Adding a new species — the plant list.

We start the obvious way: type the genus epithet, possibly select it from the completion list, then type the species epithet, or at least your best guess.

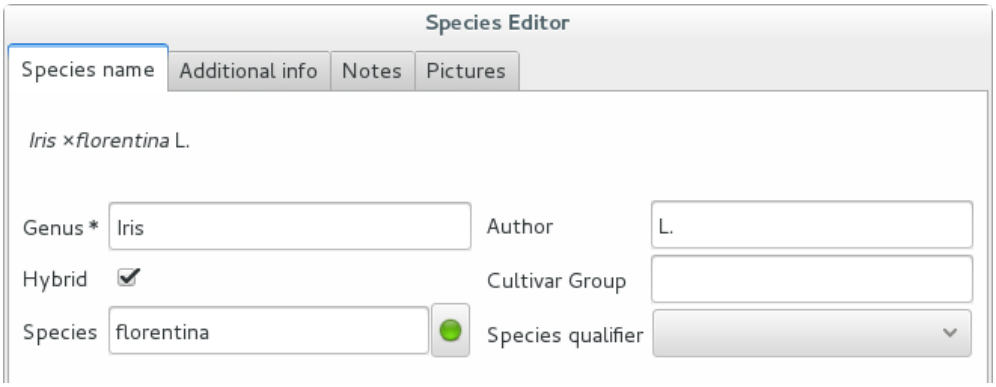


Next to the species epithet field there's a small button,  , which connects us to the plant list. Click on it, a message area appears at the top of the window.

Depending on the speed of your internet connection, but also on how close your best guess is to a correct published name, the top area will change to something like this:



Accept the hint and it will be as if you had typed the data yourself.



Reviewing a whole selection — TNRS.

This is described in the manual, it's extremely useful, don't forget about it.

Let the database fit the garden

- A never-ending task is reviewing what we have in the garden and have it match what we have in the database.

Initial status and variable resources

When we adopted ghini, we imported into it all that was properly described in a filemaker database. That database focused solely on Orchids and even so it was far from complete. In practice, we still meet labeled plants in the garden which have never been inserted in the database.

From time to time, we manage to get resources to review the garden, comparing it to the collection in the database, and the main activity is to insert accession codes to the database, take pictures of the plant in question, and note its location, all tasks that are described in the remainder of this section.

The small Android app ghini.pocket was added to the Ghini family while a Ghini programmer was here in Quito. It helps us take a snapshot of the database in our pocket while walking in the garden, but it also allows for a very swift inventory procedure.

Inventory procedure

We start ghini.pocket, we write down the name of the location where we will be conducting the inventory, for example (INV 1) for greenhouse 1. We enter (type or scan if the plant has bar code or QR code) the accession code and we look it up in ghini.pocket.

A side effect of performing the search is that ghini.pocket writes the date with time, location and the code looked for in a text file that can later be imported into the database.

For a greenhouse with around 1000 plants our estimates suggest you will need two days, working at relaxed pace, from 8:00 am to 5:00 pm.

After having imported the file generated by ghini.pocket, it is easy to reveal which plants are missing. For example: If we did the inventory of the INV3 from 4 to 5 September, this is the corresponding search:

```
plant where location.code = 'INV3' and not notes.note␣
↪like '2017090%'
```

All of these plants can be marked as dead, or lost, according to garden policy.

Visualizing the need of taxonomic attention

Our protocol includes one more detail intended to visually highlight plants that need the attention of a taxonomist.

A plant that only appears in our data base identified at family level or that wasn't yet the database receives a visual signal (e.g.: a wooden or plastic stick, for ice cream or french fries), to highlight that it is not identified. In this way the taxonomist in charge, when making a tour of the greenhouse can quickly spot them and possibly proceed to add their identification in the database.

- Naming convention in garden locations

Dettagli

code	description
CAC-B *x*	Reserved to cactus plants next to the orchids exposition glasshouses.
CRV:	Nepenthaceae exibition
IC-*xx*:	orquidearios de calor en el jardín (1A a 9C son lugares especificos entre del orquideario)
IF-xx:	orquidearios de frío en el jardín (1A a 5I son lugares específicos dentro del orquideario)
INV1:	invernadero 1 (calor)
INV2:	invernadero 2 (frío)
INV3:	invernadero 3 (calor)

- Adding an Accession for a Plant

  Obviously we keep increasing our collection, with plants coming from commercial sources, or collected from the wild, more rarely coming from expeditions to remote areas of our country, or we receive plants which were illegally collected.

  Sometimes we have to add plants to the digital collection, just because we have them physically, found in the garden, with or without its label, but without their digital counterpart.

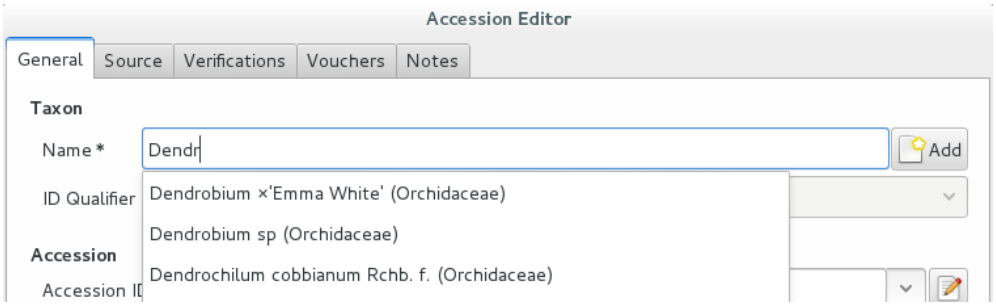
Existing plant, found in the garden with its own label

This activity starts with a plant, which was found at a specific garden location, an accession label, and the knowledge that the accession code is not in the database.

For this example, let's assume we are going to insert this information in the database.

Accessione	Specie	Collocazione
008440	*Dendrobium* ×"Emma White"	Invernadero 1 (calor)

We go straight into the Accession Editor, start typing the species name in the corresponding field. Luckily, the species was already in the database, otherwise we would use the **Add** button next to the entry field.



We select the correct species, and we fill in a couple more fields, leaving the rest to the default values:

ID delle Accessioni	Type of Material	Quantity	Provenance
008440	Pianta	1	Unknown

After this, we continue to the Plant editor, by clicking on **Add Plants**.

We do not fill in the Accession's «**Intended Locations**», because we don't know what was the original intention when the plant was first acquired.

In the Plant Editor, we insert the Quantity and the Location. And we're done.

The plant is now part of the database:

New accession: plant just entering the garden

This activity starts with a new Plant, just acquired from a known Source, a plant label, and an intended Location in the garden.

We mostly do the same as for the case that a plant is found in the garden, there are two differences: (1) we know the source of the plant; (2) acquiring this plant was a planned action, and we intend to place it at a specific location in the garden.

Again, we go straight into the Accession Editor, start typing the species and we either select if from the completion list or we add it on the fly.

ID delle Accessioni	Type of Material	Quantity	Fonte
033724	Pianta	1	specified

After this, we continue to the Plant editor, by clicking on **Add Plants**.

In the Plant Editor, we insert the Quantity and the Location.

Please note that the plant may be initially placed in a greenhouse, before it reaches its intended location in the garden.

Existing plant, found in the garden without its label

When this happens, we can't be sure the plant had never been in the collection, so we act as if we were re-labeling the plant. This is discussed in the next section, but we fall back to the case of a new accession.

- When we physically associate a label to a plant, there's always the chance that something happens either to the plant (it may die) or to the label (it may become unreadable), or to the association (they may be separated). We have software-aided protocols for these events.

We find a dead plant

Whenever a plant is found dead, we collect its label and put it in a box next to the main data insertion terminal, the box is marked "dead plants".

Definitely at least once a week, the box is emptied and the database is updated with this information.

Dead plants aren't *removed* from the database, they stay there but get a **quantity** zero. If the cause of death is known, this is also written in the database.

Please once again remember that a **Plant** is not an **Accession** and please remember we do not remove objects from the database, we just add to their history.

Insert the complete plant code (something like `012345.1`, or `2017.0001.3`, and you don't need leading zeros nor quotes), right click on the corresponding row, and click on **edit**. change the quantity to 0, fill in the reason and preferably also the date of change.

If you need add any details about the plant death, please use a **note**, and re-use the note category «death_cause».

Plants with **quantity** zero are shown with a different colour in the results view. This helps distinguish them from live plants.

We find a plant without a label

We can't be sure the plant had ever been in the collection or not. We assume it had, and that its label was lost.

Losing a plant label is unfortunate, but it just sometimes happens. What we do is to put a new label to the plant, and to clearly state that the label is a replacement of an original one.

We then handle the case as if it was a new accession, plus we add a note to the accession, category "label", text "relabeled".

- Keeping track of different sources of plant material

What different sources we can have

In this botanical garden, we receive plants from different types of origin. It could be from expeditions (plants coming from nature, collected with legal permission from MAE - Ecuadorian Environment Ministry), donated plants mostly coming as gifts from collectors or orchid commercialization enterprises, purchased, or confiscated plants (usually coming from MAE raids around the country).

If the plant comes from a wild source

The accession editor offers the option «origin» option. When a plant is traceable to a wild source, we can specified its specific origin. We want to com-

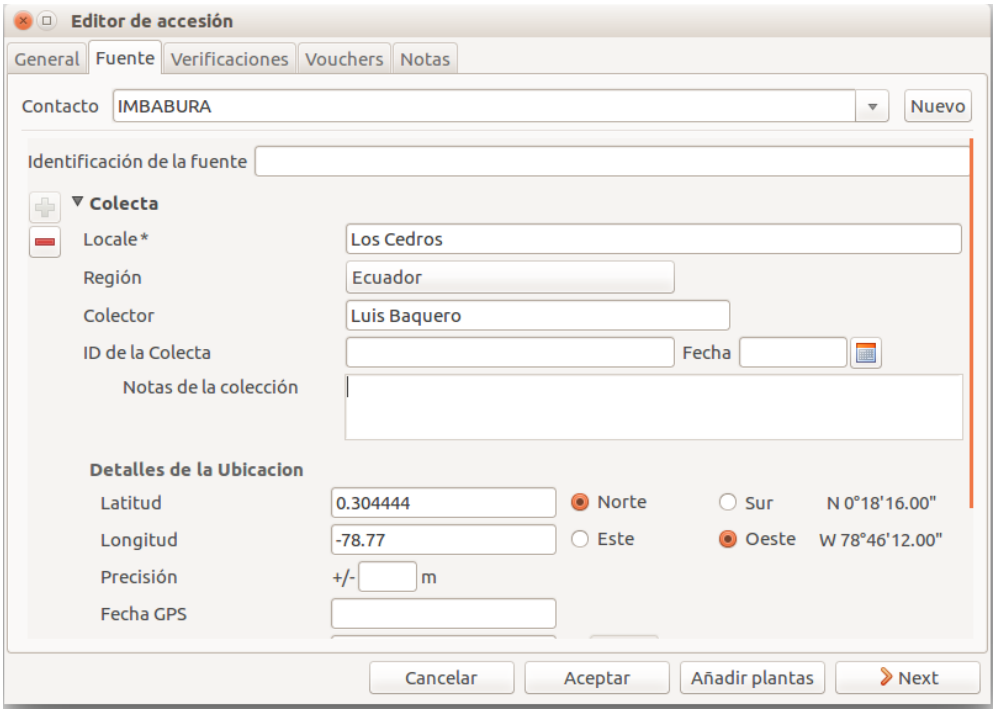
ply with ITF2, and ghini-1.0 only partly respects that standard. The ITF2 complying options are:

- Wild: Accession of wild source.
- Cultivated: Propagule(s) from a wild source plant.
- Not Wild: Accession not traceable to a wild source.
- Insufficient data

In the case of a donated plant, it is better to put detail information just as a note in the plant accession; in the case of a plant with an unknown origin, we select the Insufficient data option.

Using the source tab in the accession editor

In this section we can create or use a contact, our source of plant material. It could be from an expedition to a collecting place, and in this case we would specify the region and the expedition name, or could be the name of the person or enterprise donating a specific batch of plants.



Once you choose or create the contact information, this section deploys more options, here you can specify the region, where you can choose the country of origin, and a specific location within the region, georeferencing information (including the GPS data), habitat description collector name. For the last one, I recommend also to write the specific date next to the collector name (eg. Luis Baquero 11/10/2016).

Donated, bought or confiscated plants

However useful for expeditions or for donors where the main information is geographic, this source tab is not very practical in our remaining cases: we handle three more categories: confiscated, purchased and donated, for these categories the options available in the source tab do not apply: too much information and not to the point.

In these cases, we add a set of notes, according to the case.

— Donated plants

If the plant was donated by individual, we add the individual among our contacts and specify it as source, then we add the notes:

category	text
source-type	gift
source-detail	Contribución científica al JBQ

— Bought plants

If the plant was bought, we add the previous owner among our contacts and specify it as source, then we add the notes:

category	text
source-type	acquisto
source-detail	optional, free text
factura	the invoice number

— Confiscated plants

If the plant was confiscated, we add the previous owner among our contacts and specify it as source, then we add the notes:

category	text
source-type	confiscated
source-detail	possibly, legal details, law number . . .

- Produzione o riproduzione di etichette

Rinnovare le etichette

Sometimes we refresh the labels, for example all that is in a greenhouse, or maybe just a set of plants because their labels risk becoming unreadable.

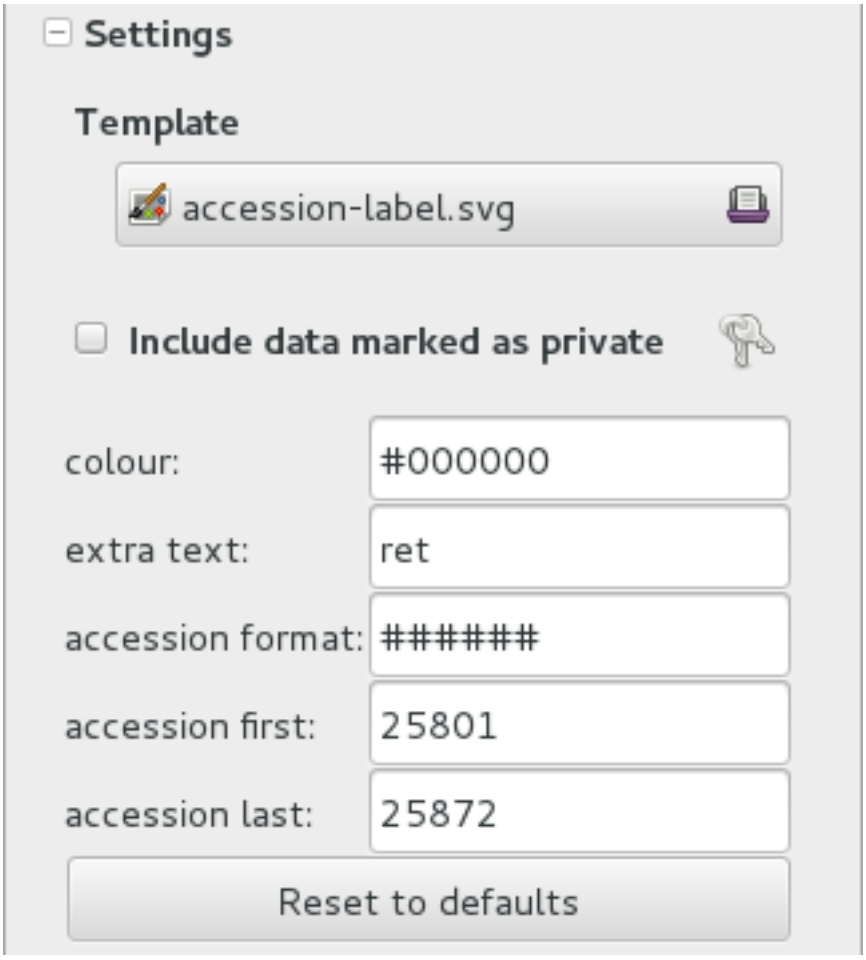
In the first case it's easy selecting all plants in the Location, we just type the location name, or give the search `location like <location name>`.

The second case it's a bit trickier. What we do is to create a temporary **Tag**, and use it to tag all plants that were found in need for a new label.

Given the selection, we start the report tool, using the mako `accession-label.svg` template. We reset its options to default values, and since we're using a simple printer, we set the colour to `black` instead of `blue`, which is meant for engraving.

Preparing labels for non-database plants

To prepare the batch of 72 labels, we use a mako report template, named `accession-label.svg`. This template accepts parameters, this is an example that would produce labels from 025801 all the way to 025872.



Labels come for us in two flavours: (1) either new plants just being acquired by the garden; (2) or plants in the garden, found without a label. We distinguish the two cases by adding a "ret" extra text for relabeled plants.

We keep two boxes with labels of the two types, ready to be used.

- Our garden has two exposition greenhouses, and several warm and cold greenhouses where we keep the largest part of our collection. Plants are moved to the exposition when flowering and back to the «warehouse» when less interesting for the exposition. For each plant in our collection we need to know its current locations and history of movements.

Planned action

The action starts by moving the plants around, and collecting the plant code either on paper, or in our mobile app, if we had one.

We then go to the desktop terminal and revise all plants one by one changing their location in the database. It is important that the date of the location change is correctly memorized, because this tells us how long a plant stays in the exposition.

If we had a mobile app, we would just upload the info to the server and we would be done.

Ex-post correction

While revising the garden, we find a plant at a location that is not what the database says. We update the database information.

For example, the plant belonging to accession "012142", species "*Acineta* sp", was found in "Invernadero 1", while the database says it is in "ICAlm3".

All we do is find the Plant in the database and update its information. We do not change anything in the initial Accession information, just the current Plant information.

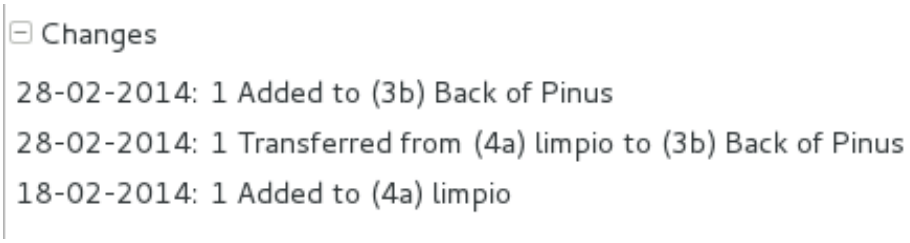
We type the accession code in the search entry field, with quotes, hit enter. The search results now shows the accession, and it tells us how many plants belong to it. Click on the squared **+** in the results row, so we now also see a row for the plant belonging to the accession.

Right click on the Plant row, the three options will show: "Edit, Split, Delete", select Edit, you land in the Plant Editor.

Just correct the Location field, and click on OK.

The InfoBox contains information about the last change to the object:

For plants, even more interesting, it builds a history of changes, list that includes Location changes, or Quantity changes.



- As plants enter the flowering stage, we can review their identification directly, or we take pictures of details of the flower, hoping that a visiting specialist could help completing the identification.

Fotografie

We are practicing with ODK Collect, a small program running on hand-held android devices. Ghini's use of ODK Collect hasn't yet frozen to a best practice. Do have a look at the corresponding issue on github.

- Regularly, we need producing reports about our collection that the Ecuadorian Environment Ministry (MAE) requires and that justify the very existence of the garden.

Generare relazioni

Each year the botanic garden has to submit a report (annual report of management and maintenance of orchids collection) complying to the requirements of the Ecuadorian Ministry of the Environment.

To this end, we start selecting the plants we have to include in the report. It might be all acquisition in the past year:

```
accession where _created between |datetime|2017,1,1|␣
↪and |datetime|2018,1,1|
```

or all plants within a location, or all plants belonging to a species, or just everything (but this will take time):

```
plant where location = 'abc'
plant where accession.species.epithet='muricata' and␣
 ↪accession.species.genus.epithet='Annona'
plant like %
```

Having selected the database objects which we want in the report, we start the report tool, which acts on the selection.

Ricerca nel database

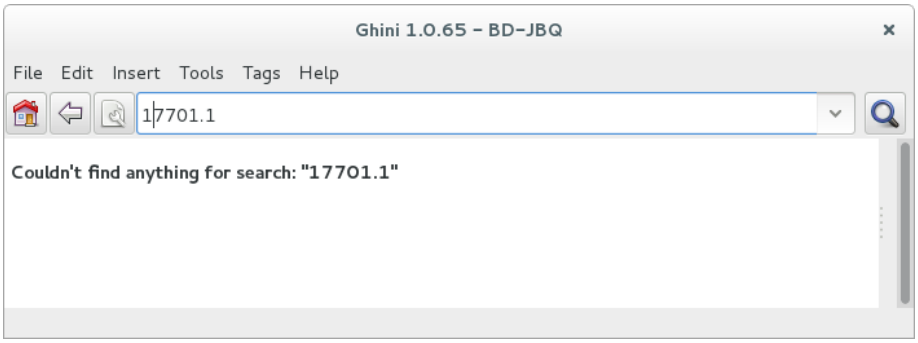
You search the database in order to edit the data further, or because you want to produce a report. Anyway you start with typing something in the search field

and you hope to see your result in the search result view.

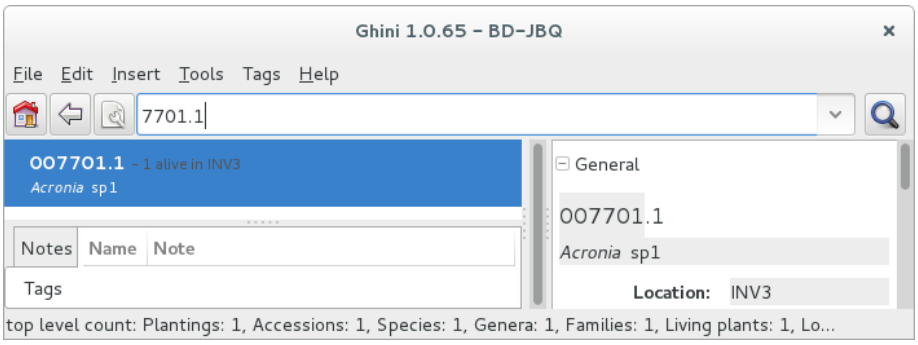
Ricerca al fine di modificare (pianta o accessione)

When searching in order to edit, you want to be very specific, and select as few objects as possible. The most fine-tuned search is the one based on plant number: you know the code, you get one object.

If your plant is not there, the screen would look like this:



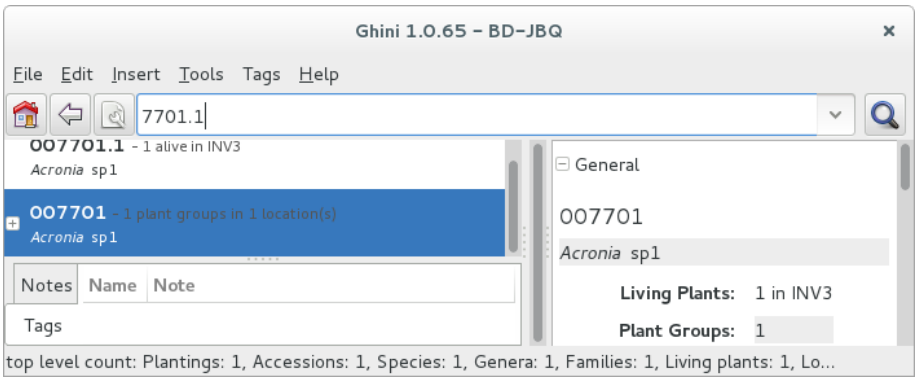
Other example, plant `007701.1` is in the database:



All fields with a darker background in the infobox on the right hand side are hyperlinks to other objects in the database. Clicking on them will

either replace the text in the search field and execute the query, or will simply add the object to the results.

Clicking on the accession does the latter.



We now have both Plant or Accession in the search result view and we can now edit either or both.

Ricerca al fine di modificare (pianta o accessione)

When searching in order to create a report, you want to be both specific (you don't want to report about irrelevant objects) and broad (you don't want to report about a single object).

Sometimes the report itself suggests the query, as for example: all plants in greenhouse 3; or: all plants belonging to endangered species (we store this information in a note associated to the species); or: all plants added to the collection this year;

```
plant where location.code = INV3
plant where accession.species.notes.note="endangered
↪"
plant where accession._created > |datetime|2017,1,1|
```

Otherwise a flexible way to achieve this is to work with **Tags**.

Using Tags as enhanced searching

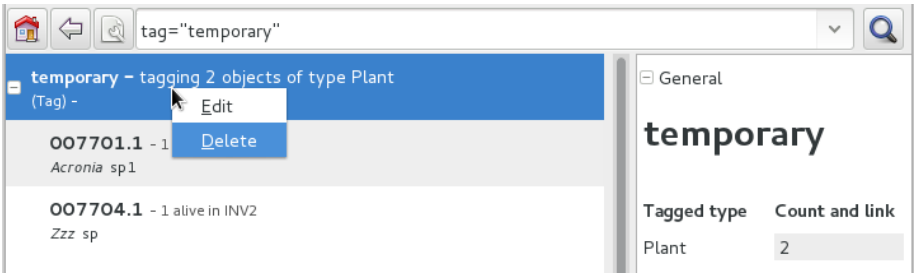
Sometimes we have to take the same action on objects of the same type, but we don't manage to quickly think of a search query that would group all that we need and exclude all we do not need.

This is one possible use of **Tags**. We start with a selection, we tag all objects in the selection under a new temporary tag. Let's say we call it «temporary».

We continue searching and adding objects to the temporary tag until the tag identifies all that we need.

Finally from the Tags menu we select the one we just created (in our example this corresponds to the search `tag="temporary"`) and we can invoke the report.

When we're done with a temporary tag, there's no point in leaving it around, so we just delete it.



Be aware of the available search strategies

This is nicely documented, «più non dimandare» and read the docs.

4.1.2 using ghini for a seed database

We keep getting involved in groups focusing on endagered plant seeds. They want to note down when seeds come in, but also when they go out to people that order the seed.

In ghini, we keep speaking of ›Plants‹, ›Locations‹, while such user groups focus on ›Seeds‹ and ›Jars‹ and ›Drawers‹ and ›Boxes‹ and ›Envelopes‹. So people wonder whether ghini could be adapted to their use case, or for directions on how to develop their own database.

Does ghini need being adapted for such a seed database?

no it doesn't need any adaptation, it's just that you need to read some of its terms differently.

the taxonomy part is just taxonomy, plant species information, no need to explain that, no way to interpret it otherwise.

›Accessions‹ and ›Plants‹, you know what an ›Accession‹ is, but since you're consistently handling ›Plants‹ still only in seed form, the Wikipedia explanation of an accession sounds like this: it is a seed or group of seeds that are of the same taxon, are of the same propagule type (or treatment), were received from the same source, were received at the same time.

If you hold seeds in jars, or in other sort of containers that is able to hold hundreds of seeds, please make sure that a jar contains seeds of just one accession, as above described: same taxon, same treatment, same source, same time.

Each one of your ›Jars‹ of seeds is in ghini speak a ›Plant‹, and the amount of seeds in the ›Jar‹ is the ›Plant‹ ›quantity‹. An ›Envelope‹ is just the same as a ›Jar‹: a container of seeds from the same ›Accession‹, just presumably smaller.

A ›Box‹ (where you keep several ›Envelopes‹) or a ›Drawer‹ (where you keep several ›Jars‹) are in ghini speak a ›Location‹.

Since a ›Jar‹ or an ›Envelope‹ contains seeds from an ›Accession‹, you will clearly label it with its ›Accession‹ code (and trailing ›Plant‹ number). You might write the amount of seeds, too, but this would be repeating information from the database, and repeating information introduces an inconsistency risk factor.

How do I handle receiving a batch of seeds?

Nota: When we receive seeds, we either collect them ourselves, or we receive it from an other seed collector. We handle receiving them possibly on the spot, or with a small delay. Even when handled together with several other batches of seeds we received, each batch keeps its individuality.

We want to be later able to find back, for example, how many seeds we still have from a specific batch, or when we last received seeds from a specific source.

As long as you put this information in the database, as long as you follow the same convention when doing so, you will be able to write and execute such queries using ghini.

One possibility, the one described here, is based on ›Notes‹. (Ghini does not, as yet, implement the concept «Acquisition». There is an issue related to the Acquisition and Donation objects, but we haven't quite formalized things yet.)

You surely already use codes to identify a batch of seeds entering the seed bank. Just copy this code in a ›Note‹, category "received", to each ›Accession‹ in the received batch. This will let you select the ›Accessions‹ by the query:

```
accession where notes[category='received'].note='<your code>
↪'
```

Use the "Source" tab if you think so, it offers space for indicating an external source, or an expedition. When receiving from an external source, you can specify the code internal to their organization. This will be useful when requesting an extra batch.

How do I handle sending seeds?

what you physically do is to grab the desired amount of seeds of the indicated species from a jar, put it in an envelope and send it. what you do from a point of view of the database is exactly the same, but precisely described in a protocol:

- Use the database to identify the ›Jar‹ containing the desired amount of the right seeds.
- remove that amount of seeds from the ›Jar‹ (decrement the quantity),
- put the seeds in an ›Envelope‹ (yes, that's a database object).
- send the envelope (but keep it in the database).

this in short.

When I send seeds, it's not just one bag, how does ghini help me keeping things together?

There's two levels of keeping things together: one is while you're preparing the sending, and then for later reference.

While preparing the sending, we advise you use a temporary ›Tag‹ on the objects being edited.

For later reference, you will have common ›Note‹ texts, to identify received and sent batches.

Can you give a complete example?

Right. Quite fair. Let's see...

Say you were requested to deliver 50 seeds of Parnassia palustris, 30 of Gentiana pneumonanthe, 80 of Fritillaria meleagris, and 30 of Hypericum pulchrum.

step 1

The first step is to check the quantities you have in house, and if you do have enough, where you have them. You do this per requested species:

```
accession where species.genus.epithet=Parnassia and species.
↪epithet=palustris and sum(plants.quantity)>0
```

Expand in the results pane the ›Accession‹ from which you want to grab the seeds, so you see the corresponding ›Jars‹, highlight one, and tag it with a new ›Tag‹. To do this the first time, go through the steps, just once, of creating a new ›Tag‹. The new tag becomes the active tag, and subsequent tagging will be speedier. I would call the tag ›sending‹, but that's only for ease of exposition and further completely irrelevant.

Repeat the task for Gentiana pneumonanthe, Fritillaria meleagris, Hypericum pulchrum:

```
accession where species.genus.epithet=Gentiana and species.
↪epithet=pneumonanthe and sum(plants.quantity)>0
accession where species.genus.epithet=Fritillaria and␣
↪species.epithet=meleagris and sum(plants.quantity)>0
accession where species.genus.epithet=Hypericum and species.
↪epithet=pulchrum and sum(plants.quantity)>0
```

Again highilight the accession from which you can grab seeds, and hit Ctrl-Y (this tags the highighted row with the active tag). Don't worry if nothing seems to happen when you hit Ctrl-Y, this is a silent operation.

step 2

Now we prepare to go to the seeds bank, with the envelopes we want to fill.

Select the ›sending‹ ›Tag‹ from the tags menu, this will bring back in the results pane all the tagged ›Plants‹ (›Jars‹ or ›Envelopes‹), and will tell you in which ›Location‹ (›Drawer‹ or ›Box‹) they are to be found. Write this information on each of your physical envelopes. Write also the ›Species‹ name, and the quantity you can provide.

Walk now to your seeds bank and, for each of the envelopes you just prepared, open the ›Location‹, grab the ›Plant‹, extract the correct amount of seeds, put them in your physical envelope.

And back to the database!

step 3

If nobody used your workstation, you still have the Tag in the results pane, and it's expanded so you see all the individual plants you tagged.

One by one, you have to ›split‹ the plant. This is a standard operation that you activate by right-clicking on the plant.

A plant editor window comes in view, in "split mode".

Splitting a plant lets you create a database image of the plant group you just physically created, eg: it lets you subtract 30 items from the Gentiana pneumonanthe plant (group number one, that is the one in the jar), and create a new plant group for the same accession. A good practice would be to specify as ›Location‹ for this new plant the "out box", that is, the envelope is on its way to leave the garden.

Don't forget to delete the temporary "sending" ›Tag‹.

step 4

Final step, it represents the physical step of sending the envelope, possibly together with several other envelopes, in a single sending, which should have a code.

Just as you did when you received a batch of plants, you work with notes, this time the category is "sent", and the note text is whatever you normally do to identify a sending. So suppose you're doing a second sending to Pino in 2018, you add the note to each of the newly created envelopes: category "sent", text: "2018-pino-002".

When you finally do send the envelopes, these stop being part of your collection. You still want to know that they have existed, but you do not want to count them among the seeds that are available to you.

Bring back all the plants in the sending "2018-pino-002":

```
plant where notes[category='sent'].note = '2018-pino-002'
```

You now need to edit them one by one, mark the ›quantity‹ to zero, and optionally specify the reason of the change, which would be ›given away‹, and the recipient is already specified in the "sent" ›Note‹.

This last operation could be automated, we're thinking of it, it would become a script, acting on a selection. Stay tuned.

CAPITOLO 5

Amministrazione

5.1 Amministrazione di basi di dati

Nel caso si stia utilizzando un vero e proprio DBMS (sistema di gestione di basi di dati) per contenere le collezioni di Ghini, è importante prendere in considerazione l'amministrazione di questo DBMS. Una descrizione del compito di amministrare una base dati è qui assolutamente fuori luogo, ma è importante che l'utente sia consapevole del problema.

5.1.1 SQLite

SQLite offre una soluzione in quanto SQLite non è esattamente quanto si potrebbe definire un DMBS: ogni base dati SQLite è un file, farne copia di emergenza (backup) sarà sufficiente. Se si è creata la connessione alla base dati SQLite accettando i valori per difetto, il file relativo alla connessione si trova nella directory `~/.bauble/` (con Windows bisognerà trovare la `AppData`).

In Windows è da qualche parte nella directory `AppData`, molto probabilmente in `AppData\Roaming\Bauble`. Tenete a mente che Windows fa del suo meglio per nascondere la struttura `AppData` agli utenti normali.

Il modo più veloce per aprirlo è con *Esplora Risorse*: inserisci `%APPDATA%` e premi INVIO.

5.1.2 MySQL

Please refer to the official documentation.

Backing up and restoring databases is described in breadth and depth starting at this page.

5.1.3 PostgreSQL

Please refer to the official documentation. A very thorough discussion of your backup options starts at chapter 24.

5.2 Configurazione di Ghini

Ghini utilizza un file di configurazione per memorizzare i valori tra le chiamate. Questo file è associato all'account dell'utente, così ogni utente avrà il suo file di configurazione.

Per esaminare il contenuto del file di configurazione Ghini, digitare `:prefs` nell'area di immissione testo dove normalmente tipo le ricerche, quindi premere INVIO.

Normalmente non è necessario modificare il file di configurazione, ma è possibile farlo con un programma di editor di testo normale. File di configurazione di Ghini è il percorso predefinito per i database SQLite.

5.3 Prospetto errori

Se notate qualcosa di imprevisto nel comportamento di Ghini, si prega di considerare un problema di deposito sul sito di sviluppo Ghini.

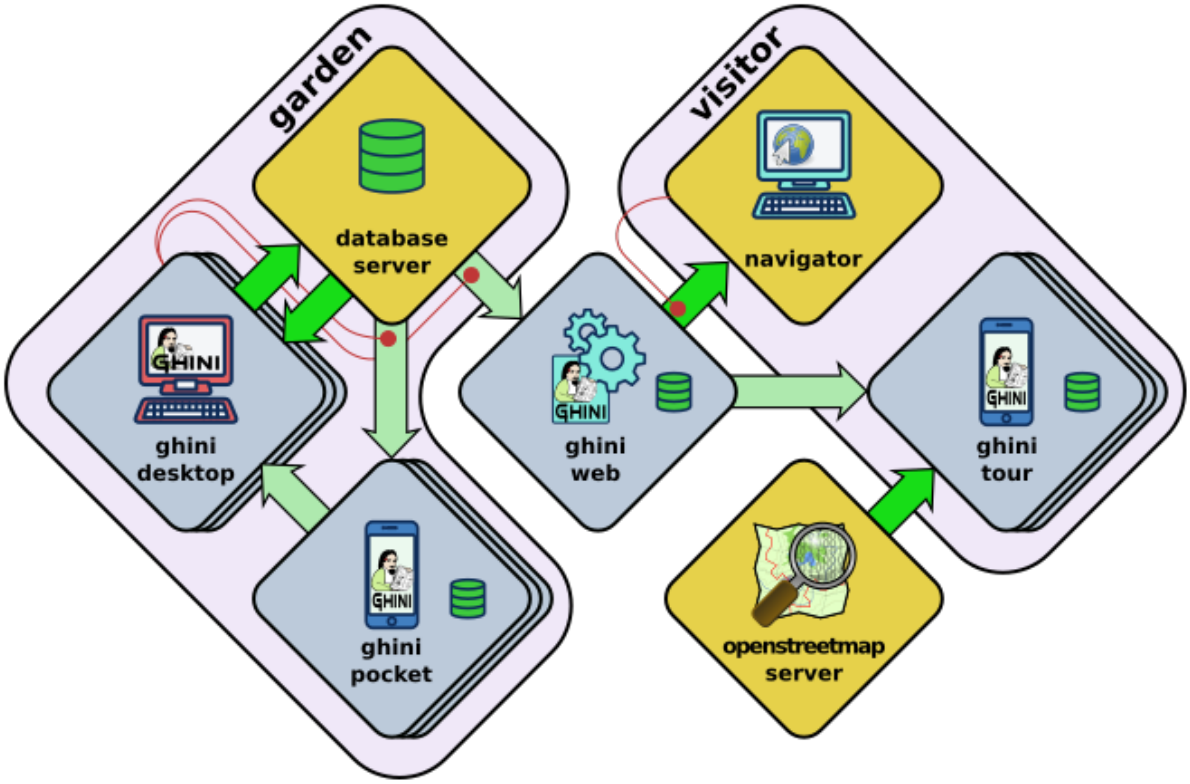
Ghini lo sviluppo del sito sono accessibili tramite il menu di aiuto.

CAPITOLO 6

Ghini Family

6.1 the Ghini family

Let's start by recalling the composition of the Ghini family, as shown in the diagram:



You have learned how to use ghini.desktop, here we introduce the other members of the family, and their interaction.

6.1.1 ghini.pocket

ghini.pocket is an Android app which you can install from the play store. ghini.pocket is definitely the tool you will use most, next to ghini.desktop.

With ghini.pocket you always have the latest snapshot of your database with you.

Type an accession number, or scan its barcode or QR label, and you know:

- the identification of the plant,
- whether it already has pictures,
- when it entered the garden and
- from which source.

Apart as a quick data viewer, you can use ghini.pocket for...

---

data correction

If by your judgement, some of the information is incorrect, or if the plant is flowering and you want to immediately take a picture and store it in the database, you do not need take notes on paper, nor follow convolute procedures: ghini.pocket lets you write your corrections in a log file, take pictures associated to the plant, and you will import this information straight into the database, with further minimal user intervention.

---

inventory review

The initial idea on which we based ghini.pocket is still one of its functionalities: inventory review.

Using ghini.pocket, reviewing the inventory of a greenhouse, in particular if you have QR codes on plant labels, goes as fast as you can walk: simply enter the location code of your greenhouse, reset the log, then one by one scan the plant codes of the plants in the greenhouse. No further data collection action is required.

When you're done, import the log in ghini.desktop. The procedure available in ghini.desktop includes adding unknown but labelled plants in the database, marking as lost/dead all plants that the database reports as alive and present in the inventoried location, but were not found during the inventory.

---

taxonomic support

As a bonus, ghini.pocket contains a phonetic genus search, and a quite complete database of botanic taxa with rank between order and genus, including tribes, and synonymies.

check further *data streams between software components*.

6.1.2 ghini.web

ghini.web is a web server, written in nodejs.

Its most visible part runs at http://gardens.ghini.me and shows as a map of the world, where you browse gardens and search their published collection.

It also serves configuration data to ghini.tour instances.

check further *data streams between software components*.

6.1.3 ghini.tour

ghini.tour is an Android app which you can install from the play store.

People visiting your garden will install ghini.tour on their phone or tablet, enjoy having a map of the garden, knowing where they are, and will be able to listen to audio files that you have placed as virtual information panels in strategic spots in your garden.

world view

at startup, you see the world and gardens. select a garden, and enter.

garden view

when viewing at garden level, you see panels. select a panel, and listen.

check further *data streams between software components*.

6.1.4 data streams between software components

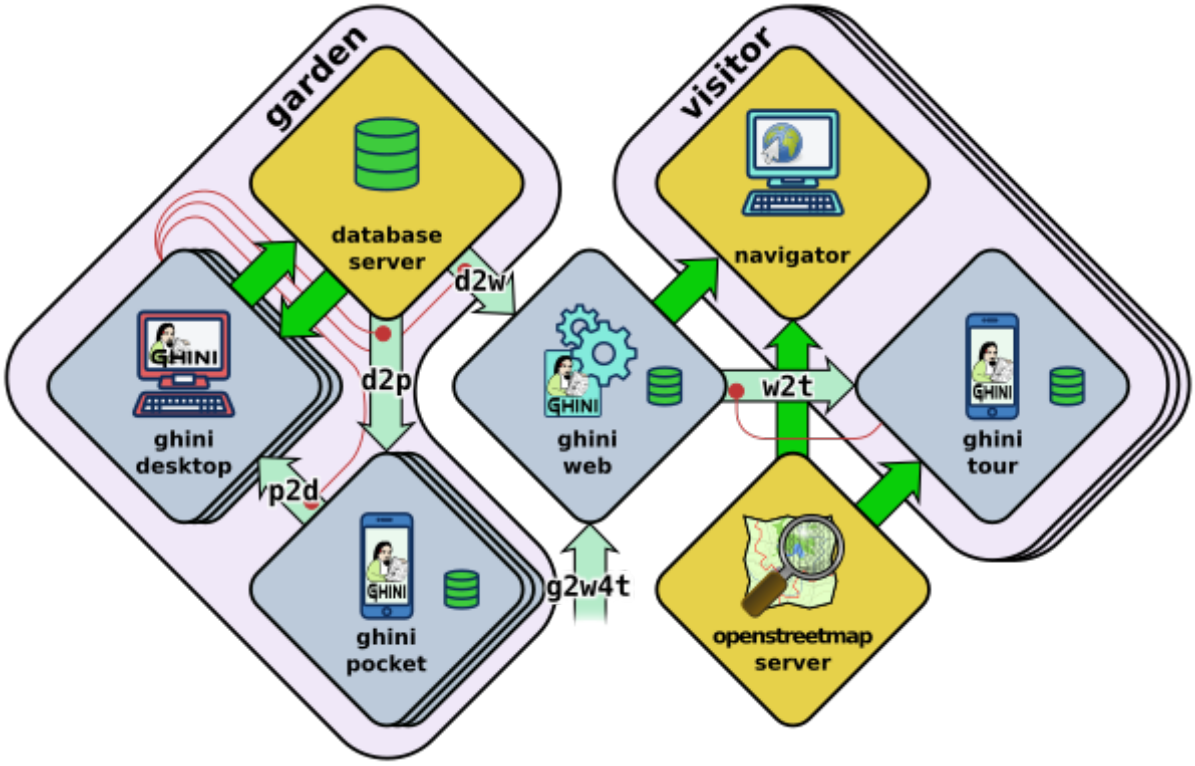
Nota: This section contains technical information for database managers and software developers.

In the diagram showing the composition of the Ghini family, the alert reader noticed how different arrows representing different data flows, had different colours: some are deep green, some have a lighter tint.

Deeper green streams are constant flows of data, representing the core activity of a component, eg: the interaction between ghini.desktop and its database server, or your internet browser and ghini.web.

Lighter green streams are import/export actions, initiated by the user at the command panel of ghini.desktop, or in the ghini.tour settings page.

This is the same graph, in which all import data streams have been given an identifier.



d2p: copy a snapshot of the desktop database to ghini.pocket

- export the desktop database to a pocket snapshot
- copy the snapshot to the handheld device

ghini.pocket integrates closely with ghini.desktop, and it's not a tool for the casual nor the external user. One task of your garden database manager is to regularly copy an updated database snapshot to your Android device.

We advise enabling USB debugging on the device. In perspective, this will allow ghini.desktop writing directily into the ghini.pocket device.

Export the file from ghini.desktop, call the file pocket.db, copy it to the phone:

```
adb -d push /tmp/pocket.db /sdcard/Android/data/me.ghini.
↪pocket/files/
```

The above location is valid even if your phone does not have a memory card.

Other options include bluetooth, or whatever other way you normally use to copy regular files into your Android device.

---

p2d: import from the ghini.pocket log file and pictures into the central database

even if we're still calling it "inventory log", ghini.pocket's log contains more than just inventory corrections.

- produce a log on the handheld device
- import the log in the desktop database

first of all, copy the collected information from ghini.pocket into your computer:

```
export DIR=/some/directory/on/your/computer
adb -d pull /sdcard/Android/data/me.ghini.pocket/files/
↪searches.txt $DIR
adb -d pull -a /sdcard/Android/data/me.ghini.pocket/files/
↪Pictures $DIR
```

then use ghini.desktop to import this information into your database.

---

d2w: send a selection of your garden data to ghini.web

Offer a selection of your garden data to a central ghini.web site, so online virtual visitors can browse it. This includes plant identification and their geographic location.

content of this flow:

- garden: coords, name, zoom level (for initial view)
- plants: coords, identification, zoom level (for visibility)
- species: binomial, phonetic approximation

---

g2w: add geographic non-botanic data to ghini.web

- Write geographic information about non-botanic data (ie: point of interest within the garden, required by ghini.tour) in the central ghini.web site.

  content of this flow:

  - virtual panels: coords, title, audio file
  - photos: coords, title, picture

  virtual panels don't necessarily have an associated photo, photos don't necessarily have an associated audio file.

---

w2t: importing locations and POIs from ghini.web to tour

content of this flow:

- Garden (coords, name, zoom level)
- Points of Interest (coords, title, audio file, photo)

CAPITOLO 7

Sviluppo di Ghini

7.1 Manuale dello sviluppatore

If you ran the `devinstall` installation instructions, you have downloaded the sources, connected to the github repository. You are in the ideal situation to start looking into the software, understand how it works, contribute to ghini.desktop's development.

7.1.1 Sostieni lo sviluppo di Ghini

Se volete contribuire a Ghini, può farlo in diversi modi:

- Utilizzare il software, notare cosa piace di meno e potrebbe essere corretto, ‘‘segnalarlo come problema <http://github.com/ghini/ghini.desktop/issues/new>‘_ separatamente. Uno sviluppatore reagirà prima di quanto si possa immaginare.
- Se avete un'idea di ciò che manca nel software ma non abbastanza e formalizzare in questioni separate, si potrebbe prendere in considerazione l'assunzione di un professionista. Questo è il modo migliore per assicurarsi che qualcosa accade rapidamente su Ghini. Assicurarsi che lo sviluppatore apra problemi su github e vi pubblichi il contributi.
- Traduci! Qualsiasi aiuto con traduzione sarà gradito, così si prega di tradurre! è possibile farlo senza installare nulla sul tuo computer, usando solo il servizio di traduzione on-line offerti da http://hosted.weblate.org/
- "Fork the repository", scegliere un problema, risolverlo, aprire una richiesta "pull request". Vedere il *flusso di lavoro per risolvere bug* qui sotto.

Se non hai ancora installato Ghini e desideri dare un'occhiata alla storia del suo codice, è possibile aprire la nostra pagina progetto github e vedere tutto ciò che s'è fatto intorno a Ghini fin dalla sua nascita nel 2004 come Bauble.

If you install the software according to the `devinstall` instructions, you have the whole history in your local git clone.

7.1.2 Fonte del software, versioni, rami

Se si desidera che una particolare versione di Ghini, abbiamo rilasciare e gestire versioni come rami. Si dovrebbe effettuare il `git checkout` del ramo corrispondente alla versione di vostra scelta.

linea di produzione

Nomi dei rami per le versioni stabili (produzione) di Ghini sono nella forma `ghini-x.y` (ad esempio: ghini-1.0); nomi dei rami dove sono pubblicate le versioni di prova di Ghini sono nella forma `ghini-x.y-dev` (ad esempio: ghini-1.0-dev).

7.1.3 Flusso di sviluppo

Il nostro flusso di lavoro è di impegnarsi continuamente per la branca di test, spesso di spingerli a github, di lasciare travis-ci e coveralls.io di controllare la qualità dei rami test spinti, infine, di volta in volta, per unire il ramo testing la versione corrispondente.

Quando si lavora a problemi più grandi, che sembrano richiedere più di un paio di giorni, io potrei aprire una filiale connessa alla questione. Non lo faccio molto spesso.

problemi più importanti

Quando affronta un problema più grande singolo, creare un tag di ramo all'estremità di una linea di sviluppo principale (ad es.: `ghini-1.0-dev`) e seguire il flusso di lavoro descritto al

https://git-scm.com/book/en/v2/Git-Branching-Basic-Branching-and-Merging

In breve:

```
git up
git checkout -b issue-xxxx
git push origin issue-xxxx
```

Lavorare sul nuovo ramo temporaneo. Quando è pronto, andare a github, unire il ramo con la linea di sviluppo principale da cui diramano, risolvere i conflitti, ove necessario, eliminare il ramo temporaneo.

Quando si è pronti per la pubblicazione, è possibile unire la linea di sviluppo nella linea di produzione corrispondente.

7.1.4 Aggiornando l'insieme di stringhe traducibili

Di tanto in tanto, durante il processo di aggiornamento del software, si verrà essere aggiungendo o modificando le stringhe nelle fonti python, nella documentazione, nelle fonti glade. La maggior parte delle nostre corde sono traducibile e sono offerti a weblate per le persone a contribuire, sotto forma di diversi file. `po`.

Un file `po` è specifico di una lingua ed è composto da coppie di testi, in lingua originale e la corrispondente traduzione. Quando un traduttore aggiunge una traduzione su weblate, questa raggiunge il nostro repositorio github. Quanto un programmatore aggiunge una stringa nel software, questa raggiunge weblate per venir tradotta.

Weblate ospita il progetto Ghini. Il progetto è suddiviso in componenti, ciascuna corrispondente ad un branch di un nostro repositorio github. Ciascuna componente accetta traduzioni in diverse lingue.

componente	repositorio	branch
Desktop 1.0	ghini.desktop	ghini-1.0-dev
Desktop 3.1	ghini.desktop	ghini-3.1-dev
Documentation 1.0	ghini.desktop-docs.i18n	ghini-1.0-dev
Documentation 3.1	ghini.desktop-docs.i18n	ghini-3.1-dev
Web 1.2	ghini.web	master
Pocket	ghini.pocket	master
Tour	ghini.tour	master

Per aggiornare i file `po` relativi al *software*, esegui il seguente script dalla linea di comando, preferibilmente dalla base del checkout di *ghini.desktop*:

```
./scripts/i18n.sh
```

Per aggiornare i file `po` relativi alla *documentazione*, esegui il seguente script dalla linea di comando, preferibilmente dalla base del checkout di *ghini.desktop-docs.i18n*:

```
./doc/runme.sh
```

Quando si esegue uno degli script di cui sopra, è probabile che si abbia bisogno di fare il commit di *tutti* i file *po* del progetto. Si potrebbero voler esaminare le modifiche prima di inserirle nel respository. Questo è più importante quando si esegue una correzione marginale di una stringa, come la rimozione di un errore di battitura.

Qualcosa che accade: trovarsi con un conflitto. Risoluzione dei conflitti non è difficile se sai come si fa. Prima di tutto, aggiungere weblate come remoto:

```
git remote add weblate-doc10 https://hosted.weblate.org/git/ghini/
↪documentation-10/
```

Quindi assicurarsi che siamo nel repository corretto, sul ramo corretto, aggiornare la fonte remota, unirla al ramo:

```
git checkout ghini-1.0-dev
git remote update
git merge weblate-doc10/ghini-1.0-dev
```

La nostra documentazione su readthedocs ha una versione originale in lingua inglese, e diverse traduzioni. Seguiamo semplicemente la descrizione della localizzazione. Qui non c'è nulla che abbiamo inventato noi.

ReadTheDocs controlla l'impostazione «Language» del progetto ed invoca `sphinx-intl` per produrre la documentazione formattata nella lingua obiettivo. Con la configurazione predefinita — che non abbiamo alterato — `sphinx-intl` prevede un file `po` per ogni documento di origine, che abbia lo stesso nome del documento di origine, e che risieda nella directory `locale/$(LANG)/LC_MESSAGES/`.

D'altra parte, Weblate (e noi stessi) preferiamo un file `po` per ogni lingua, e tenerli tutti nella stessa cartella `/po`, come facciamo per il progetto software: `/po/$(LANG).po`.

Per non ripetere informazione e per lasciare lavorare entrambi i sistemi nel loro modo naturale, abbiamo due insiemi di link simbolici (git li rispetta).

Per riassumere: quando viene aggiornato un file nella documentazione, lo script‘‘runme.sh‘‘ si occupa di:

1. copiare i file `rst` dal software alla documentazione;
2. creare un nuovo file `pot` per ciascun elemento di documentazione;
3. unire tutti i file `pot` in un unico `doc.pot`;
4. aggiornare tutti i `doc.po` (uno per lingua) sulla base del nuovo `doc.pot`;
5. creare tutti i collegamenti simbolici:
    a. quelli utilizzati da `sphinx-intl` in `/local/$(LANG)/LC_MESSAGES/`
    b. quelli utilizzati da noi e weblate in `/po/$(LANG).po`

Sicuramente potremmo scrivere tutto ciò in un Makefile, o ancora meglio includerlo nel “/ doc/Makefile “. Chissà, forse lo faremo, un giorno.

7.1.5 Producing the docs locally

The above description is about how we help external sites produce our documentation so that it is online for all to see. But what if you want to have the documentation locally, for example if you want to edit and review before pushing your commits to the cloud?

In order to run sphinx locally, you need to install it **within** the same virtual environment as ghini, and to install it there, you need to have a sphinx version whose dependencies don not conflict with ghini.desktop's dependecies.

What we do to keep this in order?

We state this extra dependency in the `setup.py` file, as an `extras_require` entry. Create and activate the virtual environment, then run `easy_install ghini.desktop[docs]`. This gets you the sphinx version as declared in the `setup.py` file.

If all you want is the html documentation built locally, run `./setup.py install docs`. For more options, enter the `doc` directory and run `make`.

7.1.6 Which way do the translated strings reach our users?

A new translator asked the question, adding: »Is this an automated process from Weblate –> GIT –> Ghini Desktop installed on users computers, or does this require manual steps?

The aswer is that the whole interaction is quite complex, and it depends on the component.

When you install `ghini.desktop` or one of the Android apps, the installation doesn't assume a specific run-time language: a user can change their language configuration any time. So what we do is to install the software in English together with a translation table from English to whatever else.

At run-time the GUI libraries (Android or GTK) know where to look for the translation strings. These translation tables are generated during the installation or upgrade process, based on the strings you see on Weblate.

The path followed by translations is: You edit strings on Weblate, Weblate keeps accumulating them until you are done, or you don't interact with Weblate for a longer while; Weblate pushes the strings to github, directly into the development line `ghini-1.0-dev`; I see them and I might blindly trust or prefer to review them, maybe I look them up in wikipedia or get them translated back to Italian, Spanish or English by some automatic translation service; sometimes I need to solve conflicts arising because of changed context, not too often fortunately. As said, this lands in the development line `ghini-1.0-dev`, which I regularly publish to the production line `ghini-1.0`, and this is the moment when the new translations finally make it to the distributed software.

Users will notice a *new version available* warning and can decide to ignore it, or to update.

For `ghini.pocket`, it is similar, but the notification is handled by the Android system. We publish on the Play Store, and depending on your settings, your phone will update the software automatically, or only notify you, or do nothing. It depends on how you configured automatic updates.

For `ghini.web`, we haven't yet defined how to distribute it.

For ghini's documentation, it's completely automatic, and all is handled by readthedocs.org.

7.1.7 Aggiunta mancante unit test

Se siete interessati a contribuire allo sviluppo di Ghini, un buon modo per farlo sarebbe aiutandoci a trovare e scrittura di unit test mancante.

Una funzione ben testata è uno cui comportamento non è possibile modificare senza rompere almeno un test di unità.

Siamo tutti d'accordo che in teoria teoria e pratica corrispondono perfettamente e che uno prima scrive i test, quindi implementa la funzione. In pratica, tuttavia, pratica non corrisponde a teoria e noi abbiamo scritto test dopo scrivendo e pubblicando anche le funzioni.

In questa sezione viene descritto il processo di aggiunta di unit test per `bauble.plugins.plants.family.remove_callback`.

Elementi di cui eseguire il test

Prima di tutto, aprire l'indice del rapporto di copertura e scegliere un file con bassa copertura.

In questo esempio, eseguire in ottobre 2015, siamo sbarcati su `bauble.plugins.plants.family`, al 33%.

https://coveralls.io/builds/3741152/source?filename=bauble%2Fplugins%2Fplants%2Ffamily.py

Le prime due funzioni che hanno bisogno di prove, di `edit_callback` e `add_genera_callback`, includono la creazione e l'attivazione di un oggetto basandosi su una finestra di dialogo personalizzata. Ci dovrebbe davvero prima scrivere unit test per tale classe, poi torna qui.

La funzione successiva, `remove_callback`, attiva anche un paio di finestre di dialogo e il messaggio, ma in forma di richiamo di una funzione che richiede l'input dell'utente tramite caselle di sì-no-ok. Queste funzioni possiamo facilmente sostituire con una funzione beffardo il comportamento.

Come provare

Così, dopo aver deciso cosa descrivere nello unit test, guardiamo il codice e vediamo che ha bisogno di discriminare un paio di casi:

correttezza dei parametri

- l'elenco delle famiglie non ha elementi.
- l'elenco delle famiglie ha più di un elemento.
- l'elenco delle famiglie ha esattamente un elemento.

cascade

- la famiglia non ha nessun generi
- la famiglia ha uno o più generi

confirm

- l'utente conferma eliminazione
- l'utente non confermare l'eliminazione

deleting

- tutto va bene quando si elimina la famiglia

- C'è qualche errore durante l'eliminazione della famiglia

Decido che mi concentrerò solo sugli aspetti **cascata** e **confermare**. Due domande binarie: 4 casi.

dove mettere i test

Individuare lo script di test e scegliere la classe dove mettere i test di unità extra.

https://coveralls.io/builds/3741152/source?filename=bauble%2Fplugins%2Fplants%2Ftest.py#L273

---

che fare con i test disattivati

La classe `FamilyTests` contiene un test saltato, attuarlo sarà essere un po" di lavoro perché abbiamo bisogno di riscrivere il FamilyEditorPresenter, separarla dalla FamilyEditorView e riconsiderare ciò che a che fare con la classe FamilyEditor, che credo dovrebbe essere rimosso e sostituito con una singola funzione.

---

le prove di scrittura

Dopo l'ultimo test nella classe FamilyTests, aggiungere i quattro casi che voglio descrivere, e mi assicuro che falliscono, e dato che io sono pigro, scrivo il codice più compatto so per la generazione di un errore:

```
def test_remove_callback_no_genera_no_confirm(self):
    1/0

def test_remove_callback_no_genera_confirm(self):
    1/0

def test_remove_callback_with_genera_no_confirm(self):
    1/0

def test_remove_callback_with_genera_confirm(self):
    1/0
```

Una prova, passo dopo passo

Cominciamo con il primo test case.

Durante la scrittura di test, seguono in genere il modello:

- T_0 (condizione iniziale),
- azione,
- T_1 (test il risultato dell'azione data le condizioni iniziali)

la ragione di un nome — test unitari

> C'è una ragione perché unit test sono chiamati unit test. Si prega di test mai due azioni in un test.

Quindi cerchiamo di descrivere T_0 per il primo test, un database che tiene una famiglia senza generi:

```
def test_remove_callback_no_genera_no_confirm(self):
    f5 = Family(family=u'Arecaceae')
    self.session.add(f5)
    self.session.flush()
```

Non vogliamo che la funzione in fase di test per richiamare la funzione interattiva `utils.yes_no_dialog`, `remove_callback` vogliamo di richiamare una funzione di sostituzione non interattiva. Raggiungiamo questo obiettivo semplicemente facendo punto di `utils.yes_no_dialog` in un'espressione `lambda` che, come l'originale funzione interattiva, accetta un parametro e restituisce un valore booleano. In questo caso: `False`:

```
def test_remove_callback_no_genera_no_confirm(self):
    # T_0
    f5 = Family(family=u'Arecaceae')
    self.session.add(f5)
    self.session.flush()

    # action
    utils.yes_no_dialog = lambda x: False
    from bauble.plugins.plants.family import remove_callback
    remove_callback(f5)
```

Successivamente abbiamo il risultato del test.

Beh, non vogliamo solo testare o meno l'oggetto è stato eliminato Arecaceae, ci dovremmo prova anche il valore restituito da "remove_callback ", e se `yes_no_dialog` e `message_details_dialog` sono stati richiamati o non.

Un'espressione `lambda` non è sufficiente per questo. Facciamo qualcosa di apparentemente più complesso, che renderà la vita molto più facile.

Innanzitutto definiamo una funzione piuttosto generica:

```
def mockfunc(msg=None, name=None, caller=None, result=None):
    caller.invoked.append((name, msg))
    return result
```

e possiamo importare `partial` dal modulo standard `functools`, applicarlo parzialmente alla `mockfunc`, lasciando solo `msg` non vincolato, ed applicare ed utilizzare questa applicazione parziale, che è una funzione che accetta un parametro e restituisce un valore, per sostituire le due funzioni in `utils`. La funzione di test ora assomiglia a questo:

```
def test_remove_callback_no_genera_no_confirm(self):
    # T_0
    f5 = Family(family=u'Arecaceae')
    self.session.add(f5)
    self.session.flush()
    self.invoked = []

    # action
    utils.yes_no_dialog = partial(
        mockfunc, name='yes_no_dialog', caller=self, result=False)
    utils.message_details_dialog = partial(
        mockfunc, name='message_details_dialog', caller=self)
    from bauble.plugins.plants.family import remove_callback
    result = remove_callback([f5])
    self.session.flush()
```

La sezione test controlla che `message_details_dialog` non è stato richiamato, che `yes_no_dialog` è stato richiamato, con il parametro di messaggio corretto, che Arecaceae è ancora lì:

```
# effect
self.assertFalse('message_details_dialog' in
                 [f for (f, m) in self.invoked])
self.assertTrue(('yes_no_dialog', u'Are you sure you want to '
                 'remove the family <i>Arecaceae</i>?')
                in self.invoked)
self.assertEquals(result, None)
q = self.session.query(Family).filter_by(family=u"Arecaceae")
matching = q.all()
self.assertEquals(matching, [f5])
```

E così via

> "ci sono due tipi di persone, coloro che completano quello che iniziano e così via"

Prossimo test è quasi la stessa, con la differenza che il `utils.yes_no_dialog` deve restituire `True` (questo raggiungiamo specificando `result=True` nell'applicazione parziale della `mockfunc` generica).

Con questa azione, il valore restituito da `remove_callback` dovrebbe essere `True`, e non ci dovrebbe essere nessuna famiglia Arecaceae nel database più:

```
def test_remove_callback_no_genera_confirm(self):
    # T_0
    f5 = Family(family=u'Arecaceae')
    self.session.add(f5)
    self.session.flush()
    self.invoked = []
```

(continues on next page)

(continua dalla pagina precedente)

```
    # action
    utils.yes_no_dialog = partial(
        mockfunc, name='yes_no_dialog', caller=self, result=True)
    utils.message_details_dialog = partial(
        mockfunc, name='message_details_dialog', caller=self)
    from bauble.plugins.plants.family import remove_callback
    result = remove_callback([f5])
    self.session.flush()

    # effect
    self.assertFalse('message_details_dialog' in
                     [f for (f, m) in self.invoked])
    self.assertTrue(('yes_no_dialog', u'Are you sure you want to '
                     'remove the family <i>Arecaceae</i>?')
                    in self.invoked)
    self.assertEquals(result, True)
    q = self.session.query(Family).filter_by(family=u"Arecaceae")
    matching = q.all()
    self.assertEquals(matching, [])
```

Date un'occhiata al commit 734f5bb9feffc2f4bd22578fcee1802c8682ca83 per le altre due funzioni di prova.

Prova registrazione

I nostri oggetti di `bauble.test.BaubleTestCase` utilizzano `handler` della classe `bauble.test.MockLoggingHandler`. Ogni volta che viene avviato un singolo unit test, il metodo `setUp` creerà un nuovo `handler` e lo associa al logger radice. Il metodo `tearDown` si prende cura di rimuoverlo.

È possibile controllare per la presenza di messaggi di registrazione specifici in `self.handler.messages`. `messages` è un dizionario, inizialmente vuoto, con due livelli di indicizzazione. Prima il nome del logger che rilascia la registrazione, quindi il nome del livello del record di registrazione. Le chiavi vengono create quando necessario. I valori contengono elenchi di messaggi formattati secondo qualsiasi formattatore si associa al gestore, per difetto `logging.Formatter("%(message)s")`.

È possibile svuotare in modo esplicito i messaggi raccolti richiamando `self.handler.clear()`.

Coordinamento

Di volta in volta si desidera attivare la classe di test che sta lavorando a:

```
nosetests bauble/plugins/plants/test.py:FamilyTests
```

E alla fine del processo si desidera aggiornare le statistiche:

```
./scripts/update-coverage.sh
```

7.1.8 Struttura dell'interfaccia utente

L'interfaccia utente è costruita secondo la **modello — vista — Presenter** modello architettonico. Per gran parte dell'interfaccia, **modello** è un oggetto di database di SQLAlchemy, ma ci sono anche gli elementi dell'interfaccia dove non esiste un corrispondente modello di database. In generale:

- Il **vista** è descritto come parte di un file **glade**. Ciò dovrebbe includere il segnale-callback e associazioni ListStore-TreeView. Basta utilizzare la classe base `GenericEditorView` definito in `bauble.editor`. Quando si crea l'istanza di questa classe generica, passarlo il **glade** nome del file e il nome del widget di radice, quindi consegnare questa istanza per il **presentatore** costruttore.

  Nel file glade, nella sezione `action-widget` chiusura tua descrizione oggetto GtkDialog, assicurarsi che ogni elemento di `action-widget` ha un valore valido `response`. Utilizza valori validi GtkResponseType, ad esempio:

  - GTK_RESPONSE_OK, -5
  - GTK_RESPONSE_CANCEL, -6
  - GTK_RESPONSE_YES, -8
  - GTK_RESPONSE_NO, -9

  Non c'è un modo facile per scrivere prove unitarie per una sottoclasse vista, cui si prega non derivare viste, davvero non ce n'è alcun bisogno.

  Nel file glade, ogni widget input deve definire quale gestore viene attivato il segnale. La classe generica di Presenter offre callback generici che coprono il maggior parte dei casi.

  - GtkEntry (voce di testo a riga singola) gestirà il segnale `changed`, con `on_text_entry_changed` o `on_unique_text_entry_changed`.
  - GtkTextView: associarla a un GtkTextBuffer. Per gestire il segnale `changed` sulla GtkTextBuffer, dobbiamo definire un gestore che richiama il generico `on_textbuffer_changed`, l'unico ruolo per questa funzione è quello di passare il nostro gestore generico il nome dell'attributo modello che riceve il cambiamento. Si tratta di un workaroud per un bug irrisolto in GTK.
  - GtkComboBox con testi tradotti non può essere facilmente gestito dal file glade, così non ci proviamo. Utilizzare il metodo `init_translatable_combo` della classe `GenericEditorView` generica, ma si prega di richiamarlo dal **presentatore**.

- Il **modello** è solo un oggetto con attributi noti. In questa interazione, la **modello** è solo un contenitore di dati passiva, che non si fa nulla di più che lasciare che la **presentatore** modificarlo.
- La sottoclasse **Presenter** definisce e implementa:

- "widget_to_field_map ", un dizionario associando i nomi widget a nome degli attributi di modello,
- "view_accept_buttons ", l'elenco dei nomi di widget che, se attivato dall'utente, significa che la vista deve essere chiusa,
- tutti gli accessori necessari i callback,
- Facoltativamente, essa svolge la **modello** ruolo, troppo.

Il **presentatore** aggiorna continuamente il **modello** in base alle modifiche nella **vista**. Se la **modello** corrisponde a un oggetto di database, la **presentatore** impegna tutti **modello** aggiornamenti al database quando il **vista** è chiuso correttamente, o il rollback se la **vista** viene annullata. (questo comportamento è influenzato dal parametro `do_commit`)

Se il **modello** è un'altra cosa, il **presentatore** farà qualcos'altro.

Nota: Un **presentatore** ben educato utilizza la API della **vista** per interrogare i valori inseriti dall'utente o per impostare lo stato dei widget. Per favore non imparare dalla pratica dei nostri presentatori anomali, alcuni dei quali direttamente gestire campi di `view.widgets`. Così facendo, questi presentatori ci impedisce di scrivere unit test.

La classe base per il presentatore, `GenericEditorPresenter` definito in `bauble.editor`, implementa molte utili callback generici. C'è una classe di `MockView`, che è possibile utilizzare durante la scrittura di test per il tuo presentatori.

Esempi

`Contact` e `ContactPresenter` sono implementati seguendo le linee di cui sopra. La visualizzazione è definita nel file `contact.glade`.

Un buon esempio di modello di visualizzazione/relatore (nessun modello) è dato dalla gestione connessione.

Usiamo lo stesso schema architettonico per l'interazione non di database, impostando il presentatore anche come modello. Facciamo questo, ad esempio, per la finestra di dialogo Esporta JSON. Il seguente comando vi darà un elenco di istanze di `GenericEditorView`:

```
grep -nHr -e GenericEditorView\( bauble
```

7.1.9 Tavolo allungabile Ghini con plugin

Quasi tutto di Ghini è estensibile tramite plugin. Plugin può creare tabelle, definire ricerche personalizzate, aggiungere voci di menu, creare comandi personalizzati e altro ancora.

Per creare un nuovo plugin è necessario estendere la classe `bauble.pluginmgr.Plugin`.

Il plugin `Tag` è un buon esempio minimo, anche se il `TagItemGUI` rientra il pattern architetturale Model-View-Presenter.

7.1.10 Struttura del plugin

Ghini è un framework per la gestione delle collezioni e viene distribuito insieme a una serie di plugin rendendo Ghini un manager collezione botanica. Ma Ghini rimane un quadro e si potrebbe in teoria rimuovere tutti i plugin distribuiamo e Scrivi la tua, o scrivere il proprio plugin che estendono o completare il comportamento attuale di Ghini.

Una volta che avete selezionato e aperto una connessione al database, si terra nella finestra ricerca. La finestra di ricerca è un'interazione tra due oggetti: SearchPresenter (SP) e SearchView (SV).

SV è quello che vedete, SP detiene lo status di programma e gestisce le richieste che si esprimono attraverso SV. Gestire queste richieste influenzano il contenuto di SV e lo stato del programma in SP.

I risultati di ricerca illustrati nella parte più larga della SV sono righe, oggetti che sono istanze di classi registrate in un plugin.

Ciascuna di queste classi deve implementare un preciso insieme di funzioni per poter correttamente operare all'interno di Ghini. Ghini riserva spazio alle classi dei *plugin*.

SP sa di tutte le classi registrate (contenute nei plugin), sono memorizzate in un dizionario, che associa ciascuna classe alla relativa implementazione di plugin. SV ha uno slot (un `gtk.Box`) dove è possibile aggiungere elementi. In qualsiasi momento, al massimo un solo un elemento nello slot è visibile.

Un plugin definisce una o più classi di plugin. Una classe di plugin svolge il ruolo di un presentatore parziale (pP - plugin presenter) come implementare i callback necessari per la visualizzazione parziale associata raccordo nello slot (pV - plugin vista), e il modello MVP è completato dal presentatore padre (SP), ancora una volta funge da modello. Vediamo di riassumere e completare:

- SP funge da modello,
- la visualizzazione parziale di pV è definita in un file glade.
- i callback implementati da pP sono nominati e riferiti nel file glade.
- un menu di scelta rapida per la riga di SP,
- una proprietà dipendente.

Quando si registra una classe di plugin, la SP:

- aggiunge il pV nello slot e lo rende invisibile.
- aggiunge un'istanza di pP nelle classi plugin registrati.
- informa il pP che il modello è SP.
- collega tutte le richiamate dal pV a pP.

Quando un elemento in pV genera un'azione in pP, pP può inoltrare l'azione a SP e può richiedere che SP aggiorni il modello e di conseguenza la vista.

Quando l'utente seleziona una riga in SP, SP nasconde tutto nello slot pluggable e mostra solo il pV corrispondente al tipo della riga selezionata e chiede al pP di aggiornare il pV con tutto ciò che è relativo alla riga selezionata.

Oltre a impostare la visibilità del pV vari, niente deve essere disabilitato né rimosso: un pV invisibile non può innescare eventi!

7.1.11 flusso di lavoro per risolvere bug

flusso di lavoro in condizioni normali

- durante l'utilizzo del software, è possibile notare un problema, o si ottiene un'idea di qualcosa che potrebbe essere meglio, pensate a questo proposito abbastanza buono per avere un'idea molto chiara di quello che realmente è, che notato. si apre un problema e descrive il problema. qualcuno potrebbe reagire con sentori.
- si apre il sito di problemi e scegliere uno che si vuole affrontare.
- assegnare il problema a se stessi, in questo modo che si è informare il mondo che avete intenzione di lavorare a questo. qualcuno potrebbe reagire con sentori.
- Facoltativamente Forkare il repository nel tuo account e preferibilmente creare un ramo, chiaramente associato al problema.
- scrivere unit test e loro si impegnano a tua filiale (per favore non spingere in mancanza di unit test per github, eseguire prima la `nosetests` localmente).
- scrivere più unit test (idealmente, le prove formano la descrizione completa della funzionalità aggiunta o correzione).
- Assicurarsi che la funzionalità che si desidera aggiungere o correggere è davvero completamente descritto dagli unit test che hai scritto.
- Assicurarsi che gli unit test sono atomici, cioè verificare variazioni sui cambiamenti lungo una singola variabile. non dare input complesso di unit test o test che non si adattano su uno schermo (25 righe di codice).
- scrivere il codice che effettua i test riescono.
- aggiornare i file i18n (Esegui `./scripts/i18n.sh`).
- Quando possibile, tradurre le nuove stringhe che metti nei file di codice o radura.
- quando cambi qualche stringa, assicurati che le sue vecchie traduzioni vengano ancora utilizzate.
- confermare le modifiche.
- spingere a github.
- aprire una richiesta di pull.

pubblicazione in produzione

please use the `publish.sh` script, in the `scritps` directory. This one takes care of every single step, and produces recognizable commit comments, it publishes the release on pypi, and in perspective it will contain all steps for producing a `deb` file, and a windows executable.

you can also do this by hand:

- Aprire il pull richiesta pagina utilizzando come base una linea di produzione `ghini-x.y`, rispetto al `ghini-x.y-dev`.
- Assicurarsi che un commit `bump` è incluso nelle differenze.
- dovrebbe essere possibile unire automaticamente i rami.
- creare la nuova richiesta di pull, lo chiamano come «pubblica la linea di produzione».
- Forse devi aspettare per travis-ci eseguire i controlli.
- unire le modifiche.

don't forget to tell the world about the new release: on facebook, the google group, in any relevant linkedin group, and on our web page.

your own fork

If you want to keep your own fork of the project, keep in mind this is full force work in progress, so staying up to date will require some effort from your side.

The best way to keep your own fork is to focus on some specific issue, work relatively quickly, often open pull requests for your work, make sure that you get it accepted. Just follow Ghini's coding style, write unit tests, concise and abundant, and there should be no problem in having your work included in Ghini's upstream.

If your fork got out of sync with Ghini's upstream: read, understand, follow the github guides configuring a remote for a fork and syncing a fork.

Fase di

- Scrivi una recensione su questo flusso di lavoro. considerare questo come una linea guida, a voi stessi e ai vostri colleghi. si prega di contribuire a rendere migliore e la pratica di corrispondenza.

7.1.12 Distributing ghini.desktop

Python Package Index - PyPI

This is not much mentioned, but we keep ghini.desktop on the Python Package Index, so you could install it by no more than:

```
pip install ghini.desktop
```

There are a couple packages that can't be installed with `pip`, but otherwise that's really all you need to type, and it's platform independent.

Publishing on PyPI is a standard `setup` command:

```
python setup.py sdist --formats zip upload -r pypi
```

Windows

For building a Windows installer or executable you need a running Windows system. The methods described here has been used successfully on Windows 7, 8 and 10. Windows Vista should also work but has not been tested.

If you are on GNU/Linux, or on OSX, you are not interested in the remainder of this section. None of Ghini's contributors knows how to produce a Windows installer without having a Windows system.

The goal of the present instructions is to help you produce a Windows installer, that is a single executable that you can run on any Windows workstation and that will install a specific version of ghini.desktop. This is achieved with the NSIS script-driven installer authoring tool.

As a side product of the installer production, you will have a massive but relocatable directory, which you can copy to a USB drive and which will let you use the software without needing an installation.

The files and directories relevant to this section:

- `scripts/build-win.bat` — the single batch script to run.
- `setup.py` — implements the NSIS and py2exe commands.
- `scripts/build-multiuser.nsi` — the nsis script, used by the above.
- `nsis/` — contains redistributable NSIS files, put here for conveniency.
- `ghini-runtime/` — built by `py2exe`, used by `nsis`.
- `dist/` — receives the executable installation file.

Most steps are automated in the `build-win.bat` script. Installation of a few tools needs to be done manually:

1. Download and install Git, Python 2.7 and PyGTK.

   This is outlined in the `devinstall`-based installation instructions.

2. Download and install NSIS v3.

3. A **reboot** is recommended.

4. Clone the ghini.desktop repository.

Use your own fork if you plan contributing patches, or the organization's repository `https://github.com/Ghini/ghini.desktop.git` if you only wish to follow development.

Clone the repository from GitHub to wherever you want to keep it, and checkout a branch. Replace `<path-to-keep-ghini>` with the path of your choice, e.g. `Local\github\Ghini\`. Production branch `ghini-1.0` is recommended as used in the example.

To do this, open a command prompt and type these commands:

```
cd <path-to-keep-ghini>
git clone <ghini.desktop repository URL>
cd ghini.desktop
git checkout ghini-1.0
```

The result of the above is a complete development environment, on Windows, with NSIS. Use it to follow development, or to propose your pull requests, and to build Windows installers.

All subsequent steps are automated in the `scripts\build_win.bat` script. Run it, and after a couple of minutes you should have a new `dist\ghini.desktop-<version>-setup.exe` file, and a working, complete relocatable directory named `ghini-runtime`.

Read the rest if you need details about the way the script works.

The `build_win.bat` script

A batch file is available that can complete the last few steps. To use it use this command:

```
scripts\build_win.bat
```

`build_win.bat` accepts 2 arguments:

1. `/e` — executable only.

   Produce an executable only, skipping the extra step of building an installer, and will copy `win_gtk.bat` into place.

2. `venv_path` — A path to the location for the virtual environment to use.

   Defaults to `"%HOMEDRIVE%%HOMEPATH%"\.virtualenvs\%CHECKOUT%-exe`, where `CHECKOUT` corresponds to the name of the branch you checked out.

If you want to produce an executable only and use a virtual environment in a folder beside where you have ghini.desktop, you could execute `scripts\build_win.bat /e ..\ghi2exe`

---

py2exe will not work with eggs

Building a Windows executable with py2exe requires packages **not** be installed as eggs. There are several methods to accomplish this, including:

- Install using `pip`. The easiest method is to install into a virtual environment that doesn't currently have any modules installed as eggs using `pip install .` as described below. If you do wish to install over the top of an install with eggs (e.g. the environment created by `devinstall.bat`) you can try `pip install -I .` but your mileage may vary.
- By adding:

```
[easy_install]
zip_ok = False
```

to setup.cfg (or similarly `zip_safe = False` to `setuptools.setup()` in `setup.py`) you can use `python setup.py install` but you will need to download and install Microsoft Visual C++ Compiler for Python 2.7 to get any of the C extensions and will need a fresh virtual environment with no dependent packages installed as eggs.

The included `build-win` script uses the `pip` method.

---

installing virtualenv and working with environments

Install virtualenv, create a virtual environment and activate it.

With only Python 2.7 on your system (where `<path-to-venv>` is the path to where you wish to keep the virtual environment) use:

```
pip install virtualenv
virtualenv --system-site-packages <path-to-venv>
call <path-to-venv>\Scripts\activate.bat
```

On systems where Python 3 is also installed you may need to either call pip and virtualenv with absolute paths, e.g. `C:\Python27\Scripts\pip` or use the Python launcher e.g. `py -2.7 -m pip` (run `python --version` first to check. If you get anything other than version 2.7 you'll need to use one of these methods.)

Populate the virtual environment

Install dependencies and ghini.desktop into the virtual environment:

```
pip install psycopg2 Pygments py2exe_py2
pip install .
```

Compile for Windows

Build the executable:

```
python setup.py py2exe
```

The `ghini-runtime` folder will now contain a full working copy of the software in a frozen, self contained state.

This folder is what is packaged by NSIS.

This same folder can also be transferred however you like and will work in place. (e.g. placed on a USB flash drive for demonstration purposes or copied manually to `C:\Program Files` with a shortcut created on the desktop). To start ghini.desktop double click `ghini.exe` in explorer (or create a shortcut to it).

Fixing paths to GTK components.

If you run the relocatable compiled program, unpackaged, you might occasionally have trouble with the GUI not displaying correctly.

Should this happen, you need to set up paths to the GTK components correctly. You can do this by running the `win_gtk.bat`, from the `ghini-runtime` folder.

You will only need to run this once each time the location of the folder changes. Thereafter `ghini.exe` will run as expected.

Finally, invoke NSIS

Build the installer:

```
python setup.py nsis
```

This should leave a file named `ghini.desktop-<version>-setup.exe` in the `dist` folder. This is your Windows installer.

about the installer

- Capable of single user or global installs.
- At this point in time ghini.desktop installed this way will not check or or notify you of any updated version. You will need to check yourself.
- Capable of downloading and installing optional extra components:
  - Apache FOP - If you want to use xslt report templates install FOP. FOP requires Java Runtime. If you do not currently have it installed the installer will let you know and offer to open the Oracle web site for you to download and install it from.
  - MS Visual C runtime - You most likely don't need this but if you have any trouble getting ghini.desktop to run try installing the MS Visual C runtime (e.g. rerun the installer and select this component only).

- Can be run silently from the commandline (e.g. for remote deployment) with the following arguments:
  - `/S` for silent;
  - `/AllUser` (when run as administrator) or `/CurrentUser`
  - `/C=[gFC]` to specify components where:

    `g` = Deselect the main ghini.desktop component (useful for adding optional component after an initial install)

    `F` = select Apache FOP

    `C` = select MS Visual C runtime

Debian

Between 2009 and 2010 someone packaged the then already obsolete Bauble 0.9.7 for Debian, and the package was included in Ubuntu. That version is still being distributed, regardless being it impossible to install.

Only recently has Mario Frasca produced a new bauble debian package, for the latest bauble.classic version 1.0.56, and proposed for inclusion in Debian. View it on mentors. This version depends on `fibra`, a package that was never added to Debian and which Mario also has packaged and proposed for inclusion in Debian. Mario has been trying to activate some Debian Developer, to take action. There's not much more we can do, other than wait for a sponsor, and hoping the package will eventually get all the way to Ubuntu.

Once we get in contact with a Debian Sponsor who will review what we publish on mentors, then we will be definitely expected to keep updating the debian package for `ghini.desktop` and `fibra`.

I am not going to explain in a few words the content of several books on Debian packaging. Please choose your sources. For a very compact idea of what you're expected to do, have a look at `scripts/pubish.sh`.

7.2 Template Letters

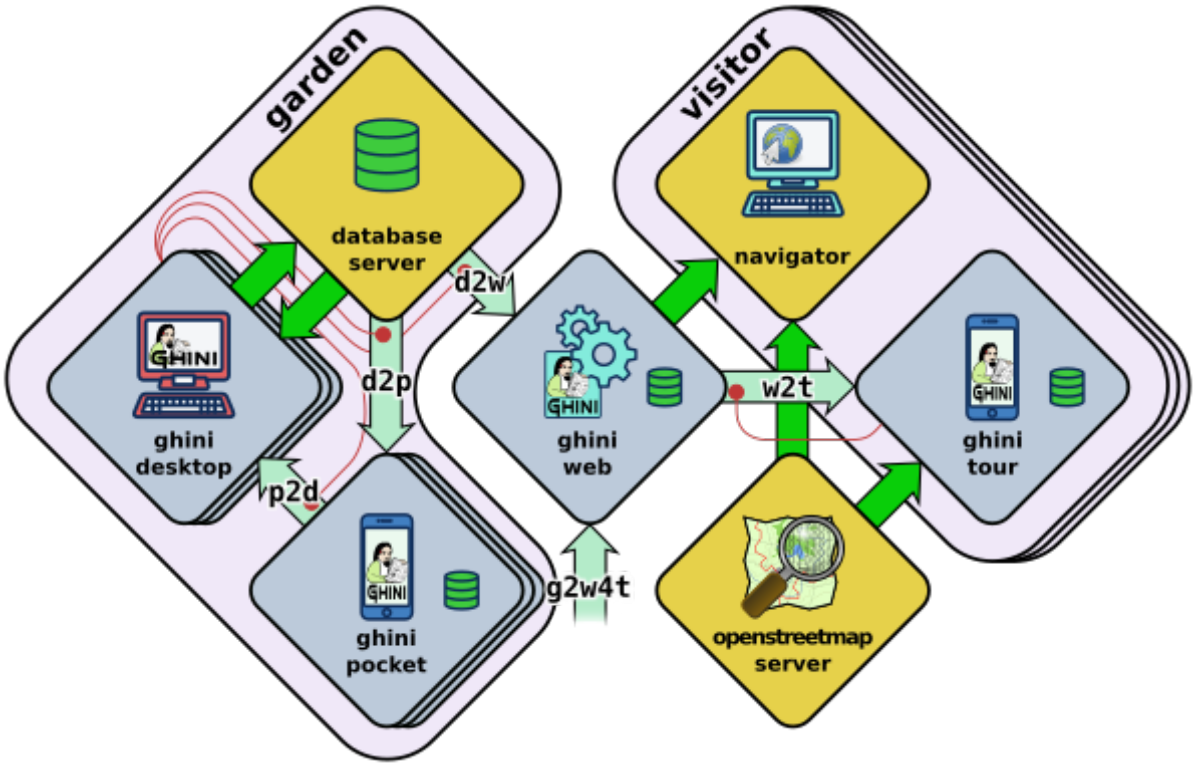
The reader getting to this point in the documentation probably understood that this Ghini project is above all a very open and collaborative project.

Here in this page you find some template letters, used to welcome new users, or that you can correct, print, and go with it to a garden, and propose them to adopt Ghini, or share with a group of your local friends, so you can make Ghini become a (voluntary, or paid) part-time job for you.

7.2.1 Dear conservator or scientist,

You are reading Ghini's presentation letter. Ghini is a libre software project on GitHub, focusing on botany. Brought to you by a small community of coders, botanists, translators, and supported by a few institutions around the world, among which, gardens that have adopted it for all their collection management needs.

The Ghini family is a software suite composed of standalone programs, data servers and handheld clients, for data management, and publication:



- Ghini's core, `ghini.desktop`, lets you
  - enter and correct your data
  - navigate its links,
  - produce reports
  - import and or export using several standard or ad-hoc formats
  - review your taxonomy using online sources

  all according best practices suggested by top gardens, formalized in standard formats like ABCD, ITF2, but also as elaborated by our developers, based on the feedback of Ghini users.

  `ghini.desktop` is developed and continously tested on GNU/Linux, but runs equally well on Windows, or OSX. [1]

- `ghini.pocket` is your full time garden companion, an Android app installed from the Play Store,
  - assisting you in collecting or correcting data while in the field,

  - associate pictures to your plants, and verify taxonomic information.
  - Import your collected data into the desktop client when back in the office,

  `ghini.pocket` reduces the time spent in front of your desktop PC to a true minimum.

- `ghini.web` is a web server and a courtesy data hub service, offering you world wide visibility: Export a selection of your data from your desktop database, and handle it for publication to the Ghini project, and we will include it at http://gardens.ghini.me/, at no cost while we're able to do that, or for a guaranteed minimal amount of time if you are able to support our hosting costs. `ghini.web` serves a world map to help locate participating gardens, and within each garden, its contributed georeferenced plants.
- `ghini.tour`, a geographic tour Android app aimed at visitors, using OpenStreetMap as a base map, retrieving its data, gardens and virtual panels, from the web data aggregator `ghini.web`.

All software within the Ghini family is either licensed GNU Public License v2+ or v3+. It is a strong copyleft license. In short, the GPL translates the ethical scientific need to share knowledge, into legal terms. If you want to read more about it, please refer to https://www.gnu.org/licenses/copyleft.html

Ghini's idea about knowledge and software ownership is that software is procedural knowledge and as such, should be made a «commons»: With software as a commons, «libre software» and more specifically «Copylefted software», you not only get the source code, you receive the right to adapt it, and the invitation to study and learn from it, and to share it, both share forward to colleagues, and share back to the source. With proprietary software, you are buying your own ignorance, and with that, your dependency.

This fancy term «copyleft» instead of just «libre software», means the software you received is libre software with one extra freedom, guaranteeing every right you were granted upon receiving the software is never lost.

With copylefted software you are free —actually welcome— to employ local software developers in your neighbourhood to alter the software according to your needs, please do this on GitHub, fork the code, develop just as openly as the common practice within Ghini, and whenever you want, open a pull request so your edits can be considered for inclusion in the main branch. Ghini is mostly continuously unit tested, so before your code is added to the main branch, it should follow our quality guidelines for contributions. With libre software you acquire freedom and contribute to it, something that earns you visibility: Your additions stays yours, you share them back to the community, and will see them completed and made better by others. Having your code added to the main branch simplifies your upgrade procedure.

You can also contribute to the software by helping translate it into your native language. [5]

Some videos are published on YouTube, highlighting some of the software capabilities. [6]

Share back with the community. Several developers have spent cumulatively many thousand hours developing this software, and we're sharing with the community. We hope by this to stimulate a community sentiment in whoever starts using what we have produced.

Thanks for your consideration; please let me know if you have any questions,

In case you're interested in publishing your tree collection on the net, I would be happy to include your plants, species, coordinates to http://gardens.ghini.me. Georeferenced textual information panels are also very welcome, all offered as a courtesy: We're still defining the offer. The idea behind this is allowing visitors to explore aggregated garden collections, and the current focus is on trees.

A small example: http://gardens.ghini.me/#garden=Jardín%20el%20Cuchubo

Mario Frasca MSc

[1] http://ghini.readthedocs.io/ - http://ghini.github.io/
[2] https://play.google.com/store/apps/details?id=me.ghini.pocket
[3] http://gardens.ghini.me/
[4] https://play.google.com/store/apps/details?id=me.ghini.tour
[5] https://hosted.weblate.org/projects/ghini/#languages
[6] https://www.youtube.com/playlist?list=PLtYRCnAxpinU_8WEDuRlgsYnNVe4J_4kv

7.2.2 free botanic data management systems

Many institutions still consider software an investment, an asset that is not to be shared with others, as if it was some economic good that can't be duplicated, like gold.

As of right now, very few copylefted programs exist for botanic data management:

- `ghini.desktop`, born as `bauble.classic` and made a commons by the Belize Botanical Garden. `ghini.desktop` has three more components, a pocket data collecting Android app, a Node.js web server, aggregating data from different gardens and presenting it geographically, again a geographic tour app aimed at visitors using the web data aggregator as its data source. You can find every Ghini component on GitHub: http://github.com/Ghini
- Specify 6 and 7, made a Commons by the Kansas University. A bit complex to set up, very difficult to configure and tricky to update. The institutions I've met who tried it, only the bigger ones, with in-house software management capabilities manage to successfully use it. They use it for very large collections. Specify is extremely generic, it adapts to herbaria, seed collections, but also to collections of eggs, organic material, fossils, preserved dead animals, possibly even viruses, I'm not sure. It is this extreme flexibility that makes its configuration such a complex task. Specify is also on GitHub: https://github.com/specify and is licensed as GPLv2+.
- Botalista, a French/Swiss cooperation, is GPL as far as rumours go. Its development has yet to go public.
- `bauble.web` is an experimental web server by the author of `bauble.classic`. `bauble.classic` has been included into Ghini, to become `ghini.desktop`. Bauble uses a very permissive license, making it libre, but not copylefted. As much as 50% of bauble.web and possibly 30% of ghini.desktop is shared between the two projects. Bauble seems to be stagnating, and has not yet reached a production-ready stage.

- `Taxasoft-BG`, by Eric Gouda, a Dutch botanist, specialist in Bromeliaceae, collection manager at the Utrecht botanical garden. It was Mario Frasca who convinced Eric to publish what he was doing, licensing it under the GPL, but the repository was not updated after 2016, April 13th and Eric forgot to explicitly specify the license. You find it on github: https://github.com/Ejgouda/Taxasoft-BG
- `BG-Recorder`, by the BGCI, runs on Windows, and requires Access. Developed mostly between 1997 and 2003, it has not been maintained ever since and isn't actively distributed by the BGCI. I've not managed to find a download link nor its license statement. It is still mentioned as *the free option* for botanic database management.

Of the above, only `ghini.desktop` satisfies these conditions: Copylefted, available, documented, maintained, easy to install and configure. Moreover: Cross platform and internationalized.

7.2.3 Welcome to Ghini/Bauble

Dear new user,

Welcome to Ghini/Bauble.

As the maintainer, I have received your registration for bauble.classic/ghini.desktop, many thanks for taking your time to fill in the form.

I see you are using bauble.classic-1.0.55, whereas 1.0.55 is the last released version of bauble.classic, however, bauble.classic is now unmaintained and superseded by the fully compatible, but slightly aesthetically different ghini.desktop. Install it following the instructions found at http://ghini.rtfd.io

The registration service says you're not yet using the newest Python2 version available. As of 2018-05-01, that is 2.7.15. Using any older version does not necessitate problems, but in case anything strange happens, please update your Python (and PyGTK) before reporting any errors.

Also thank you for enabling the «sentry» errors and warnings handler. With that enabled, Ghini/Bauble will send any error or warning you might encounter to a central server, where a developer will be able to examine it. If the warning was caused by an error in the software, its solution will be present in a subsequent release of the software

If you haven't already, to enable the sentry and warnings handler, open the «:config» page in Ghini and double click on the row «bauble.use_sentry_client».

I hope Ghini already matches your expectations, if this is not the case, the whole Ghini community would be very thankful if you took the time to report your experience with it.

The above is one way to contribute to Ghini's development. Others are: - contribute ideas, writing on the bauble google forum (https://groups.google.com/forum/#!forum/bauble), - contribute documentation, or translations (https://hosted.weblate.org/projects/ghini/), - give private feedback, writing to ghini@anche.no, - rate and discuss Ghini openly, and promote its adoption by other institutions, - open an issue on GitHub (https://github.com/Ghini/ghini.desktop/issues/), - contribute code on GitHub (fork the project on (https://github.com/Ghini/ghini.desktop/), - hire a developer and have a set of GitHub issues solved, per-haps your own - let me include your garden on the still experimental worldmap (http://gardens.ghini.me)

I sincerely hope you will enjoy using this copylefted, libre software

Best regards, Mario Frasca

https://ghini.github.io https://github.com/Ghini/ghini.desktop/issues/

7.2.4 Do you want to join Ghini?

Nota: I generally send a note similar to the following, to GitHub members who «star» the project, or to WebLate contributors doing more than one line, and at different occasions. If it's from GitHub, and if they stated their geographic location in their profile, I alter the letter by first looking on institutos botánicos if there's any relevant garden in their neighbourhood.

Dear GitHub member, student, colleague, translator, botanist,

Thank you warmly for your interest in the Ghini project!

From your on-line profile on github, I see you're located in Xxxx, is that correct?

If you are indeed in Xxxx, you live very close to gardens Yyyy and Zzzz. Maybe you would consider the following proposition? All would start by contacting the botanical garden there, and get to know what software they use (what it offers, and at which price) and if they're interested in switching to ghini.desktop+pocket+tour+web.

The business model within Ghini is that the software is free and you get it for free, but time is precious and if a garden needs help, they should be ready to contribute. Maybe you already have a full-time job and don't need more things to do, but in case you're interested, or you have friends who would be, I'm sure we can work something out.

Let me know where you stand.

best regards, and again thanks for all your contributed translations.

Mario Frasca

CAPITOLO 8

Appoggiare Ghini

If you're using Ghini, or if you feel like helping its development anyway, please consider donating.