Gestione Apiario

Sommario

[1 Introduzione 3](#_Toc59139610)

[1.1 Informazioni sul progetto 3](#_Toc59139611)

[1.2 Abstract 3](#_Toc59139612)

[1.3 Scopo 3](#_Toc59139613)

[2 Analisi 4](#_Toc59139614)

[2.1 Analisi del dominio 4](#_Toc59139615)

[2.2 Analisi e specifica dei requisiti 4](#_Toc59139616)

[2.3 Use case 7](#_Toc59139617)

[2.4 Pianificazione 7](#_Toc59139618)

[2.5 Analisi dei mezzi 8](#_Toc59139619)

[2.5.1 Software 8](#_Toc59139620)

[2.5.2 Hardware 8](#_Toc59139621)

[3 Progettazione 9](#_Toc59139622)

[3.1 Design dell’architettura del sistema 9](#_Toc59139623)

[3.2 Design dei dati e database 10](#_Toc59139624)

[3.2.1 Descrizione delle tabelle del database 10](#_Toc59139625)

[3.3 Design delle interfacce 12](#_Toc59139626)

[3.4 Design procedurale 14](#_Toc59139627)

[4 Implementazione 15](#_Toc59139628)

[4.1 Database 15](#_Toc59139629)

[4.2 Applicativo 15](#_Toc59139630)

[4.2.1 Struttura 15](#_Toc59139631)

[4.2.2 Sviluppo 16](#_Toc59139632)

[4.2.2.1 Connessione al database 16](#_Toc59139633)

[4.2.2.2 Connessione al Database nelle classi 16](#_Toc59139634)

[4.2.2.3 Select, Insert, Update, delete sul Database 17](#_Toc59139635)

[4.2.3 Controllo connessione internet 19](#_Toc59139636)

[4.2.4 Mappatura json 19](#_Toc59139637)

[4.2.5 Meteo 20](#_Toc59139638)

[4.2.6 Generatore di eventi 20](#_Toc59139639)

[4.2.7 Ascoltatore di eventi 21](#_Toc59139640)

[4.3 Password Database 21](#_Toc59139641)

[5 Test 21](#_Toc59139642)

[5.1 Protocollo di test 21](#_Toc59139643)

[5.2 Risultati test 23](#_Toc59139644)

[5.3 Mancanze/limitazioni conosciute 23](#_Toc59139645)

[6 Consuntivo 24](#_Toc59139646)

[7 Conclusioni 24](#_Toc59139647)

[7.1 Sviluppi futuri 24](#_Toc59139648)

[7.2 Considerazioni personali 24](#_Toc59139649)

[8 Bibliografia 24](#_Toc59139650)

[8.1 Sitografia 24](#_Toc59139651)

[9 Allegati 25](#_Toc59139652)

# Introduzione

## Informazioni sul progetto

Titolo Progetto: Gestione Apiario

Docente responsabile: Luca Muggiasca, Geo Petrini

Allievo responsabili: Alessandro Aloise

CPT Trevano sezione Informatica

Classe: I3AC

Materia: Modulo 306

Inizio progetto 3 settembre 2020

Fine progetto 20 dicembre 2020

## Abstract

Al giorno d’oggi è impensabile che un apicoltore si debba ancora marcare lo stato delle sue arnie o i suoi trattamenti effettuati tutto su un foglio di carta che poi deve immettere in una cartella per tenere memoria di tutto questo. Questo progetto permette di avere una gestione delle proprie arnie in modo moderno. Grazie al comodo calendario e alla possibilità di vedere il meteo della propria arnia in qualsiasi momento. Con questo semplice progetto non è più necessario utilizzare carta e penna per la gestione delle proprie arnie, grazie a Gestione Apiario le tue api sono a portata di mano.

*“Nowadays it is unthinkable that a beekeeper still has to mark the state of his hives or his treatments carried out all on a sheet of paper which he then has to put in a folder to keep track of all this. This project allows you to manage your hives in a modern way. Thanks to the convenient calendar and the ability to see the weather of your hive at any time. With this simple project it is no longer necessary to use pen and paper to manage your hives, thanks to Gestione Apiario your bees are at your fingertips.”*

## Scopo

Lo scopo di questo progetto didattico è quello di creare un sistema di gestione delle arnie dislocate in vari punti geografici. La gestione viene tramite applicativo Java. Si ha la possibilità di avere vari utenti e ognuno gestisce le proprie arnie, l’applicativo da anche la possibilità di gestire il meteo e varie altre funzioni.

# Analisi

## Analisi del dominio

Questo progetto serve per gestire più apiari. Potrà e verrà utilizzato da qualunque apicoltore voglia utilizzarlo sullo stesso pc. Questo applicativo java, ovvero verrà avviata una schermata dalla quale si può gestire il le proprie arnie.

## Analisi e specifica dei requisiti

|  |  |
| --- | --- |
| **ID: REQ-001** | |
| **Nome** | Possibilità di inserire il da una a più arnie |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** |  |

|  |  |
| --- | --- |
| **ID: REQ-002** | |
| **Nome** | Registrare delle annotazioni in stile diario per ogni arnia |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** |  |

|  |  |  |
| --- | --- | --- |
| **ID: REQ-003** | | |
| **Nome** | | Trattamenti sanitari per ogni arnia |
| **Priorità** | | 1 |
| **Versione** | | 1.0 |
| **Note** | | Data o giorni |
| **Sotto requisiti** | | |
| **001** | Deve essere indicato tramite data di inizio e di finite | |
| **002** | Può essere indicato tramite data di inizio e quantità di giorni che deve durare e il sistema fa il calcolo fino a quando dura | |
| **003** | Notifica di fine attività | |

|  |  |
| --- | --- |
| **ID: REQ-004** | |
| **Nome** | Tutte le attività registrate devono poter essere visibili in stile calendario. |
| **Priorità** | 2 |
| **Versione** | 1.0 |
| **Note** | Formattazione stile calendario di Google |

|  |  |
| --- | --- |
| **ID: REQ-005** | |
| **Nome** | Data di nascita di ogni ape regina |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** |  |

|  |  |
| --- | --- |
| **ID: REQ-006** | |
| **Nome** | Meteo visibile nel calendario |
| **Priorità** | 2 |
| **Versione** | 1.0 |
| **Note** | Pop up |

|  |  |
| --- | --- |
| **ID: REQ-007** | |
| **Nome** | Formazione data 01.12.2020 (Data usata in tutto l’applicazione ) |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** |  |

|  |  |
| --- | --- |
| **ID: REQ-008** | |
| **Nome** | Esportazio del calendario con un riassunto |
| **Priorità** | 4 |
| **Versione** | 1.0 |
| **Note** | bonus |

|  |  |
| --- | --- |
| **ID: REQ-009** | |
| **Nome** | Esportazioni del diario con un riassunto |
| **Priorità** | 4 |
| **Versione** | 1.0 |
| **Note** | bonus |

|  |  |
| --- | --- |
| **ID: REQ-010** | |
| **Nome** | L'utente deve poter scegliere il tipo di attività |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** |  |

|  |  |
| --- | --- |
| **ID: REQ-011** | |
| **Nome** | L'inserimento delle attività deve essere il più semplice possibile |
| **Priorità** | 2 |
| **Versione** | 1.0 |
| **Note** |  |

## Use case

In questo digramma dell’use case possiamo osservare come l’utente accede all’applicativo facendo un login o un register, da li in poi l’utente potrà fare tutte le azioni che vuole sulle sue arnie. Importante dopo un register bisogna far ripartire l’applicazione

## Pianificazione

|  |
| --- |
| Figura 1: L'immagine indica il gantt stilato ad inizio progetto. Le due fasi che sulla carta impiegano più tempo è l'implementazione. Questa fase sono circa 2/3 del tempo della durata del progetto. Questo è il gantt iniziale, potrebbe essere differente dal gantt consuntivo indicato nell'immagine sotto |

## Analisi dei mezzi

Elencare e descrivere i mezzi disponibili per la realizzazione del progetto. Ricordarsi di sempre descrivere nel dettaglio le versioni e il modello di riferimento.

### Software

Per realizzare questo progetto sono stati utilizzati i seguenti software:

• ProjectLibre - 1.9.1: Per la creazione del Gantt.

• Java – 14.0.2 Per la creazione dell'applicativo principale

• SQLite studio 3.2.1: Per creare e contenere il database

• Netbeans 12.0: Per tutto ciò che riguarda la scrittura del codice Java

• LGoodDatePicker-11.1.0 Libreria per la creazione del calendario

• Sqlite-jdbc-3.7.2: Libreria per implementare sqlite su Java

• Microsoft Project 2016

• Draw.io

• Mockflow

L'applicativo è multipiattaforma (Windows, MacOS, Linux). Il progetto non necessa forze computazionali esagerate, necessita però di ava installato sul pc. Senza Java l'applicativo non funzionerà.

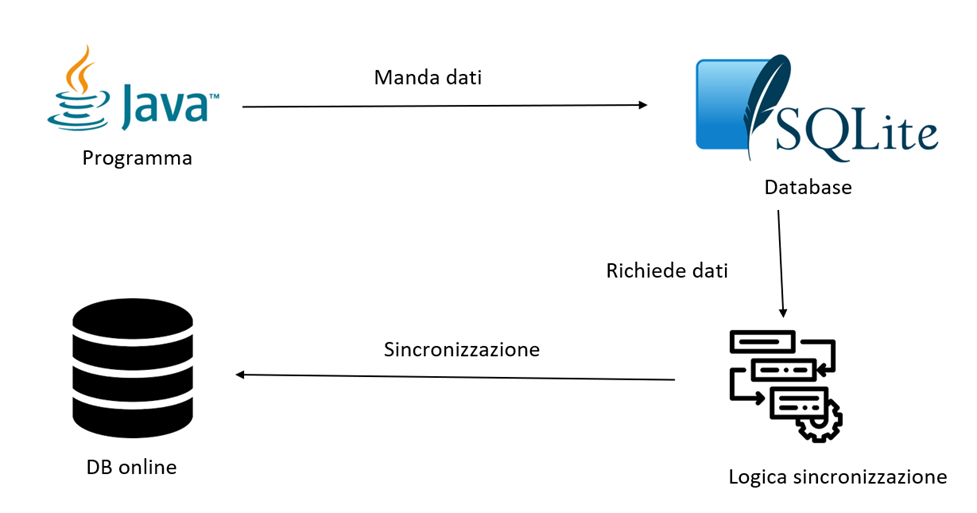
### Hardware

Per questo progetto sono stati usati i seguenti software:

* Pc fisso di scuola
* Del XPS 15 9500 (2020)

# Progettazione

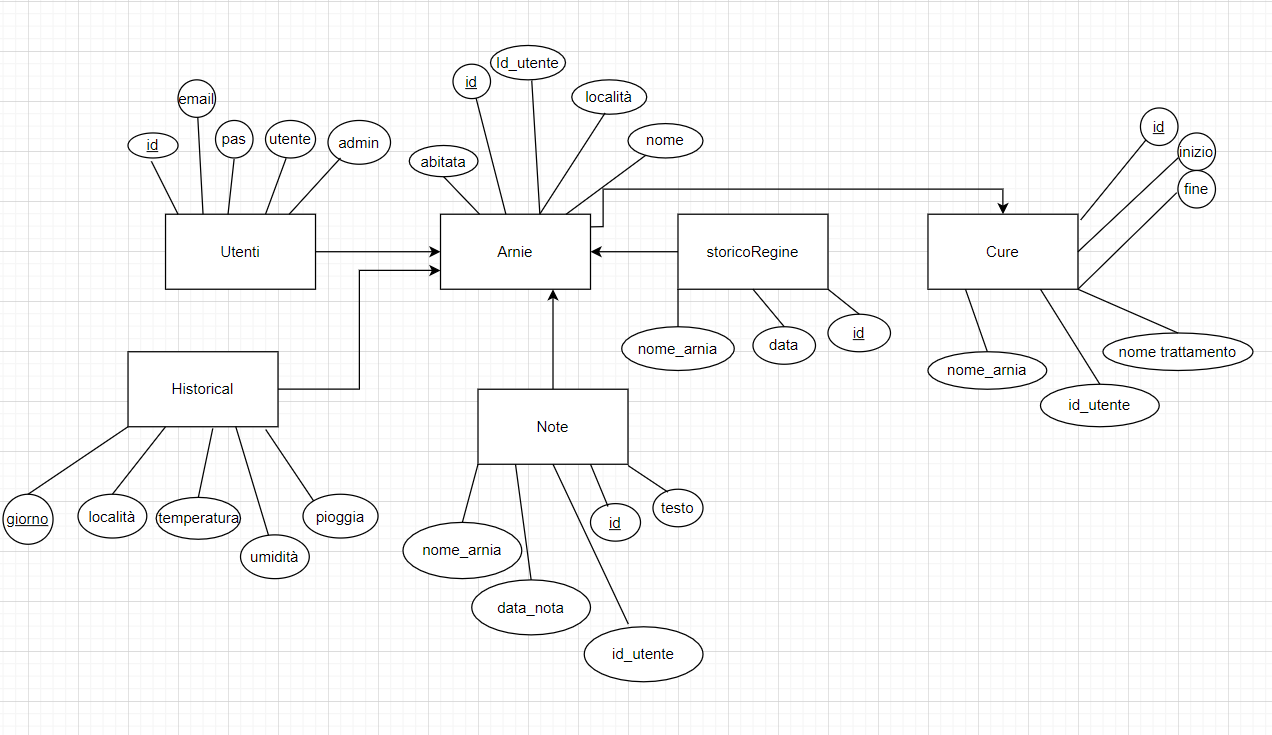
## Design dell’architettura del sistema



L'immagine mostra lo schema della struttura del progetto. Il progetto è composto da un programma che riceve i dati dal utente. Ad ogni modifica inserisce un dato all'interno del database (all'inizio lo crea se non esiste). La logica per sincronizzare i due database è che ogni volta che mi collego tramite applicativo in locale il programma scarica la versione più recente del database da database online, invece ogni volta che faccio il logout il database appena modificato viene caricato online. In caso che non ci sia una connessione a internet il programma usa l’ultima versione che trova del database locale.

## Design dei dati e database

Gli utenti e i tutti i dati relativi ad uno di essi verranno salvati in questo database.



Questo è lo schema ER utilizzato dalla banca dati, il database è diviso in 6 tabelle.

## Descrizione delle tabelle del database

|  |  |
| --- | --- |
| **Utenti** | |
| Attributo | Descrizione |
| id | L’id incrementale univoco |
| utente | Il nome dell’utente |
| pas | La password non cifrata con almeno una minuscola, un numero e almeno 6 caratteri fino a un massimo di 15. |
| email | L’email con la quale l’utente si è registrato |
| admin | Se l’utente è admin o no |

|  |  |
| --- | --- |
| **Arnie** | |
| id | L’id incrementale univoco |
| abitata | True se è abitata false se non è abitata |
| id\_utente | Id dell’arnia univoco |
| località | Luogo dove si trova l’arinia |
| nome | Il nome dell’arina |

|  |  |
| --- | --- |
| **storicoRegine** |  |
| id | L’id incrementale univoco |
| data | Data di nascita della regina |
| nome\_arina | Nome che ha l’arnia |

|  |  |
| --- | --- |
| **cure** | |
| id | L’id incrementale univoco |
| Inizio | Data di inizio delle cure |
| fine | Data di fine delle cure |
| nome\_trattamento | Nome del trattamento |
| nome\_arina | Nome dell’arnia |
| id\_utente | Id dell’utente che ha creato le cure |

|  |  |
| --- | --- |
| **note** | |
| id | L’id incrementale univoco |
| teso | Note utente |
| data\_nota | Data della nota |
| nome\_arina | Nome dell’arnia |
| id\_utente | Id dell’utente che ha creato le cure |

|  |  |
| --- | --- |
| **Historical** | |
| giorno | giorno meteo |
| Località | Posto riferito al meteo |
| Temperatura | La temperatura della giornata |
| Umidità | Umidità del posto |
| Pioggia | Pioggia del posto |

## Design delle interfacce

Figura La prima interfaccia che sono andato a definire é stata quella di avvio

Figura Questa progettazione di interfaccia rappresenta la schermata di login dell’utente

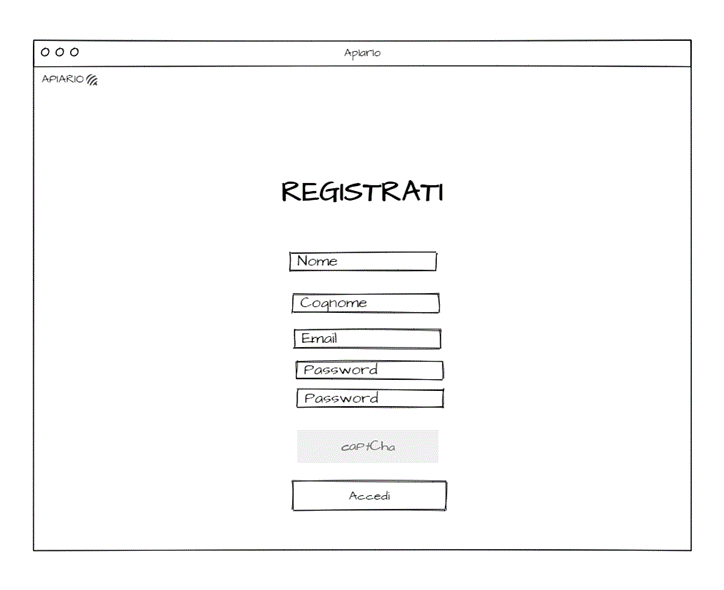


Figura Progettazione di interfaccia rappresenta la schermata di registrazione

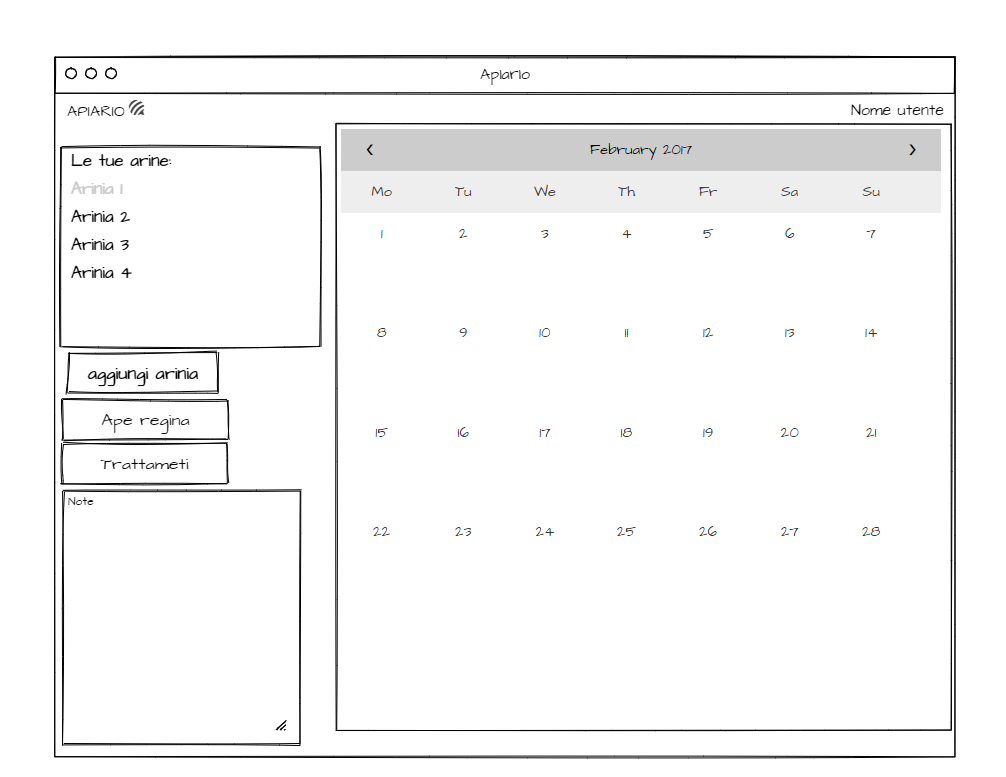


Figura Questa schermata rappresentava la prima progettazione della schermata di utilizzo della applicazione

## Design procedurale

Register

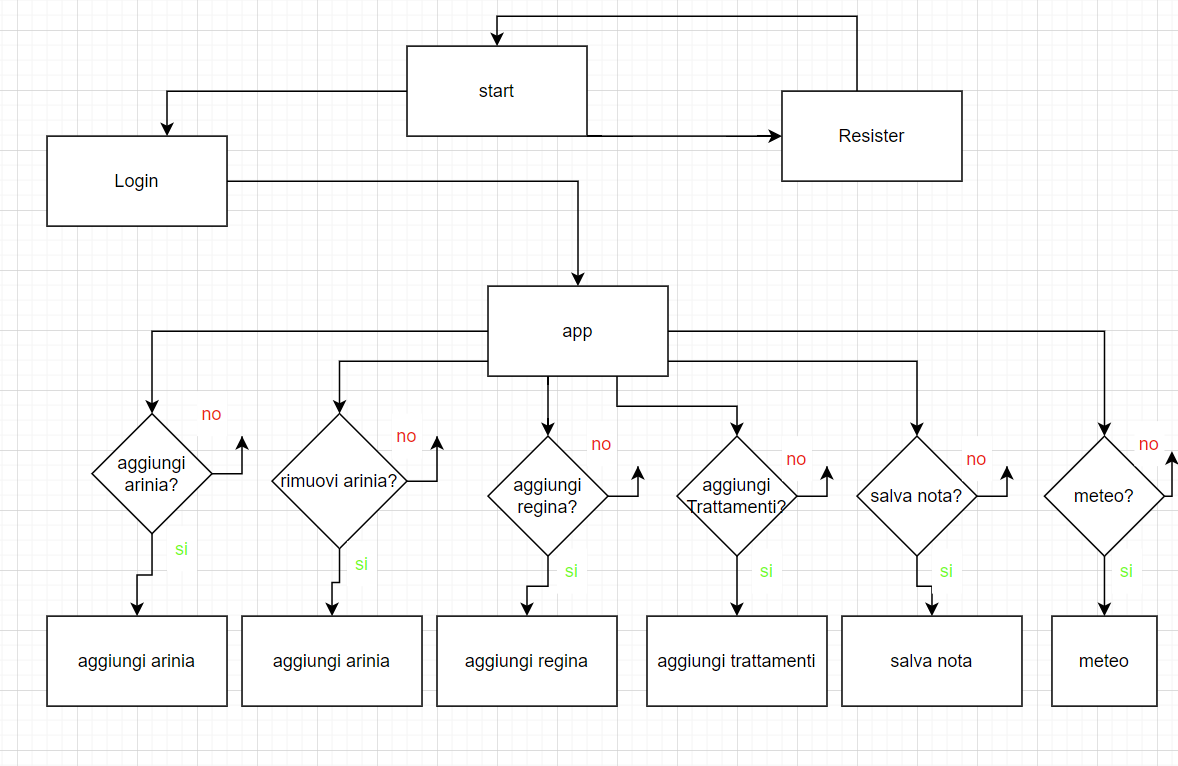


Figura Diagramma di flusso del applicazione

# Implementazione

## Database

Il database è stato implementato sulla base del diagramma ER. Le parti fondamentali nello sviluppo del

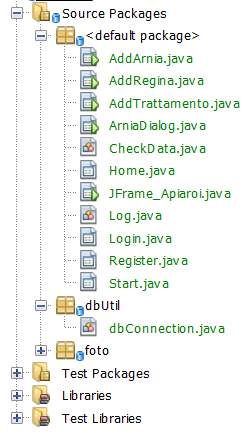
database sono le correlazioni tra le varie tabelle.

Quando un arinia viene cancellata dal database anche tutte le sue informazioni relative vengono cancellate, cure, note e trattamenti. Per quanto riguarda le api regine non vengono mai cancellate ma solo aggiunte cosi da avere uno storico delle regine di tutte le arnie. Per il meteo era prevista una tabella per lo stock dei dati meteo vecchi ma non è stata implementata.

## Applicativo

## Struttura

La struttura di Gestione Apiario è la seguente:



La cartella **default package** contiene tutti i panel, il frame e le classi che non centrano con la connessione al database.

La cartella **dbUtil** contiene la classe per la connessione al database.

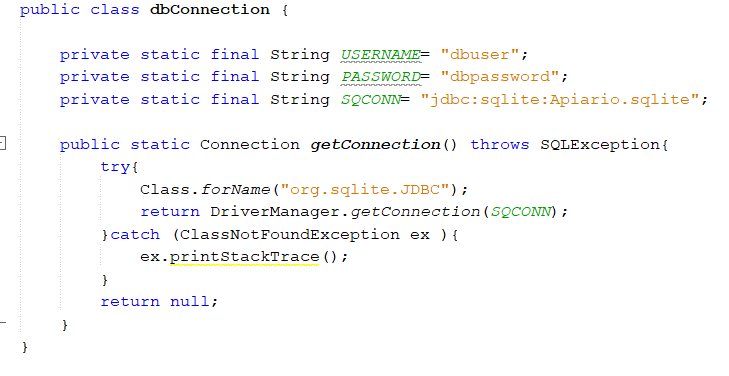
La cartella **Libraries** contiene la jdk 14 e tutte le librerie utile per funzionare l’applicazione

Le due cartelle rimanenti sono carelle di default.

## Sviluppo

## Connessione al database

La connessione al database SQLite viene effettuata attraverso una classe chiamata dbConnection la quale

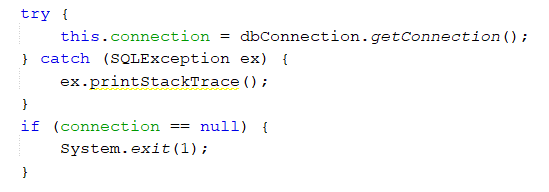
permette di connettersi al server SQLite. IL metodo in questione è statico, Attraverso il seguente codice viene eseguita la connessione al server SQLite utilizzando un nome utente e password dell’utente. Viene ritornato un oggetto Connection.

## Connessione al Database nelle classi

Per utilizzare il database nelle varie classi ho dovuto importare il package dbUtil e implementare il seguente codice:

Come prima cosa dobbiamo mettere questo attributo.



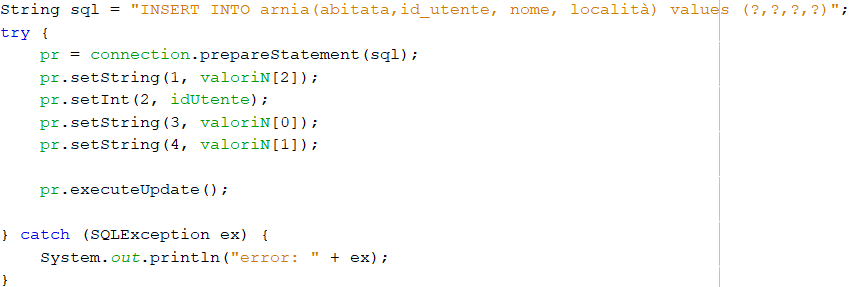
Mentre nel costruttore ho messo questo codice per potermi collegare al Database. In caso di mancata connessione al Database chiude l’applicazione.

## Select, Insert, Update, delete sul Database

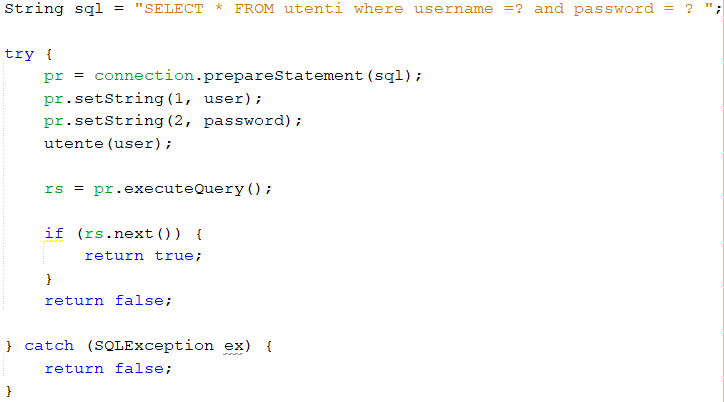
Tutto il codice quando si prova a fare un operazione sul database viene messo tra un try e un catch per un evitare errori.

come prima cosa dobbiamo definire pr che ci servirà per creare la query.

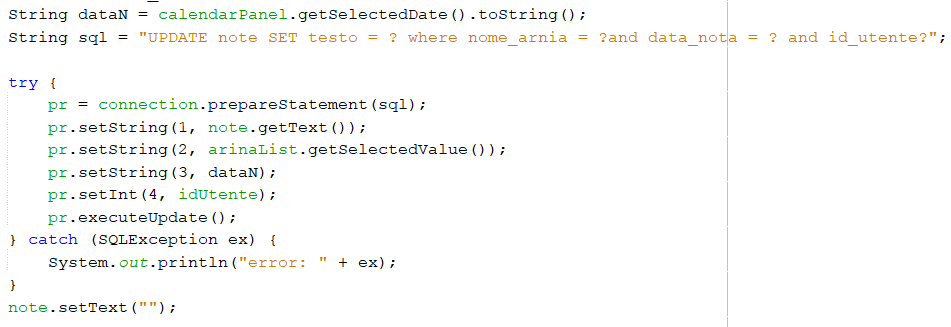
Per fare un **insert** con Sqlite dobbiamo come prima cosa passargli la tabella e subito dopo i parametri, in fine tra parentesi i punti di domanda che rappresentato i dati che gli vogliamo passare. In fine creiamo la query su pr, in fine uso per farla eseguire executeQuery() che esegue la query sul db.

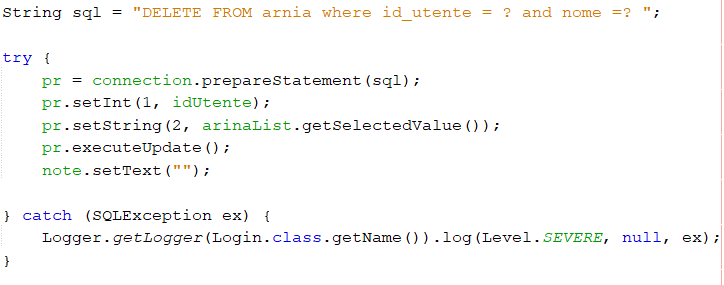


Per fare un **select** con Sqlite dobbiamo come prima cosa passargli la tabella facoltativamente possiamo mettere il where per scegliere maggiormente i dati. In fine creiamo la query su pr. Usando per farla eseguire executeQuery() che esegue la query sul db. Con il .next passo tutte le soluzioni trovate



Per fare un **update** con Sqlite dobbiamo come prima cosa passargli la tabella facoltativamente possiamo mettere il where per scegliere maggiormente i dati. In fine creiamo la query su pr. Uso executeUpdate() per far eseguire la query sul db.



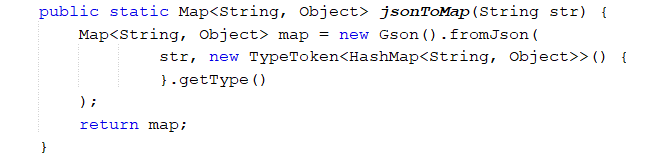
Per fare un **delete** con Sqlite dobbiamo come prima cosa passargli la tabella facoltativamente possiamo mettere il where per scegliere maggiormente i dati. In fine creiamo la query su pr. Uso executeUpdate() per far eseguire la query sul db.

## Controllo connessione internet

Ho creato un metodo per verificare se il pc su qui sta andando l’applicazione ha internet o meno in caso che il pc sia collegato a una rete Wi-Fi o sia collegato via cavo il metodo non fa nulla, in caso contrario il metodo avverte tramite pop up l’utilizzatore che non c’è internet e di conseguenza il meteo non è visibile.

## Mappatura json

Questo metodo si occupa di mappare le varie parti del json ricevendo il “nome” della sezione da mappare. Ritorna un pezzo di json mappato



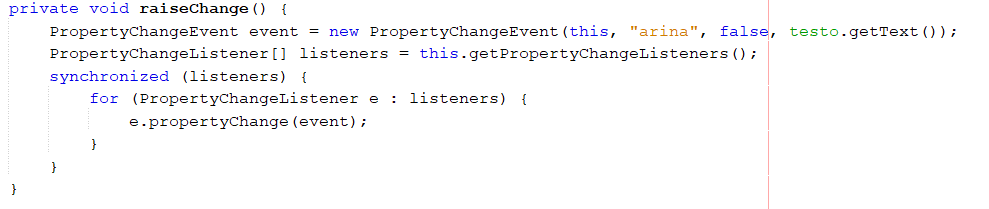
## Meteo

Per permettere all’utente di vedere il meteo in formato pop up ho dovuto scrivere questo codice. Una variabile che definisce la mia api\_key univoca, una stringa contente il paese del meteo che ci serve e poi la stringa url che serve per fare la richiesta al sito e ricevere in dietro un file json, che tramite string Buildre vado a leggere e comporre. Tutto questo poi viene salvato all’intero di result che tramite il metodo visto prima mi permette estrapolare solo le informazioni che voglio io. Tutto questo ci permette tramite un allert di far veder il meteo al nostro utente.

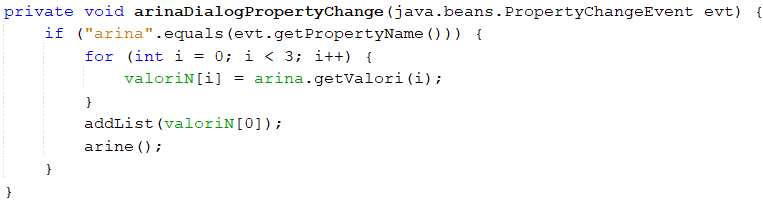


## Generatore di eventi

Questo metodo ci permette di creare un evento personalizzato e chiunque sia in ascolto(vedi 4.2.7) tramite il nome fornito in questo caso(“arina”) potrà sapere che è stato scatenato quell’evento



## Ascoltatore di eventi

Questo metodo ci permette di ascoltare tutti gli eventi personalizzati (vedi punto 4.2.6)

## Password Database

Nel database ci sono due utenti di default che sono:

* Nome utente: admin password: Ciao12345
* Nome utente: utente.prova password: Ciao12345

# Test

## Protocollo di test

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-001  REQ-001 | **Nome:** | Possibilità di inserire il numero di arnie |
| **Descrizione:** | Verificare che si possa creare delle nuove arnie | | |
| **Prerequisiti:** | Il database deve avere la tabella arnie | | |
| **Procedura:** | 1. Fare login con un utente esistente 2. Cliccare sul tasto aggiungi arnia e aggiungerla e andare a vedere se è stata aggiunta al db davvero. | | |
| **Risultati attesi:** | Che nel database ci sia l’Arina appena aggiunta | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-002  REQ-002 | **Nome:** | Registrare delle annotazioni in stile diario per ogni arnia |
| **Descrizione:** | Verificare che si possa scrivere una nota nel calendario a un determinato giorno e controllare che venga salvata | | |
| **Prerequisiti:** | avere il database funzionante e la gui che funziona | | |
| **Procedura:** | 1. Fare login con un utente esistente 2. Cliccare su un’arinia e dopo selezionare il giorno su qui si vuole mettere la nota aggiungi arnia digitare il testo e cliccare salva. Andare a vedere se è stata aggiunta al nota al db davvero. | | |
| **Risultati attesi:** | Che nel database ci sia una nuova nota collegata a quell’arnia selezionata | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-003  REQ-003 | **Nome:** | Trattamenti sanitari per ogni arnia |
| **Descrizione:** | Verificare che si possa immettere un trattamento per un arnia scelta | | |
| **Prerequisiti:** | avere il database funzionante e la gui che funziona | | |
| **Procedura:** | 1. Fare login con un utente esistente 2. Cliccare su un’arinia e dopo selezionare il giorno su qui si vuole mettere la nota aggiungi arnia digitare il testo e cliccare salva. Andare a vedere se è stata aggiunta al nota al db davvero. | | |
| **Risultati attesi:** | Che nel database ci sia un nuovo trattamento collegata a quell’arnia selezionata e che nell’applicazione andando a cliccare quel giorno ci sia una nota con scritto fine trattamento e dopo il nome | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-004  REQ-004 | **Nome:** | Tutte le attività registrate devono poter essere visibili in stile calendario. |
| **Descrizione:** | controllare il corretto funzionamento del calendario | | |
| **Prerequisiti:** | avere il database funzionante e la gui che funziona | | |
| **Procedura:** | 1. Fare login con un utente esistente 2. Vedere se il calendario funziona e se ci sono delle note | | |
| **Risultati attesi:** | che ci sia un calendario simile a quello di Google e ci siano delle note | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-005  REQ-005 | **Nome:** | Data di nascita di ogni ape regina. |
| **Descrizione:** | Che si possa aggiungere una regina con la sua relativa data di nascita | | |
| **Prerequisiti:** | avere il database funzionante e la gui che funziona | | |
| **Procedura:** | 1. Fare login con un utente esistente 2. Scegliere un arnia e cliccare sul bottone aggiungi regina e prove a inserire una data. 3. Andare sul db e vedere se è stata davvero aggiunta l’ape regina con la data che immessa | | |
| **Risultati attesi:** | Che nel database ci sia la nuova regina che abbiamo appena scritto. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-006  REQ-006 | **Nome:** | Meteo visibile nel calendario |
| **Descrizione:** | Che in qualche modo si possa vedere il meteo dell’arinia selezionata | | |
| **Prerequisiti:** | avere il database funzionante e la gui che funziona | | |
| **Procedura:** | 1. Fare login con un utente esistente 2. Cliccare sul bottone meteo | | |
| **Risultati attesi:** | Che si visualizzi a schermo il meteo in qualche modo . | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-010  REQ-010 | **Nome:** | L'utente deve poter scegliere il tipo di attività |
| **Descrizione:** | Con questo test si vuole vedere il funzionamento generico dell’applicativo se tutti i test precedenti sono andati a buon fine anche questo non darà problemi | | |
| **Prerequisiti:** | avere il database funzionante e la gui che funziona | | |
| **Procedura:** | 1. Fare login con un utente esistente 2. Usare un po’ l’applicazione e eseguire varie azioni | | |
| **Risultati attesi:** | Che il programma funzioni correttamente . | | |

## Risultati test

Tabella

|  |  |  |
| --- | --- | --- |
| **Codice test** | **Risultato** | **Note** |
| **TC-001** | PASSATO | Nessuna |
| **TC-002** | PASSATO | Nessuna |
| **TC-003** | PASSATO | Nessuna |
| **TC-004** | PASSATO | Nessuna |
| **TC-005** | PASSATO | Nessuna |
| **TC-006** | PASSATO | Nessuna |
| **TC-007** | PASSATO | Nessuna |
| **TC-008** | PASSATO | Nessuna |
| **TC-009** | PASSATO | Nessuna |
| **TC-010** | PASSATO | Nessuna |

## Mancanze/limitazioni conosciute

Descrizione con motivazione di eventuali elementi mancanti o non completamente implementati, al di fuori dei test case. Non devono essere riportati gli errori e i problemi riscontrati e poi risolti durante il progetto.

# Consuntivo

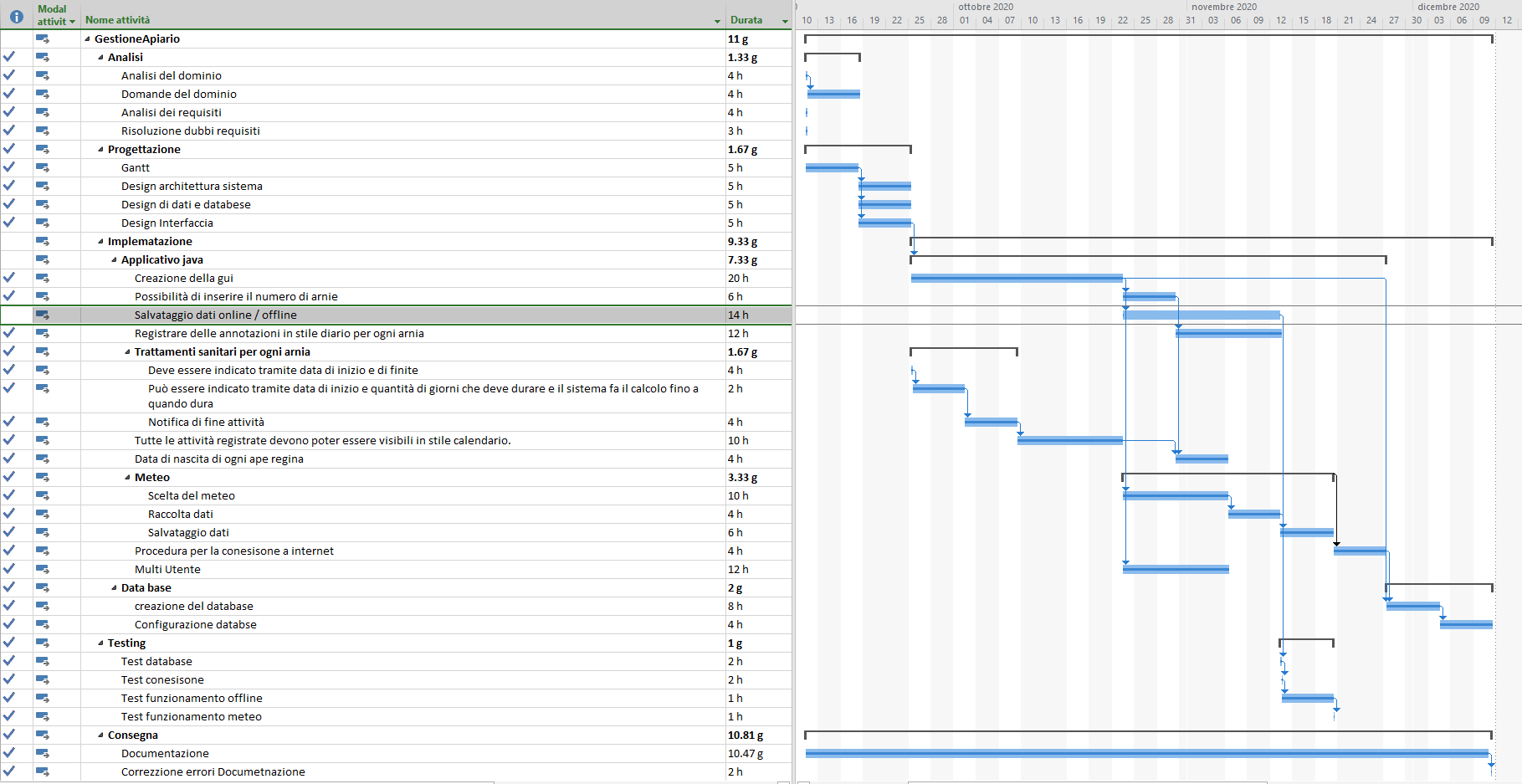


Figura Gantt consuntivo come prossimo vedere la parte di sviluppo della GUI ha portato via molto più tempo del previsto e la registrazione delle note impiegato più tempo di quanto stipulato.

# Conclusioni

Il mio progetto probabilmente non cambierà il mondo, ma se pensiamo che prima di fare questo progetto non esisteva un soluzione pratica e veloce per la gestione digitale delle argine. Non so se verrà mai utilizzato da qualcuno ma spero che un giorno potrò avere un feedback da parte di qualcuno che lo usa per lavoro. Come in ogni progetto ho imparato tanto di nuovo e migliorato tecniche che già conoscevo.

## Sviluppi futuri

Avevo già programmato degli upgrade come si può vedere da codice, la possibilità avere il software che funziona sia online che offline e ogni volta che può carica i dati su un server online. Sotto la cartella desing si può trovare un bozza di schema di flusso.

## Considerazioni personali

Questo progetto è stato utile per imparare l'utilizzo di Sqlite e a conoscere meglio l’utilizzo dei database ma anche di java e delle sue numerose librerie, rimane comunque un progetto davvero molto interessante che con più tempo come già detto si potrebbe migliorare molto.

# Bibliografia

## Sitografia

* <https://stackoverflow.com/>, Stackoverflow, 17-12-2020.
* <https://www.w3schools.com/> w3schools 17-12-2020
* <https://openweathermap.org/> openweathermap 17-12-2020

# Allegati

* Diari di lavoro
* Prodotto