

e-Sport

Object Design Document

1 Introduzione	1
1.1 Object design trade-offs	1
1.2 Linee guida per la documentazione delle interfacce	2
2 Package	4
2.1 Struttura sistema	5
2.2 Descrizione delle classi	5
2.2.1 Data Layer	5
2.2.1.1 Beans	5
2.2.1.2 Model	6
2.2.2 Application Layer	7
2.2.2.1 Controller	7
2.2.3 Presentation Layer	9
2.2.3.1 View	9

1 Introduzione

1.1 Object design trade-offs

Dopo aver stilato il Requirements Analysis Document e il System Design Document, in cui il sistema è descritto in modo sommario, definendo gli obiettivi ma tralasciando gli aspetti implementativi, si procede ora con lo stilare il documento di Object Design che ha come obiettivo quello di produrre un modello che sia in grado di integrare in modo coerente e preciso tutte le funzionalità individuate nelle fasi precedenti.

In particolar modo, nell'ODD si definiscono le interfacce delle classi, le operazioni, i tipi, gli argomenti e la signature dei sottosistemi definiti in fase di System Design. Inoltre sono specificati i trade-offs e le linee guida.

Comprensibilità vs Tempo:

Il codice del sistema deve essere il più comprensibile possibile in modo da facilitare la fase di testing e migliorare la supportabilità del sito per eventuali future modifiche da apportare.

Per rispettare queste linee guida il codice sarà accompagnato da log e commenti volti a semplificarne la comprensione. Ovviamente questo comporterà un aumento del tempo di sviluppo.

Prestazioni vs Costi:

Poiché il nostro progetto è sprovvisto di budget, per poter mantenere i costi e i tempi di sviluppo verranno utilizzati dei framework open source esterni, in particolare Bootstrap.

Interfaccia vs Usabilità:

L'interfaccia grafica è stata realizzata in modo da essere semplice e con un look minimale così da risultare chiara e concisa.

Verranno utilizzati form e pulsanti con lo scopo di rendere semplice l'utilizzo del sistema da parte dell'utente finale.

Sicurezza vs Efficienza:

La sicurezza, come descritto nei requisiti non funzionali del RAD, rappresenta uno degli aspetti importanti del sistema. Tuttavia, dati i tempi di sviluppo molto limitati, ci limiteremo ad implementare sistemi di sicurezza basati su username, password e ruolo degli utenti.

1.2 Linee guida per la documentazione delle interfacce

Gli sviluppatori dovranno seguire determinate linee guida nella stesura del codice.

1.2.1 Naming Convention

È buona norma utilizzare nomi:

- Descrittivi
- Pronunciabili
- Non abbreviati
- Utilizzando Camel Case
- Utilizzando solo caratteri consentiti

1.2.1.1 Variabili

- I nomi delle variabili devono iniziare con la lettera minuscola e le parole successive, che li compongono, con la lettera maiuscola. In ogni riga di codice vi deve essere una sola dichiarazione di variabile e va effettuato l'allineamento per migliorare la leggibilità.
- In caso di costanti statiche, utilizzare solo caratteri maiuscoli.
- In determinati casi, è possibile utilizzare il carattere underscore “_”, ad esempio quando si fa uso di variabili costanti oppure quando si fa uso di proprietà statiche.

1.2.1.2 Metodi

- I nomi dei metodi devono iniziare con la lettera minuscola, e le parole successive, che li compongono, con la lettera maiuscola. Di solito il nome del metodo è costituito da un verbo che identifica un'azione, seguito dal nome di un oggetto.
- I nomi dei metodi per l'accesso e la modifica delle variabili dovranno essere del tipo `getNomeVariabile()` e `setNomeVariabile()`.
- Ai metodi va aggiunta una descrizione che deve essere posizionata prima della dichiarazione del metodo e che deve descriverne lo scopo, i parametri, il valore di ritorno ed eventualmente le eccezioni che può lanciare. I metodi devono essere raggruppati in base alla loro funzionalità.

1.2.1.3 Classi e Pagine (JSP/HTML)

- I nomi delle classi e delle pagine devono iniziare con la lettera maiuscola, e anche le parole successive all'interno del nome devono iniziare con la lettera maiuscola. I nomi delle classi e delle pagine devono essere evocativi in modo da fornire informazioni sullo scopo di quest'ultime.
- Ogni file sorgente *.java deve contenere una singola classe e deve essere strutturato in un determinato modo:

1. Introduzione alla classe:

```
/**
```

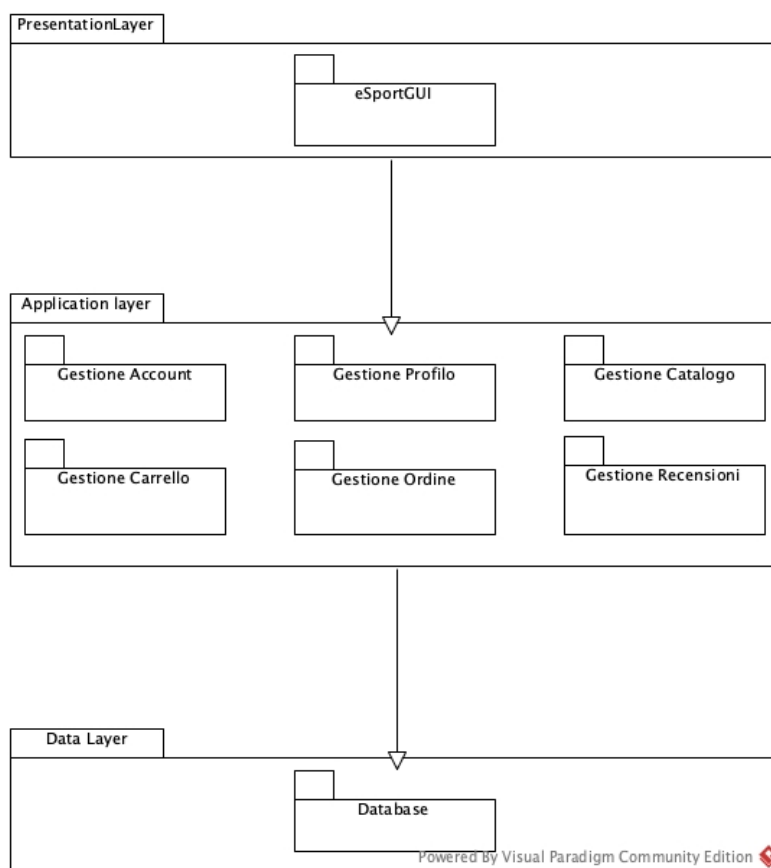
* Astrazione modellata dalla classe
**/

2. Dichiarazione della classe, costituita da:

- Dichiarazione della classe pubblica
- Dichiarazione delle costanti
- Dichiarazione delle variabili d'istanza
- Dichiarazione di variabili statiche
- Costruttore di default
- Dichiarazione metodi che definiscono il comportamento della classe

2 Package

La struttura del sistema è modellata secondo una divisione in package e sotto-package che raggruppano le classi che hanno il compito di gestirne la logica in base alle richieste dell'utente.



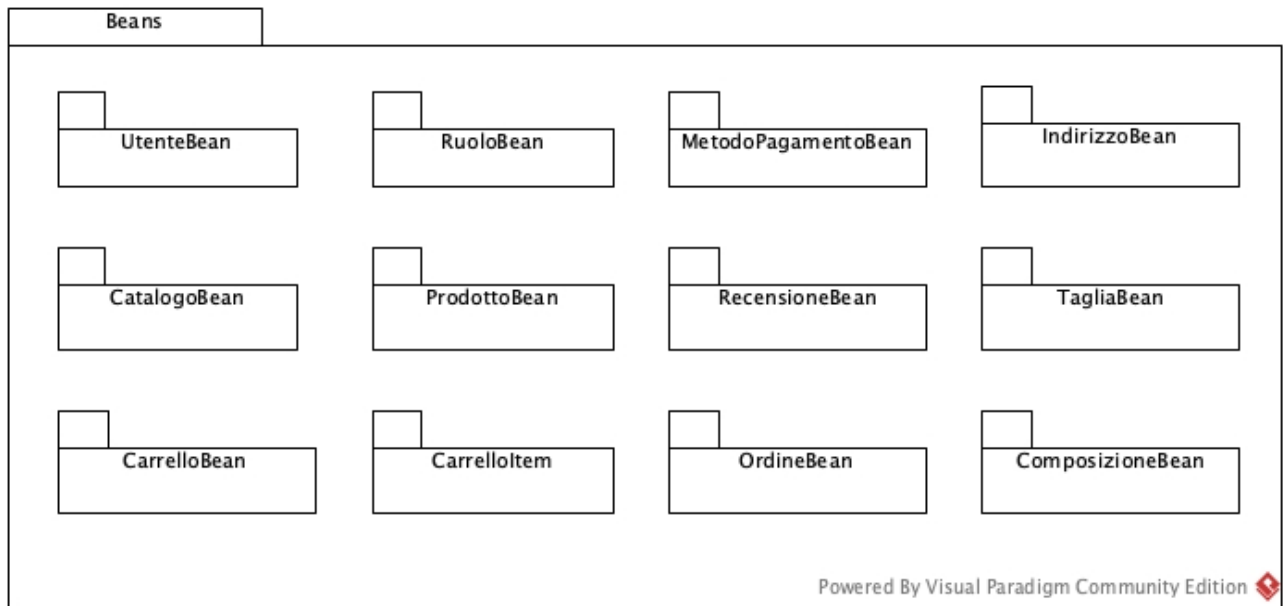
2.1 Struttura sistema

2.2 Descrizione delle classi

2.2.1 Data Layer

2.2.1.1 Beans

Contiene le classi che modellano le entità del sistema.

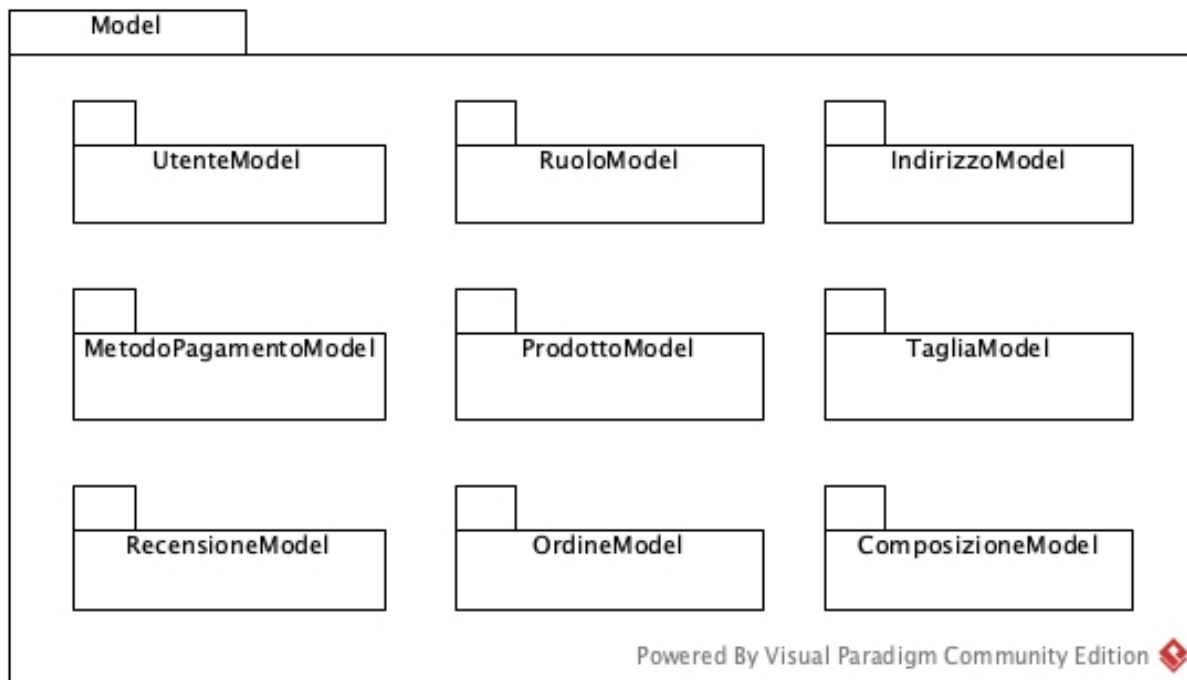


Classe	Descrizione
UtenteBean	Modella un utente registrato
RuoloBean	Modella un ruolo dell'utente
MetodoPagamentoBean	Modella un metodo di pagamento dell'utente
IndirizzoBean	Modella un indirizzo dell'utente
CatalogoBean	Modella il catalogo
ProdottoBean	Modella un prodotto
RecensioneBean	Modella una recensione di un prodotto
TagliaBean	Modella una taglia di un prodotto
CarrelloBean	Modella il carrello
CarrelloItem	Modella un elemento del carrello
OrdineBean	Modella un ordine

ComposizioneBean	Modella la composizione di un ordine
------------------	--------------------------------------

2.2.1.2 Model

Contiene le classi che modellano la conoscenza sulle entità del sistema.

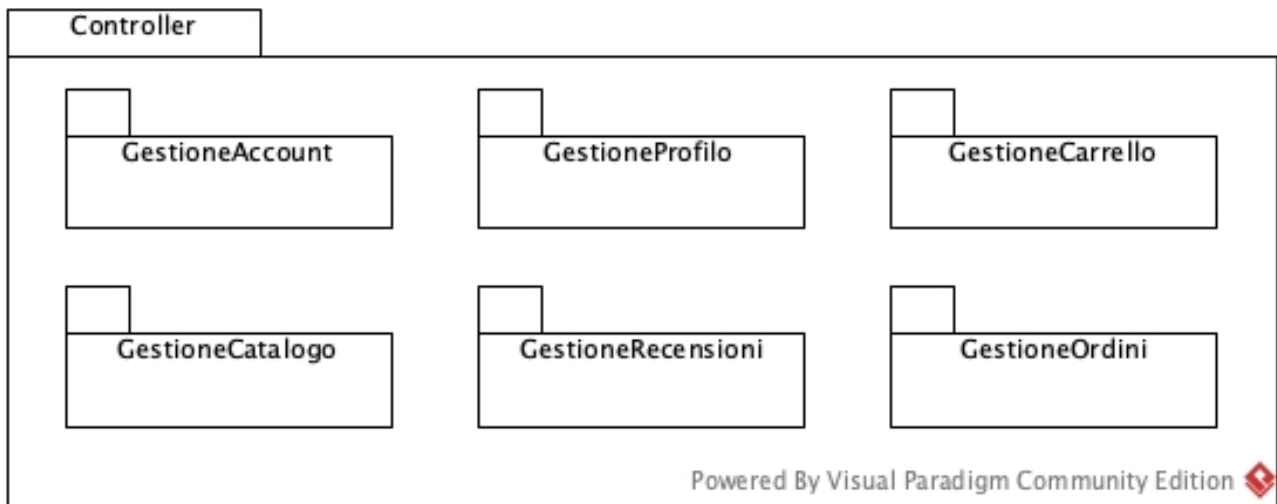


Classe	Descrizione
UtenteModel	Responsabile della conoscenza dei dati sugli utenti registrati
RuoloModel	Responsabile della conoscenza dei dati sui ruoli degli utenti registrati
MetodoPagamentoModel	Responsabile della conoscenza dei dati sui metodi di pagamento degli utenti registrati
IndirizzoModel	Responsabile della conoscenza dei dati sugli indirizzi degli utenti registrati
ProdottoModel	Responsabile della conoscenza dei dati sui prodotti
RecensioneModel	Responsabile della conoscenza dei dati sulle recensioni dei prodotti
TagliaModel	Responsabile della conoscenza dei dati sulle taglie dei prodotti
OrdineBean	Responsabile della conoscenza dei dati sugli ordini
ComposizioneBean	Responsabile della conoscenza dei dati sulle composizioni degli ordini

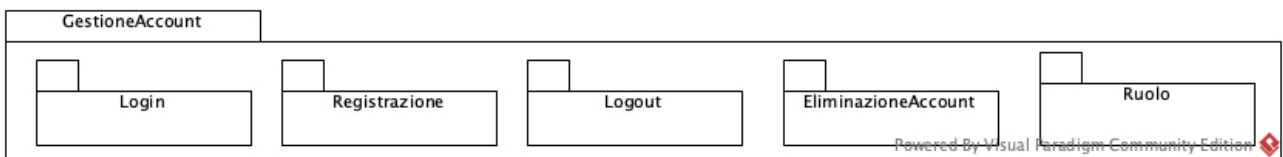
2.2.2 Application Layer

2.2.2.1 Controller

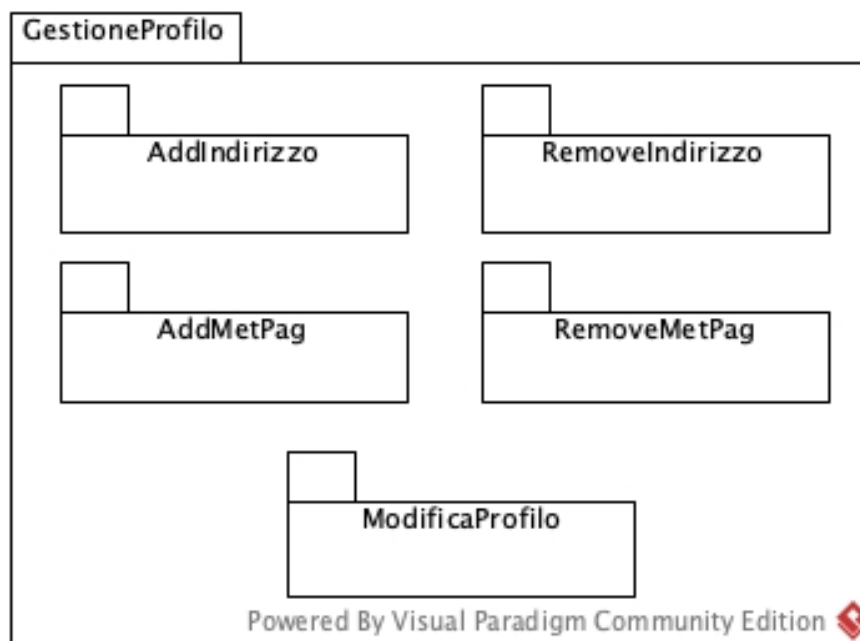
Contiene le servlet che modellano la logica di business del sistema.



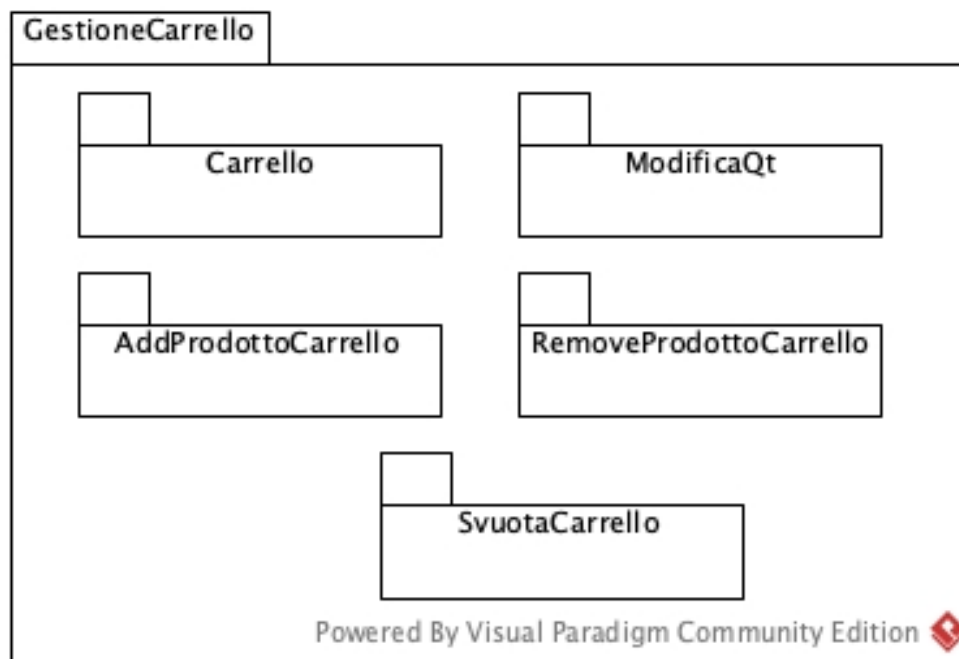
Gestione Account



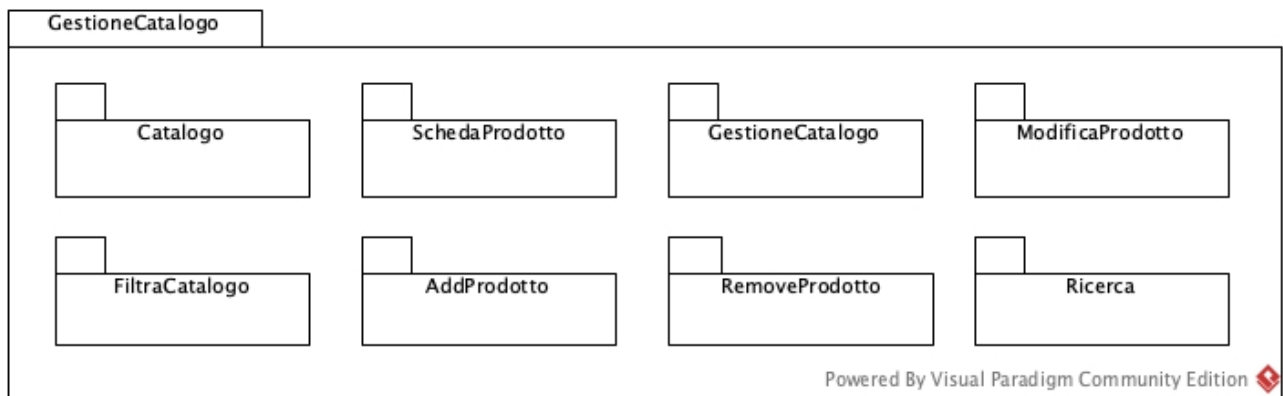
Gestione Profilo



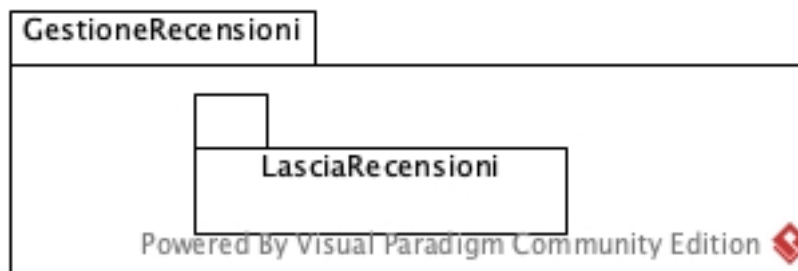
Gestione Carrello



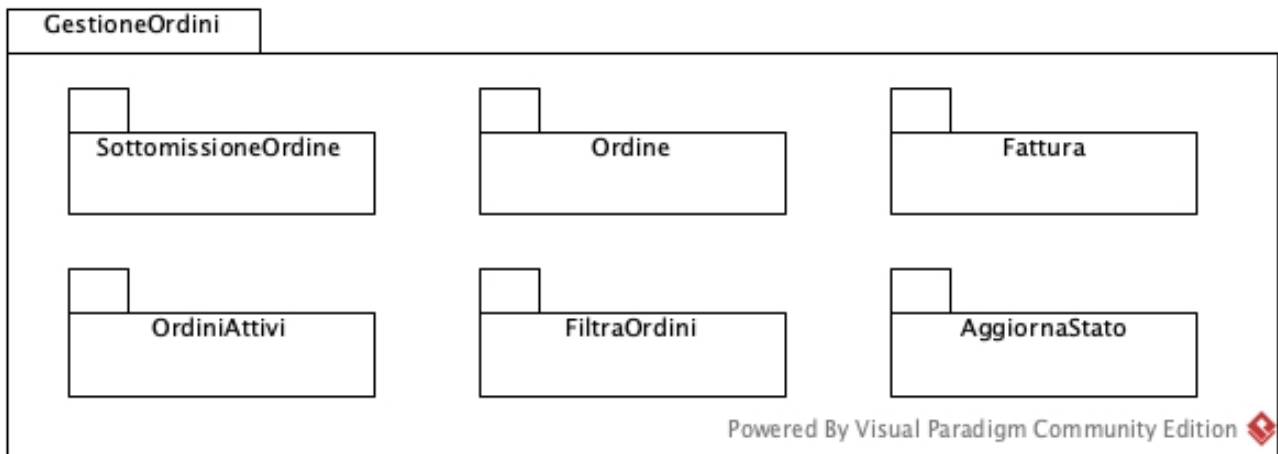
Gestione Catalogo



Gestione Recensioni



Gestione Ordini



2.2.3 Presentation Layer

2.2.3.1 View

Contiene le pagine JSP e HTML che modellano l'interfaccia grafica del sistema.

