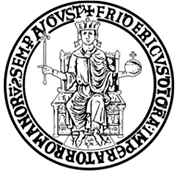
UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

SCUOLA POLITECNICA E DELLE SCIENZE DI BASE

DIPARTIMENTO DI INGEGNERIA ELETTRICA E TECNOLOGIE DELL’INFORMAZIONE



CORSO DI LAUREA IN INFORMATICA

INSEGNAMENTO DI BASI DI DATI E SISTEMI INFORMATIVI I

ANNO ACCADEMICO 2021/2022

**Documento di progettazione**

**Gestione Riferimenti Bibliografici**

*Autori: Docente:*

Francione Giuseppe

Matricola: N86003734 Prof. Sergio Di Martino

[gius.francione@studenti.unina.it](mailto:gius.francione@studenti.unina.it)

Ascolese Alessia

Matricola: N86003704

[ale.ascolese@studenti.unina.it](mailto:ale.ascolese@studenti.unina.it)

Gruppo: OOBD2122\_34

URL GitHub: <https://github.com/Snesnopic/Progetto-bibliografie>

Sommario

[1 Introduzione 3](#_Toc96355099)

[1.1 Scopo 3](#_Toc96355100)

[1.2 Panoramica del sistema 3](#_Toc96355101)

[1.3 Obiettivi di design 3](#_Toc96355102)

[1.4 Definizioni 3](#_Toc96355103)

[2 Panoramica del progetto 4](#_Toc96355104)

[2.1 Introduzione 4](#_Toc96355105)

[2.2 Ambiente 4](#_Toc96355106)

[2.2.1 Struttura ad alto livello 4](#_Toc96355108)

[2.2.2 Sotto-sistema del datalpkg 5](#_Toc96355109)

[2.2.3 Sotto-sistema del ctrlpkg 6](#_Toc96355110)

[2.2.4 Sotto-sistema del guipkg 7](#_Toc96355111)

[2.3 CRC Cards 8](#_Toc96355112)

[3 GUI 12](#_Toc96355113)

[3.1 Mockup 12](#_Toc96355114)

[3.2 Interfacce 15](#_Toc96355115)

[4 Progettazione strutturale 19](#_Toc96355116)

[4.1 Spiegazione del progetto 19](#_Toc96355117)

[4.2 Class Diagram completo 19](#_Toc96355118)

[4.3 Sequence Diagram 21](#_Toc96355119)

# Introduzione

## Scopo

Questo documento ha l’intento di introdurre i contenuti e le motivazioni del design dell’applicazione di gestione riferimenti bibliografici.

## Panoramica del sistema

L’intento del sistema è di poter comunicare con il database dei riferimenti bibliografici tramite un’interfaccia intuitiva, e di poter comunicare al 100% con i dati in esso contenuti.

## Obiettivi di design

Gli obiettivi di design sono di poter avere un sistema che permette piena compatibilità con diverse interfacce grafiche e database driver, entrambe intercambiabili con una suddivsione in package e con un approccio Object Oriented.

## Definizioni

Con “riferimento” si intende un qualsiasi documento, libro, saggio, rivista, articolo o pubblicazione su un particolare argomento.

# Panoramica del progetto

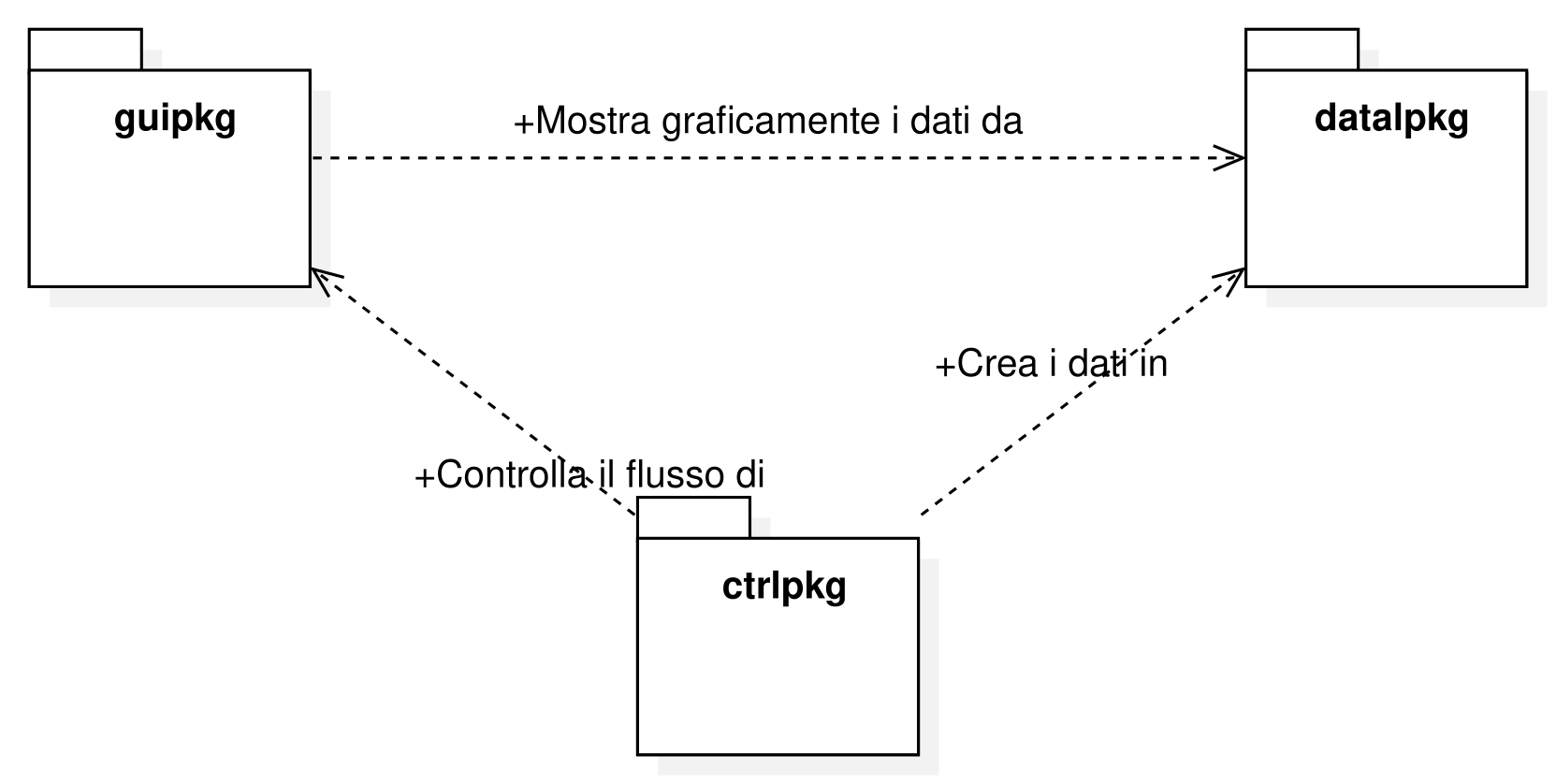
## Introduzione

L’approccio di design utilizzato è quello di un sistema Object Oriented sviluppato in Java che dipende strettamente da un database PostgreSQL.

## Ambiente

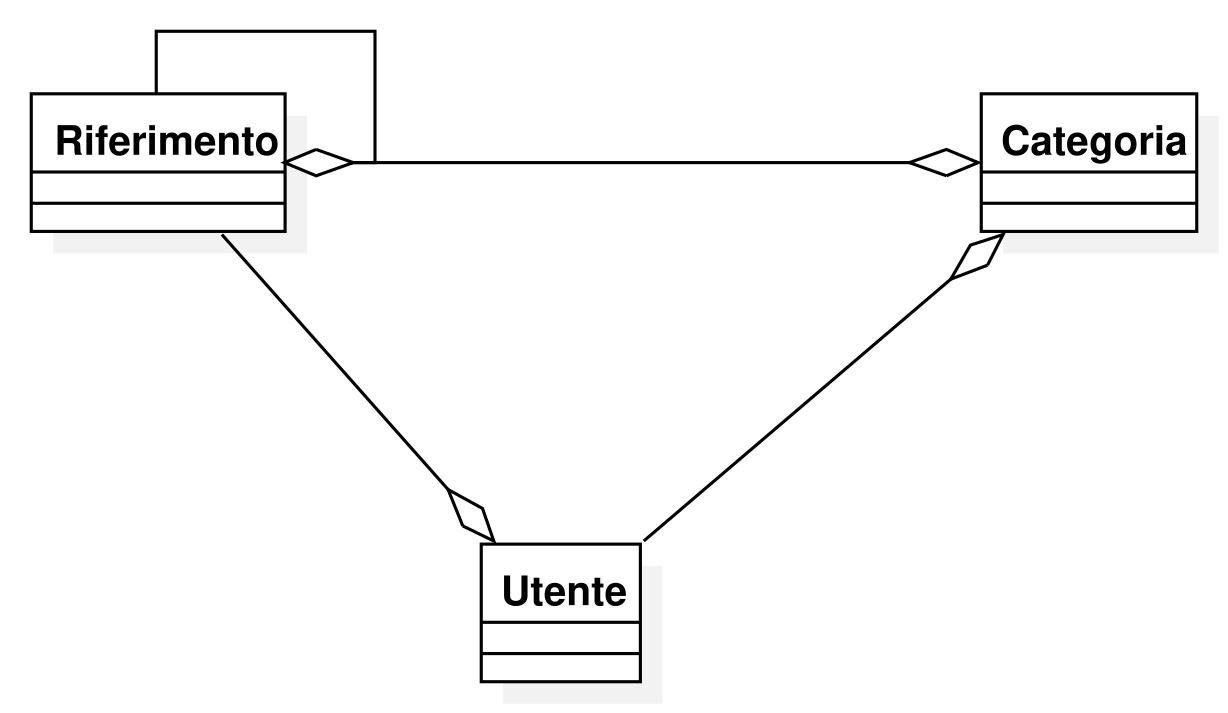
L’ambiente di sistema è un qualsiasi sistema operative non-mobile (quindi desktop) fornito di una connessione al database.

### Struttura ad alto livello



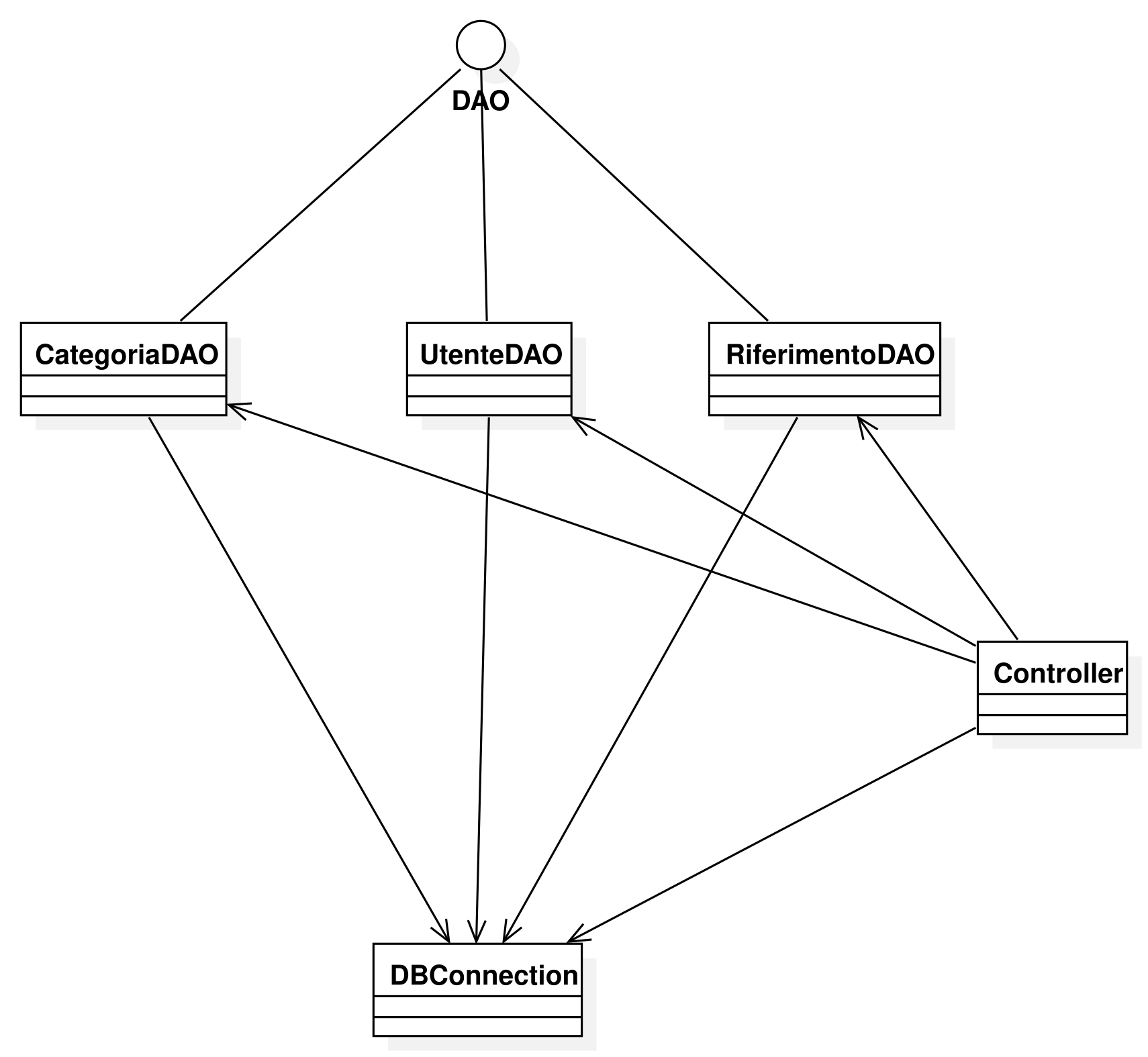
Il sistema è costituito da 3 elementi principali, rappresentati in package: **guipkg**, per la definizione delle interfacce grafiche e le loro interazioni; **datalpkg**, per la definizione delle classi di dati che andranno trattati e mostrati; e **ctrlpkg**, per la definizione dei collegamenti e delle varie interazioni tra sistema e database esterno.

### Sotto-sistema del datalpkg



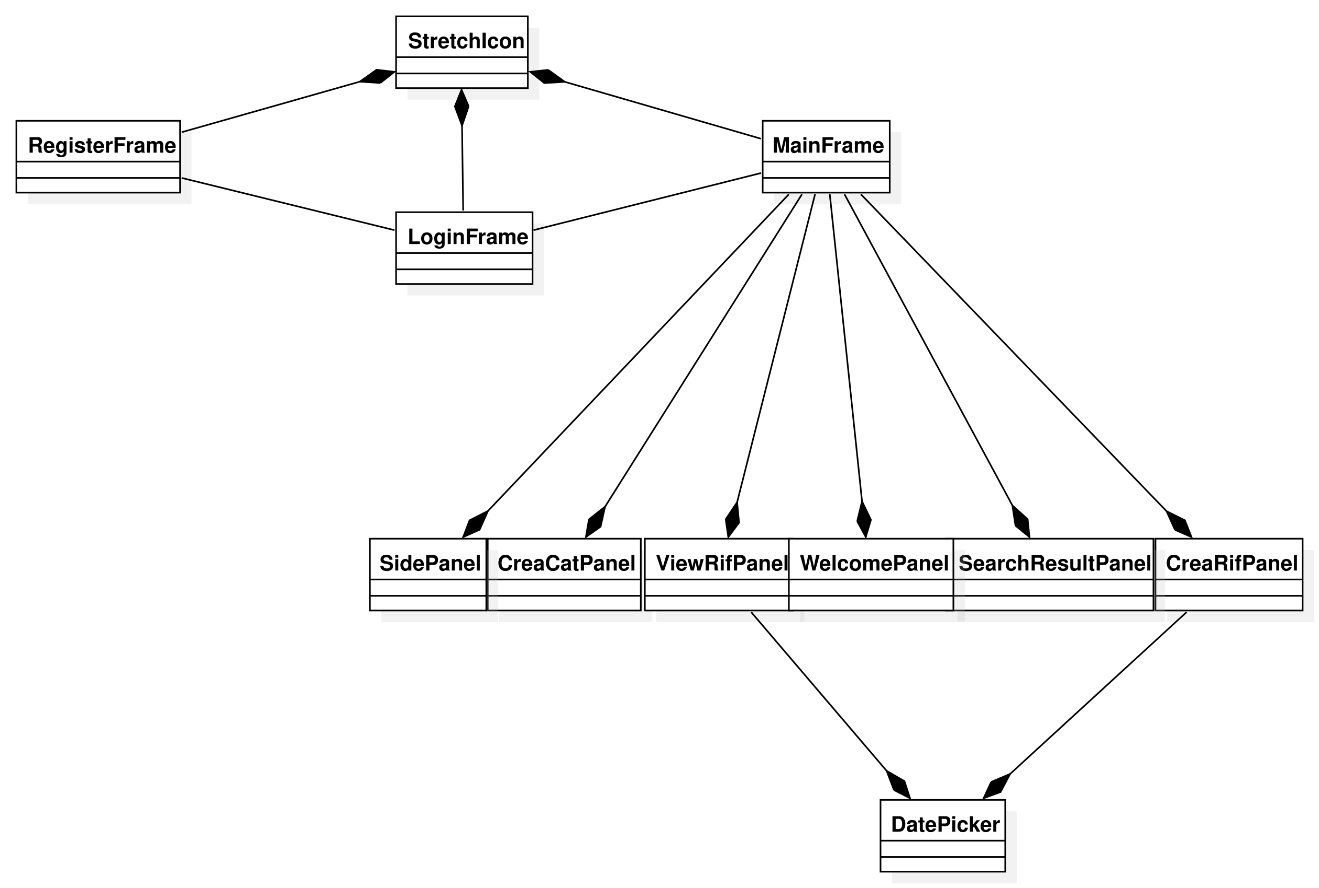
Il datalpkg definisce le classi di dati che verranno manipulate per flusso di dati tra database esterno e programma. Inoltre queste classi posseggono dipendenze tra di loro.

### Sotto-sistema del ctrlpkg

**

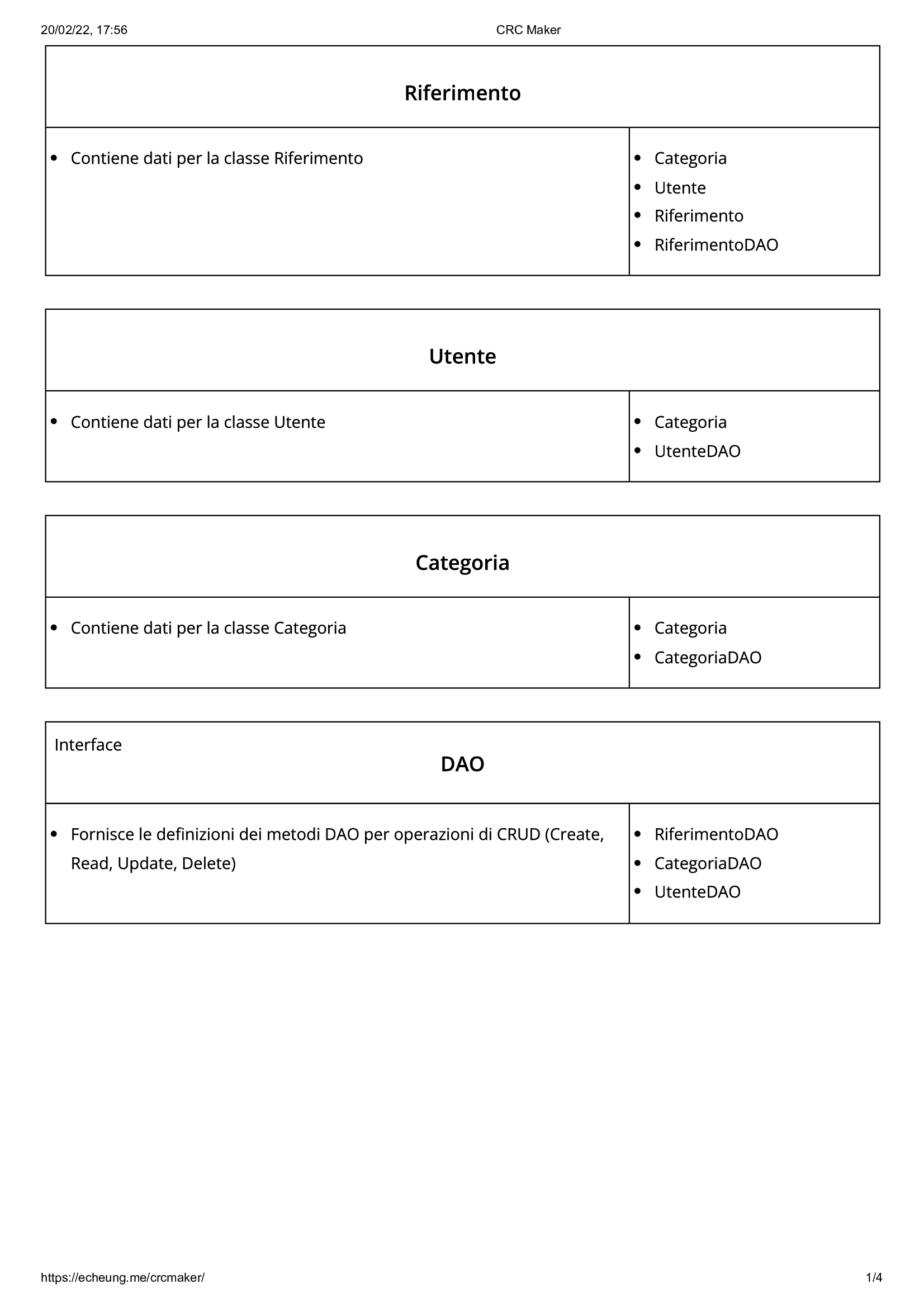
Il ctrlpkg definisce le classi che controllano il flusso di dati tra il programma e il database, e nel programma stesso. Viene definite un’interfaccia DAO e da essa 3 implementazioni per le 3 classi di dati nel datalpkg. Ognuna di questa si connette alla classe singleton DBConnection che dà a queste gli strumenti necessari per avere una connessione affidabile al database. Infine la classe Controller ha i metodi per gestire tutto il flusso del programma, potendo istanziare i vari DAO e connettendosi anch’essa con DBConnection.

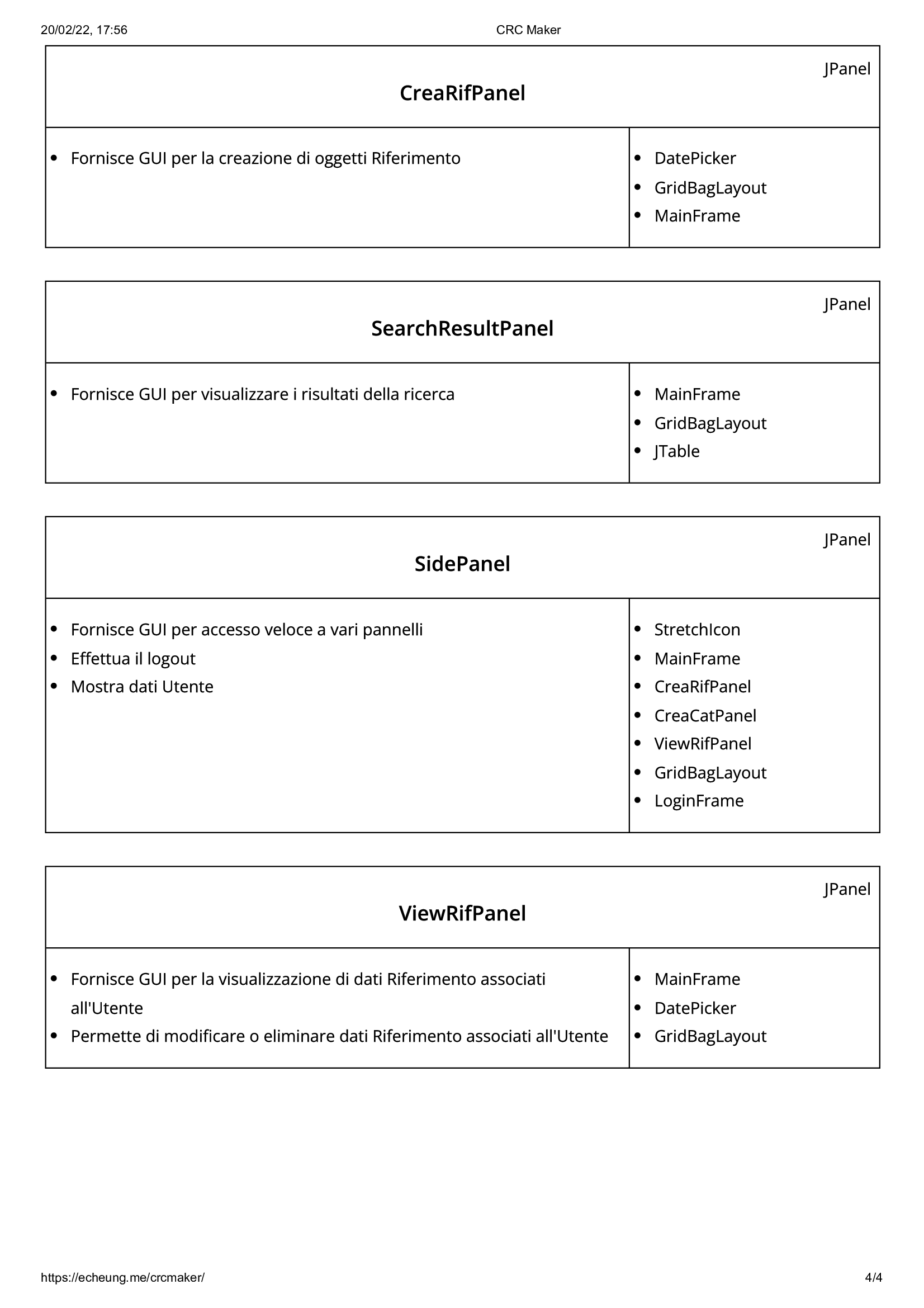
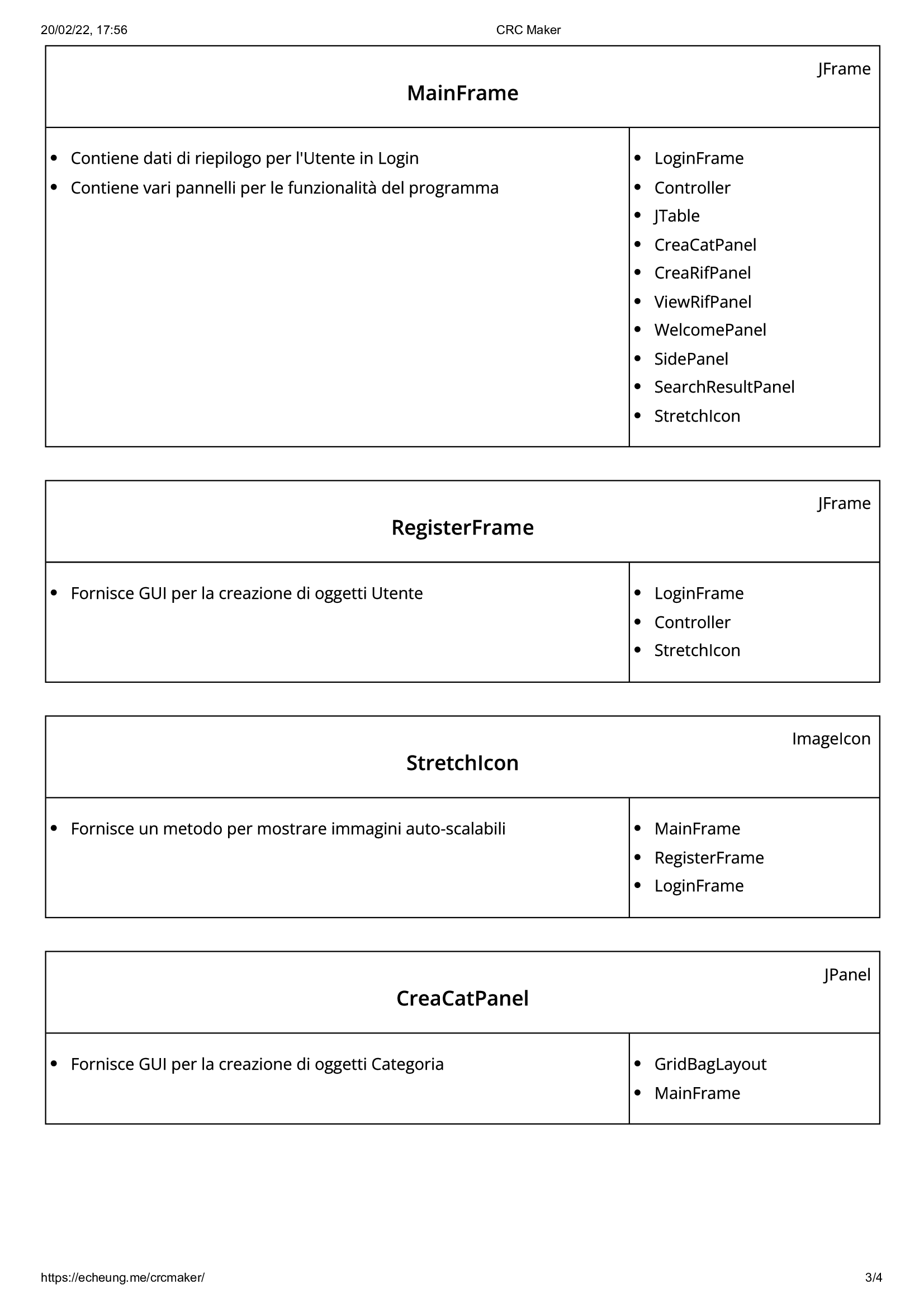
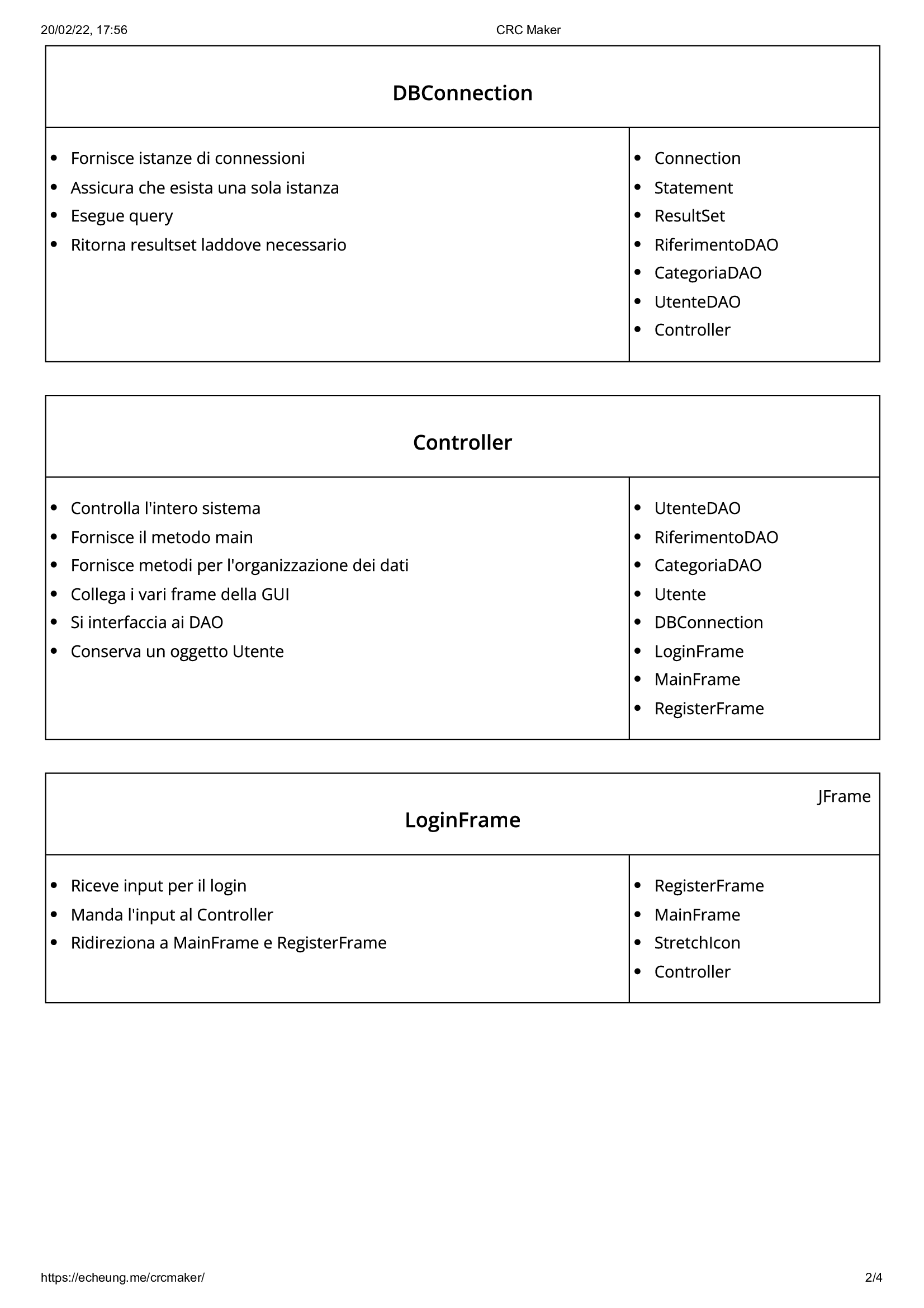
### Sotto-sistema del guipkg



Il guipkg definisce le classi che vengono utilizzate per definire l’interfaccia grafica del programma. Esso è costituito da un LoginFrame da cui inizia il programma, un RegisterFrame e un MainFrame che contiene vari Panel. Ogni Frame contiene la classe StretchIcon per inserimenti dinamici di immagini, e due pannelli che permettono inserimento di date contengono DatePicker per inserimento facilitato di date.

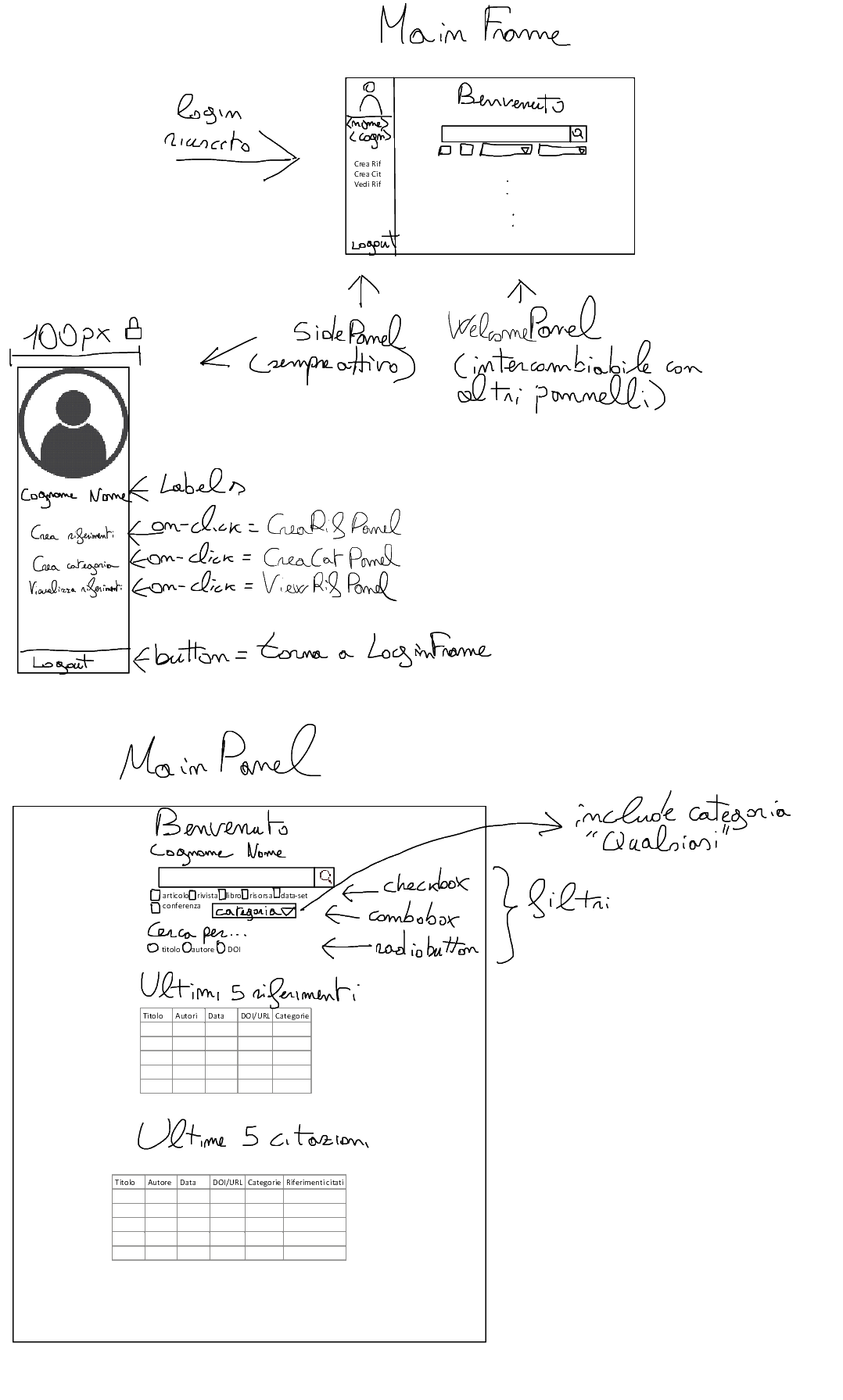
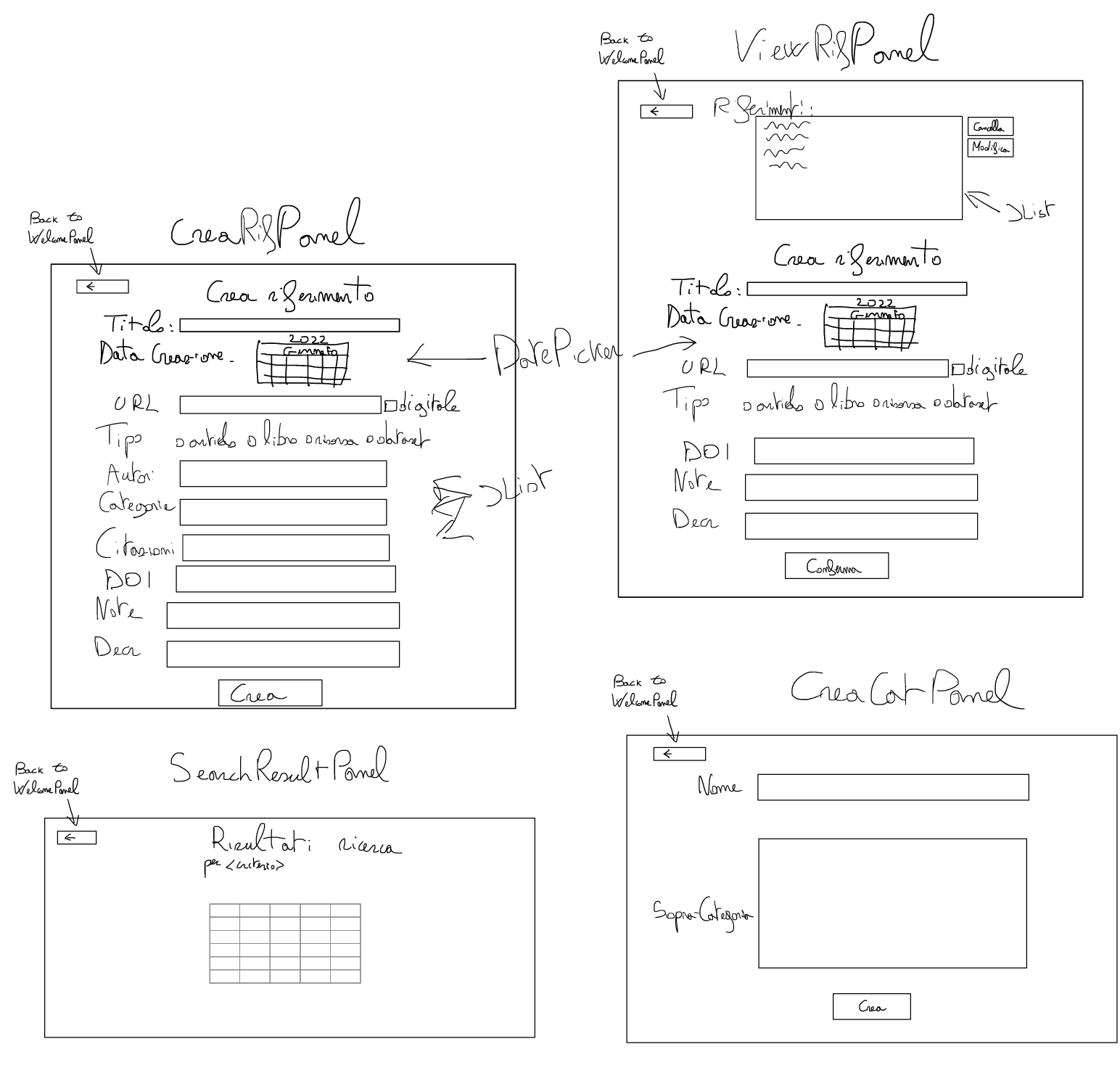
## CRC Cards





# GUI

## Mockup



## Interfacce

*CreaCatPanel*

**

*CreaRifPanel*

**

*LoginFrame*

**

*MainFrame (con SidePanel a sinistra e WelcomePanel a destra)*

**

*RegisterFrame*

*ViewRifPanel*

# Progettazione strutturale

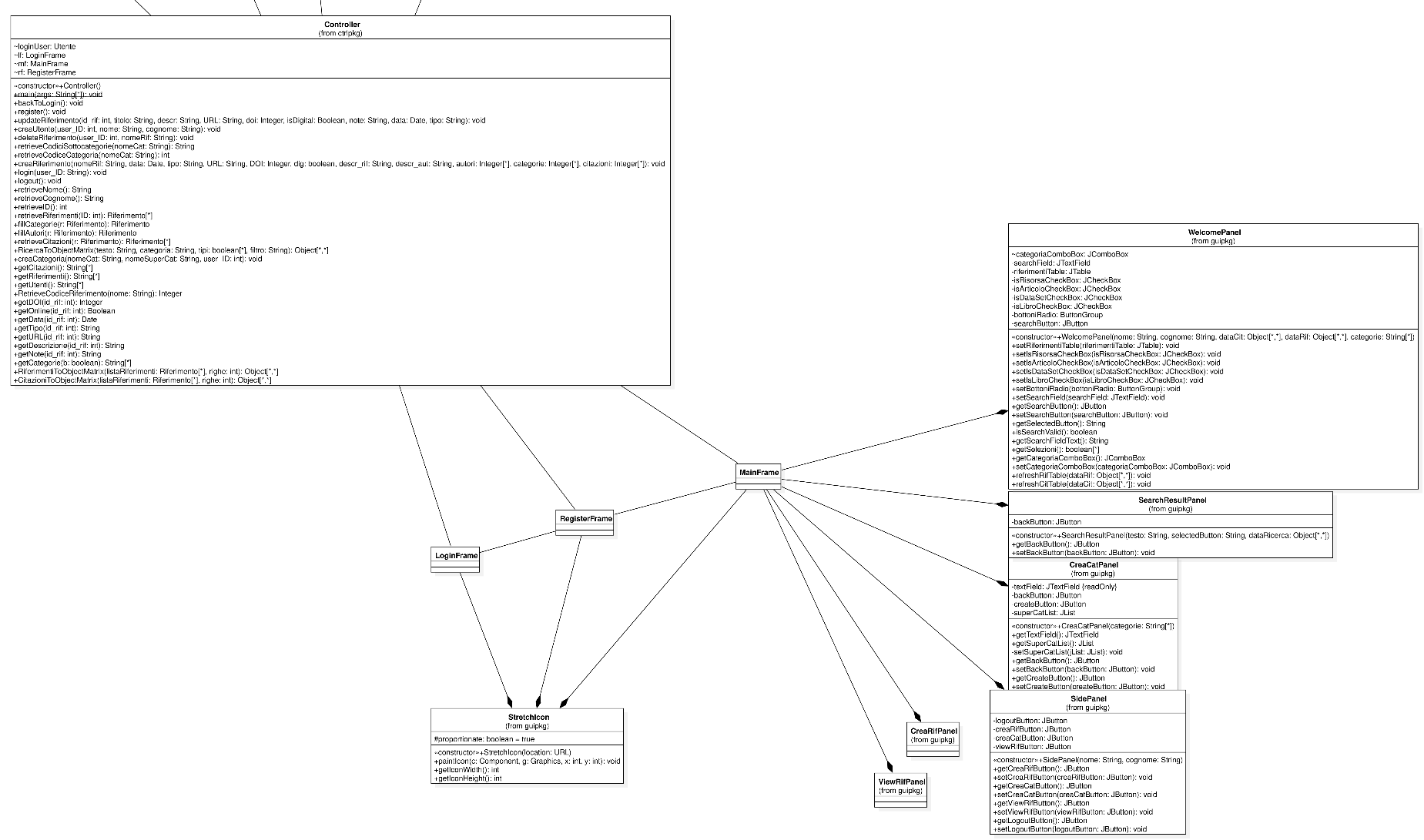
## Spiegazione del progetto

Il sistema implementa un oggetto di classe Controller, che inizialmente istanza un oggetto LoginFrame (reso visibile), un oggetto MainFrame e un oggetto RegisterFrame, quest’ultimi invisibili. Si può passare da LoginFrame a RegisterFrame e viceversa, oppure da LoginFrame e MainFrame e viceversa. La classe Controller inoltre definisce i metodi che usano i vari pannelli collegati a MainFrame, che restituiscono i dati da mostrare organizzati nel modo richiesto. Ad esempio, in WelcomePanel del MainFrame sono presenti due JTable che richiedono una matrice di Object. Quindi il MainFrame, collegato al Controller, invoca il metodo RiferimentiToObjectMatrix, che restituirà la matrice.

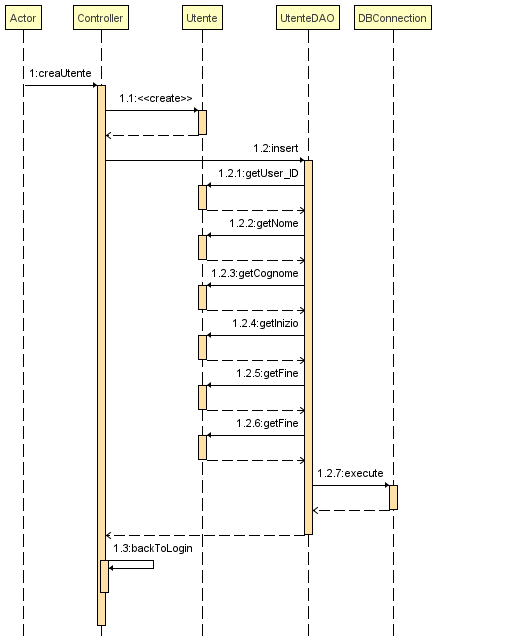
Il Controller si college ai DAO per richiedere gli oggetti delle classi di dati necessarie (Riferimento, Categoria e Utente). I vari DAO, che implementano le funzioni CRUD definite dall’interfaccia DAO, si collegano al database tramite la classe singleton DBConnection, che utilizza il driver PostgreSQL (facilmente intercambiabile) con credenziali fisse e metodo statico per ricavare l’istanza di connessione. I DAO quindi manipolano i dati prelevati dal database per creare gli oggetti delle classi a cui sono associate.

## Class Diagram complete

## 



## Sequence Diagram



*Funzione*

*void creaUtente(int user\_id,String nome,String cognome) nella classe Controller*

## 

*Funzione*

*Object[][] RiferimentiToObjectMatrix(List<Riferimento> listaRiferimenti, int righe) nella classe Controller*