Università degli Studi di Salerno

Corso di Ingegneria del Software

COMMERCIALISTA

Object Design Document

VERSIONE 1.0

Data: 21/10/2016

Partecipanti:

Nome	Matricola
Mazzillo Pasquale	05121/02562
De Luca Danilo	05121/02242

Cronologia delle revisioni:

Data	Versione	Descrizione	Autore
09/09/16	0.1	Stesura Trade-Off dell'Object Design	Pasquale Mazzillo Danilo De Luca
11/09/16	0.2	Stesura Documentazione delle Interfacce	Pasquale Mazzillo Danilo De Luca
13/09/16	0.3	Stesura Interfacce delle Classi	Pasquale Mazzillo Danilo De Luca
15/09/16	0.4	Revisione Trade-Off dell'Object Design	Pasquale Mazzillo Danilo De Luca
16/09/16	0.5	Revisione Documentazione delle Interfacce	Pasquale Mazzillo Danilo De Luca
17/09/16	0.6	Stesura classi Frame e classi Diagram	Pasquale Mazzillo Danilo De Luca
20/09/16	0.7	Revisione classi Frame e classi Diagram	Pasquale Mazzillo Danilo De Luca
29/09/16	1.0	Revisione finale	Pasquale Mazzillo Danilo De Luca

INDICE

1. Introduzione

1.1. Trade-Off dell'Object Design

1.1.1. Modularità vs. Efficienza

Il documento Commercialista si contraddistingue per una vasta modularità , grazie a cui il sistema gode di grande indipendenza nella gestione delle sue caratteristiche. In parallelo, questa modularità vieni a trovarsi in contrasto con l'indispensabile efficienza nelle elaborazioni lato server. Si è deciso di favorire la modularità, piuttosto che l'efficienza, in quanto essa facilita molto l'implementazione e la manutenzione del software e la sua comprensione sia da parte dei programmatori intenzionati a modificarlo, sia da parte degli utenti.

1.1.2. Portabilità vs. Efficienza

Si è stabilito di dare molto rilievo alla portabilità del sistema Commercialista. Dunque, si è optato per l'utilizzo del linguaggio php ,javascript ,html ,css come linguaggi di programmazione con cui costruire il software, ma ciò penalizza l'efficienza del sistema stesso. Si è giunti, quindi, ad un compromesso indispensabile, a causa dei numerosi supporti forniti dai linguaggi utilizzati nello sviluppo di sistemi avanzati e dei numerosi vantaggi di un sistema con grande efficienza. Affinché il linguaggio non perda le caratteristiche di portabilità, si è deciso di attenersi ad alcune regole base per rendere il codice portabile, qualità di fondamentale importanza per un programma.

1.1.3. Spazio di Memoria vs. Tempo di Risposta

Siccome il sistema Commercialista sarà un sistema software utilizzato per rendere più veloci e semplici le operazioni effettuate da uno studio di commercialisti ,è parso opportuno ottimizzare il più possibile il tempo di risposta per ogni funzione, piuttosto che ottimizzare lo spazio usato, dato che lo spazio di memoria per il sistema Commercialista non è il problema principale.

1.2. Linee Guida per la Documentazione delle Interfacce

Sono state definite varie convenzioni che saranno rispettate dagli sviluppatori del sistema durante la fase di implementazione del software Commercialista.
Tali convenzioni riguardano, in particolare, i nomi delle classi, metodi, campi e variabili che faranno parte del codice,

*

Classi

- Ogni classe deve avere un nome che sia composto da uno o più sostantivi singolari della lingua italiana o della lingua inglese, ognuno dei quali deve avere la lettera della prima parola in minuscolo e la lettera della seconda parola in maiuscolo; tale nome non deve presentare spazi, punteggiatura o altri caratteri diversi da quelli alfabetici.
- Ogni classe deve avere un nome che sia significativo per il suo scopo relativamente al software di cui fa parte.
- Ogni classe deve avere un nome che sia diverso da quelli delle altre classi, e dei metodi.
- Ogni classe che non è un frame e che si occupa di gestione ed operazioni relativi a un determinato tipo di entità (dati persistenti) deve avere il nome di tale tipo come prefisso del proprio nome.

1.3.1. Definizioni

CLIENTE	Colui che utilizza il sistema per ottonere informazioni sulle pratiche,sugli avvisi,sulle newsletter definite dal commercialista
COMMERCIALISTA	Colui che interagisce con il sistema al fine di manipolare i dati delle pratiche , delle newsletter,degli avvisi per i clienti registrati al sito
STUDIO	Insieme delle informazioni riguardanti pratiche,newsLetter,avvisi e dati dei clienti

AVVISI	Informazione a lungo termine data dal commercialista a tutti gli iscritti
NEWSLETTER	Informazioni a breve termine data dal commercialista al cliente

Interfacce delle classi

AVVISO

AVVISO
-codice_avviso: Int; -titolo: String -data: Date -visibilita: String -testo: String
+ getCodice(): Int + settaAvviso(Int codice_avviso): void + getTitolo(): String + getData (): String + getVisibilita (): String

+ getTesto (): String + eliminaAvviso(Int codice_avviso): void + modificaAvviso(Int codice_avviso, String titolo,String data,String visibilita,String testo): void + aggiungiAvviso(String titolo,String data,String visibilita,String testo): void + elencaAvvisi(): void

Campi

- codice_avviso: Int = Codice univoco dell'avviso
- titolo: String = Titolo dell'avviso
- data: Date = Data di creazione dell'avviso
- visibilita: String = La visibilità dell'avviso
- testo: String = Il testo dell'avviso

Metodi

- getCodice(): Int = Restituisce un intero contenente il codice dell'avviso.
- getTitolo (): String = Restituisce una stringa contenente il titolo dell'avviso.
- getData(): String = Restituisce una stringa contenente la data dell'avviso.
- getVisibilita(): String = Restituisce una stringa contenente la visibilità dell'avviso.
- getTesto(): String = Restituisce una stringa contenente il testo dell'avviso.
- settaAvviso(Int codice_avviso)= Effettua una query al database impostado gli attributi dell'avviso con il risultato della query
- eliminaAvviso(Int codice_avviso) = Elimina l'avviso che ha come codice codice_avviso.
- aggiungiAvviso(String titolo,String data,String visibilita,String testo)= Aggiunge un avviso con i valori passati come parametro.
- modificaAvviso(Int codice_avviso, String titolo,String data,String visibilita,String testo) = Permette di modificare l'avviso che ha come codice codice_avviso.
- elencaAvvisi()= Restituisce un iteratore contenente tutti gli avvisi presenti nel database.

Newsletter

Newsletter
-codice_news: Int; -titolo: String -data: Date -testo: String -codice_cliente: String
+ getCodice(): Int + settaNews(Int codice_avviso): void + getTitolo(): String + getData (): String + getCodiceCliente (): String + getTesto (): String + eliminaNews(Int codice_avviso): void + modificaNews(Int codice_avviso, String titolo,String data,String visibilita,String testo): void + aggiungiNews(String titolo,String data,String visibilita,String testo): void + elencaNews(): void

Campi

- codice_news: Int = Codice univoco della newsletter
- titolo: String = Titolo della newsletter
- data: Date = Data di creazione della newsletter
- testo: String = Il testo della newsletter
- codice_cliente = Il codice fiscale del cliente a cui appartiene la newsletter.

Metodi

- getCodiceNews(): Int = Restituisce un intero contenente il codice della newsletter
- getTitoloNews (): String getTitoloAvviso(): String = Restituisce una stringa contenente il titolo della newsletter.
- getDataNews (): String = Restituisce una stringa contenente il testo della newsletter.
- getTestoNews (): String = Restituisce una stringa contenente il testo della newsletter.
- eliminaNews (Int codice_news): void = Elimina la newsletter che ha come codice codice_news.
- modificaNews (Int codice_news, String titolo, String data, String visibilita, String testo) = Permette di modificare la newsletter che ha come codice codice_news.
- aggiungiNews(String titolo,String data,String visibilita,String testo)= Permette di aggiungere nel database una nuova newsletter.
- elencaNews() = Restituisce in iteratore contenente tutte la newsletter presenti nel database.

Utente

Utente
-codice_fiscale: String; -nome: String; -cognome: String; -data_di_nascita: Date; -professione: String; -email: String; -password: String;
+ getCodiceUtente(): String; + getNomeUtente (): String

+ getCognomeUtente (): String

+ getDataUtente (): void

+ getProfessioneUtente (): String

+ getEmailUtente (): String

+ getPasswordUtente (): String

+ eliminaUtente (Int codice_fiscale): void

+ aggiungiUtente (Int codice_fiscale, String nome, String cognome, String data_di_nascita, String professione, String email, String password): void

+ modificaUtente(Int codice_fiscale, String nome, String cognome, String data_di_nascita, String professione, String email, String password):void

+ elencaUtenti():void.

Campi

- codice_fiscale: String = Codice univoco ell'utente.
- nome: String = Nome dell'utente.
- cognome: String = Cognome dell'utente.
- data_di_nascita: Date = Data di nascita dell'utente.
- professione: String = La professione dell'utente.
- email: String = E-mail dell'utente.
- password: String = Password dell'utente per accedere al sito.

Metodi

- getCodiceUtente() = Restituisce il codice fiscale.
- getNomeUtente ()= Restituisce il nome.
- getCognomeUtente ()= Restituisce il cognome.
- getDataUtente ()= Restituisce la data di nascitaa.
- getProfessioneUtente ()= Restituisce la professione.
- getEmailUtente ()= Restituisce l'email.
- getPasswordUtente ()= Restituisce la password.

- eliminaUtente (Int codice_fiscale) = permette di eliminare un utente dal database.
- aggiungiUtente (Int codice_fiscale, String nome, String cognome, String data_di_nascita, String professione, String email, String password) = Permette di aggiungere un nuovo utente del database.
- modificaUtente (Int codice_fiscale, String nome, String cognome, String data_di_nascita, String professione, String email, String password) = Permette di modificare i dati di un utente.
- elencaUtenti()= Restituisce un iteratore contenente tutti gli utenti presenti nel database.

Pratica

Pratica
-codice_pratica: int; -stato: String; -data: Date; -codice_cliente: String; -testo: String;
+ getCodicePratica(): String; + getStatoPratica (): String + getDataPratica (): String + getClientePratica (): String

+ getTestoPratica (): String + eliminaPratica (Int codice_pratica): void + aggiungiPratica (Int codice_pratica, String stato, Date data, String codice_cliente, String testo): void + elencaPratiche(): void + elencaPratiche(String codice_cliente): void

Campi

- codice_pratica = Codice univoco della pratica.
- stato = indica se la pratica è conclusa o no.
- data = data di creazione della pratica.
- codice_cliente = codice fiscale del cliente a cui è assegnata la pratica.
- testo = Testo della pratica.

Metodi

- getCodicePratica() = Restituisce il codice della pratica.
- getStatoPratica ()= Restituisce lo stato della pratica.
- getDataPratica ()= Restituisce la data di creazione della pratica.
- getClientePratica ()= Restituisce il codice fiscale del cliente.
- getTestoPratica ()= Restituisce il testo della pratica.
- eliminaPratica (Int codice_pratica) = Permette di eliminare una pratica dal database.
- aggiungiPratica (Int codice_pratica, String stato, Date data, String codice_cliente, String testo) = Permette di aggiungere una pratica nel database.
- elencaPratiche() = Restituisce un iteratore contenente tutte la pratiche.
- elencaPratiche(String codice_cliente) = Restituisce un iteratore contentente tutte le pratiche del Cliente.

CLASSI FRAME

La classe home presenta un frame dove è possibile effettuare un login oppure una registrazione di un nuovo utente

HOME
- email_login: text. - password_login: text. - button_login: button. - nome_registrazione: text. - cognome_registrazione: text. - codice_fiscale_registrazione: text. - data_nascita_registrazione: text. - email_registrazione: text. - password_registrazione: text. - button_registrazione: button.
+ registrazione (String nome_registrazione, String cognome_registrazione, String codice_fiscale_registrazione, Date data_nascita_registrazione, String email_registrazione, String password_registrazione): void + login(String email_login, String password_login): String

La classe commercialistaForm presenta un frame con il quale il commercialista può effettuare diverse operazioni

COMMERCIALISTAFORM
- esci_button: button. - button_commercialista_newsletter: button. - button_commercialista_avvisi: button. - button_commercialista_pratica: button. - button_commercialista_clienti: button. - button_commercialista_modifica: button.
+ logout(): void + creaAvvisi (): void

+ creaNews (): void + creaPratiche (): void + creaModifica (): void + creaClienti (): void

La classe commercialistaRisposte crea dei frame per le diverse operazioni disponibili al commercialista.

COMMERCIALISTARISPOSTE
- titolo_avviso: text. - data_avviso: text. - visibilita_avviso: text. - testo_avviso: text. - aggiungi_avviso_button: button. - elimina_avviso_button: button. - modifica_avviso_button: button. - titolo_news: text. - data_ news: text. - codice_cliente_ news: text. - testo_ news: text. - aggiungi_ news _button: button. - elimina_ news _button: button. - modifica_ news _button: button. - stato_pratica: text. - data_ pratica: text. - codice_cliente_ pratica: text.

- testo_ pratica: text.
- aggiungi_ pratica _button: button.
- elimina_ pratica _button: button.
- modifica_ pratica _button: button.
- elimina_utente: button.
- codice_fiscale_modifica: text.
- nome_ modifica: text.
- cognome_ modifica: text.
- data_nascita_ modifica: text.
- email_ modifica: text.
- password_ modifica: text.
- modifica_dati_button: button.

+ logout(): void

+ aggiungiAvviso (String titolo_avviso, String visibilita_avviso, String testo_avviso, String data_avviso): void

+ modificaAvviso(String codice_avviso): void

+ eliminaAvviso (String codice_avviso): void

+ aggiungiNews (String titolo_news, String codice_cliente_ news, String testo_ news, String data_ news): void

+ modificaNews(String codice_ news): void

+ eliminaNews (String codice_ news): void

+ aggiungiPratica (String stato_pratica, String codice_cliente_ pratica, String testo_ pratica, String data_ pratica): void

+ eliminaPratica (String codice_ pratica): void

+ eliminaUtente (String codice_ fiscale_utente): void

+ modificaDati(String nome_modifica, String cognome_modifica, String codice_fiscale_modifica, String data_nascita_modifica, String email_modifica, String password_modifica): void.

La classe utenteForm presenta un frame con il quale l'utente può effettuare diverse operazioni

UTENTEFORM
- esci_button: button. - button_utente_newsletter: button. - button_ utente _avvisi: button. - button_ utente _pratica: button. - button_ utente _clienti: button. - button_ utente _modifica: button.
+ logout(): void + creaAvvisi (): void + creaNews (): void + creaPratiche (): void + creaModifica (): void

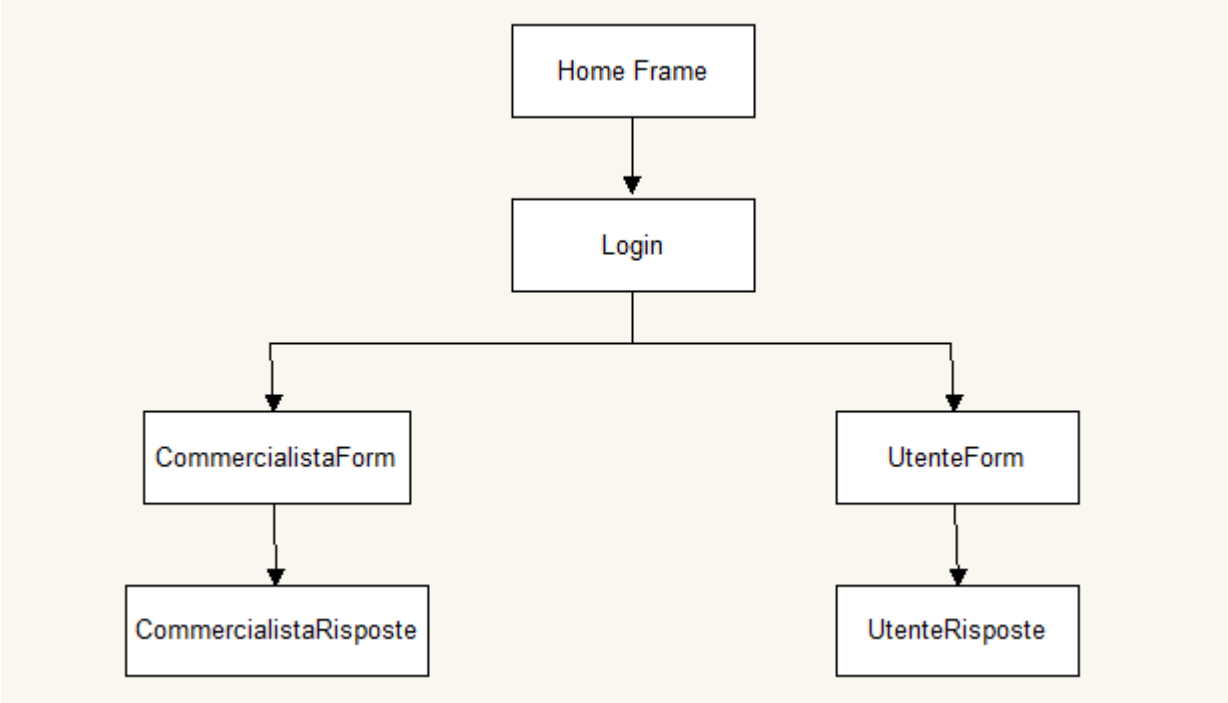
La classe utenteRisposte crea dei frame per le diverse operazioni disponibili all' utente.

UTENTERISPOSTE
- codice_fiscale_modifica: text. - nome_ modifica: text. - cognome_ modifica: text. - data_nascita_ modifica: text. - email_ modifica: text. - professione_modifica: text. - password_ modifica: text.

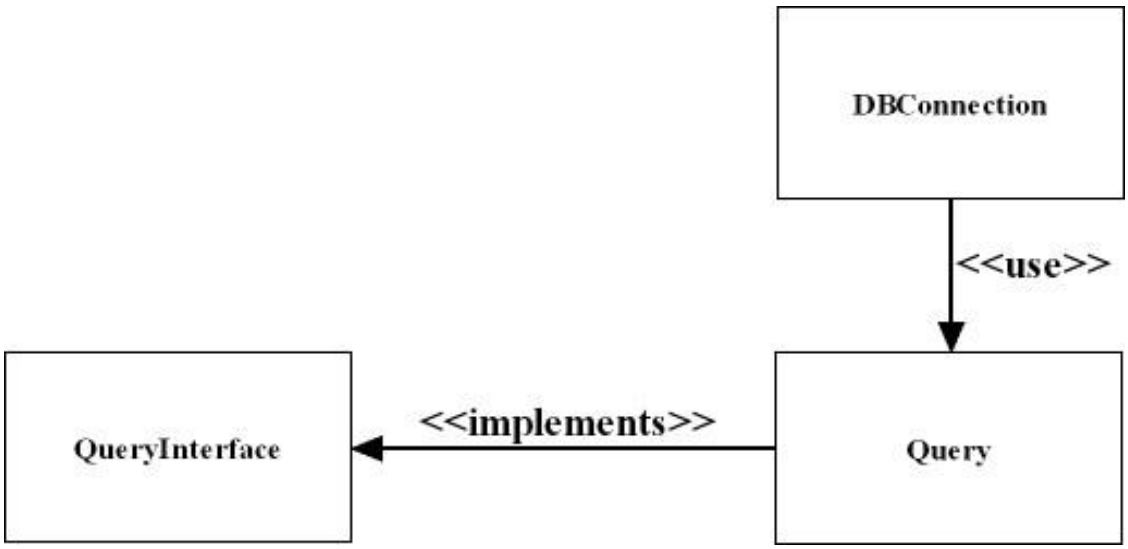
- modifica_dati_button: button.
+ logout(): void + visualizzaAvvisi(): void + visualizzaNews(): void + visualizzaPratica (): void + modificaDati(String nome_modifica, String cognome_modifica, String professione_modifica, String codice_fiscale_modifica, String data_nascita_modifica, String email_modifica, String password_modifica): void.

Diagramma delle classi

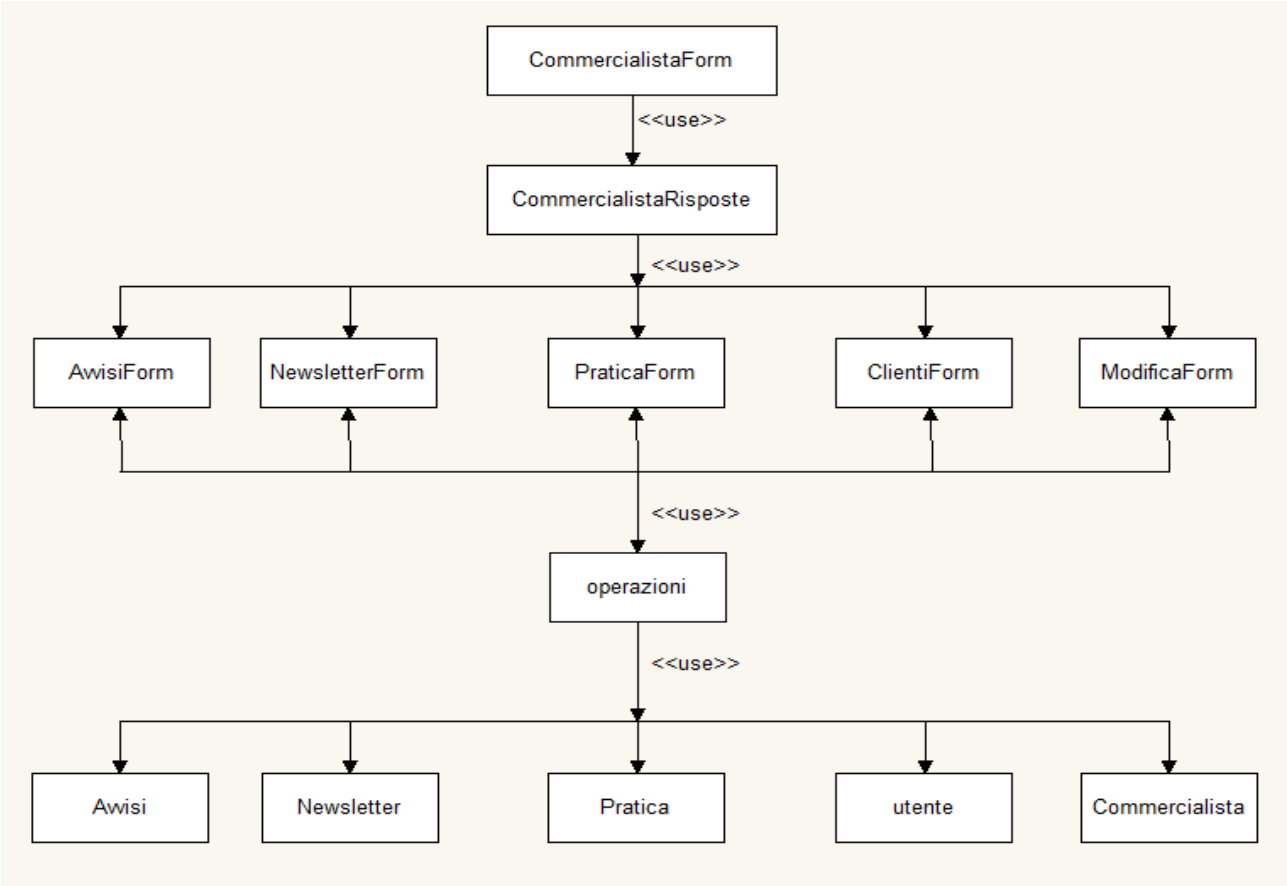
Interfaccia grafica



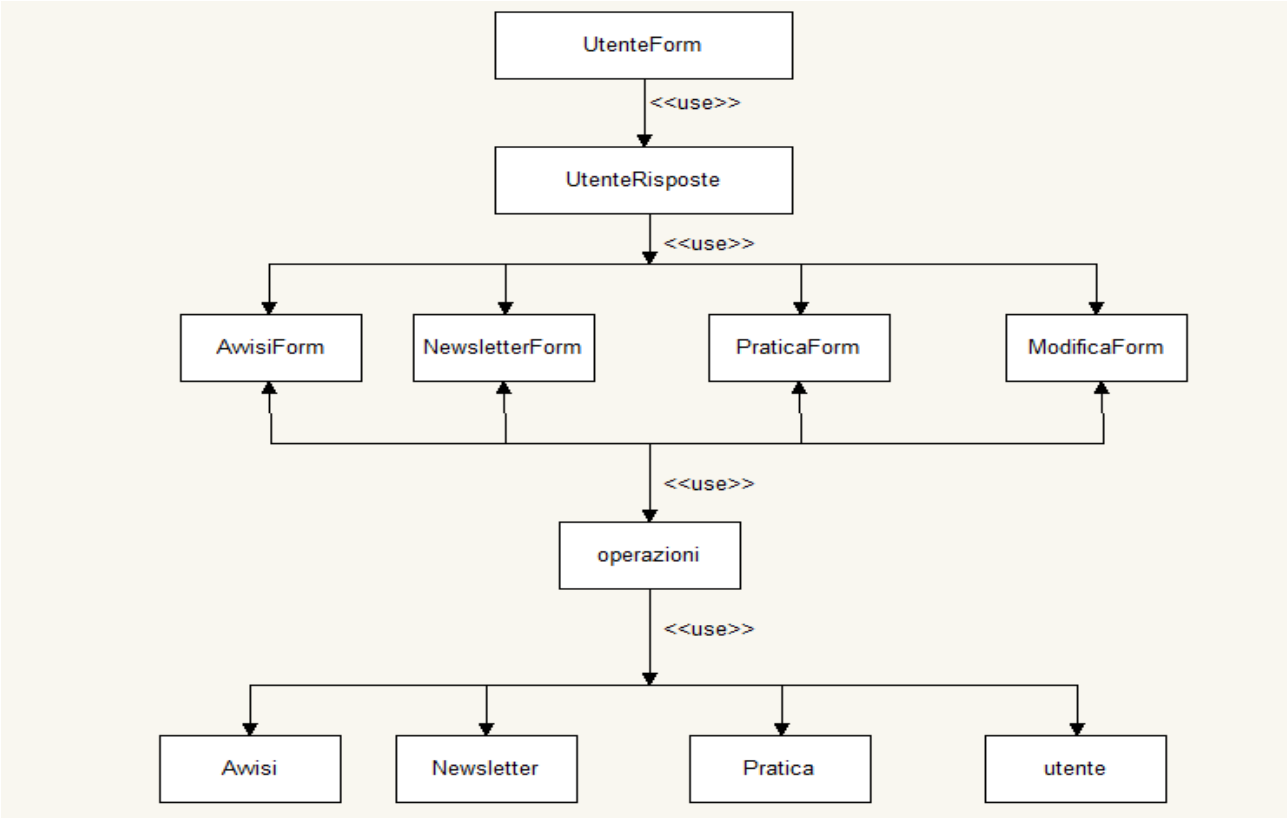
Database



CommercialistaForm


CommercialistaForm
<<use>>
CommercialistaRisposte
<<use>>
AvvisiForm
NewsletterForm
PraticaForm
ClientiForm
ModificaForm
<<use>>
operazioni
<<use>>
Avvisi
Newsletter
Pratica
utente
Commercialista

Utente


UtenteForm
<<use>>
UtenteRisposte
<<use>>
AvvisiForm
NewsletterForm
PraticaForm
ModificaForm
<<use>>
operazioni
<<use>>
Avvisi
Newsletter
Pratica
utente