iSpesa

Prodotto da:

Grotti Leonardo (mat. 0000692534)
Nanni Lorenzo (mat. 0000695157)
Proscia Michele(mat. 0000695791)

Analisi

Requisiti

L'applicazione si prepone di proporre all'utente una lista per la spesa dinamica e funzionale, con varie opzioni a disposizione dell'utente.

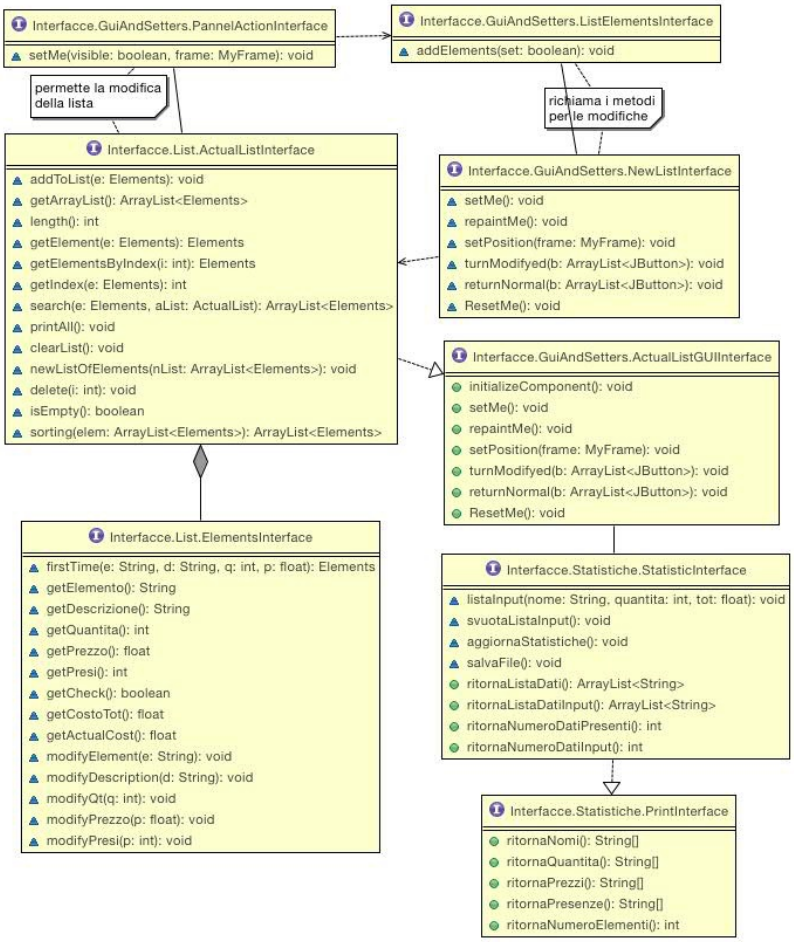
Il programma iSpesa da la possibilità ad un utente di crearsi la propria lista della spesa, dove esso potrà aggiungere elementi, eliminarne, cercarli e modificarli, tutto ciò durante la creazione della lista stessa. Gli elementi inseriti avranno un nome, una descrizione, una quantità che si desidera comprare e il prezzo (all'inserimento dei nomi degli oggetti si avranno dei suggerimenti su cosa inserire, da notare che essi non sono i soli elementi usabili, ma solo consigli). Una volta che la lista sarà terminata si potrà salvare e procedere ad un utilizzo differente della lista, ovvero si potranno, sempre modificare i prezzi dei prodotti acquistati, ma anche aggiungere la quantità di elementi presi. Sempre in questa sezione del programma sarà possibile stampare su carta la lista in precedenza creata, eseguire il checkout (per checkout si intende controllare se tutti gli elementi che si volevano comprare sono stati comprati nella quantità pre indicata e restituire il totale aspettato e il totale effettivo, basato sulla quantità effettivamente presa).
Vi è poi una sezione riguardante le liste dei preferiti calcolate da noi in base alle abitudini dell'utente definite tramite tutte le liste salvate da parte di quest'ultimo.
Infine, vi è un'altra sezione riguardante i suggerimenti, in questa sarà possibile cercare tutti i nostri suggerimenti, inseriti in precedenza su un file e aggiungere degli altri suggerimenti a scelta, sempre dell'utente. Qui, sarà anche possibile eliminarne alcuni (ovvero quelli che l'utente ha inserito, quelli presenti di default non sono modificabili).

Problema

per generare una lista complessa dove i dati vengono mantenuti in memoria i problemi che si devono affrontare sono la memorizzazione dei dati per l'utilizzo al momento dell'esecuzione, con le relative funzioni che permettono di modificare tali dati seguendo certe regole (se un valore deve essere un numero, non può contenere lettere o altri caratteri che non siano numeri), oltre a renderli visibili per l'utente, in modo tale che siano comprensibili da un punto di vista grafico(che non sia una banale lista simile ad una scritta con un editor di testo). Altri problemi che si devono affrontare sono la gestione dei dati alla chiusura del programma, e alla sua riapertura avere la stessa condizione descritta in precedenza e poterla stampare su supporto fisico.
Un ulteriore problema in cui si incorre è gestire i dati per suggerimenti e calcolare le abitudini dell'utente in modo tale da potergli riferire una lista calcolata da noi.
I problemi hanno avuto una difficoltà elevata dal punto di vista della visualizzazione grafica della lista in modo dinamico(problemi di stampa da un punto visivo, non di trascrizione di dati).
Altrettanto difficoltosa è stata la stampa della lista tramite stampante, a causa della compatibilità per ogni sistema operativo. Altra problematica difficoltosa è risultata la lista suggerimenti dal punto grafico dinamico.
Gli altri problemi sono stati di difficoltà inferiore rispetto a quelli appena citati.
Abbiamo un bug nell'annulla quando cerchiamo di stampare il file della lista attuale, premendo tale pulsante otteniamo un nulla pointer exeption

In futuro miglioreremo il pannello dei consigli, rendendo il click su pulsante più reattivo, inoltre il checkout sarà grafico e non da terminale.
Ulteriore miglioria saranno le prestazioni, rendendo il codice meno pesante durante l'esecuzione.
Tutto ciò non è stato possibile a causa di un monte ore non sufficiente a completare queste operazioni

UML del problema proposto

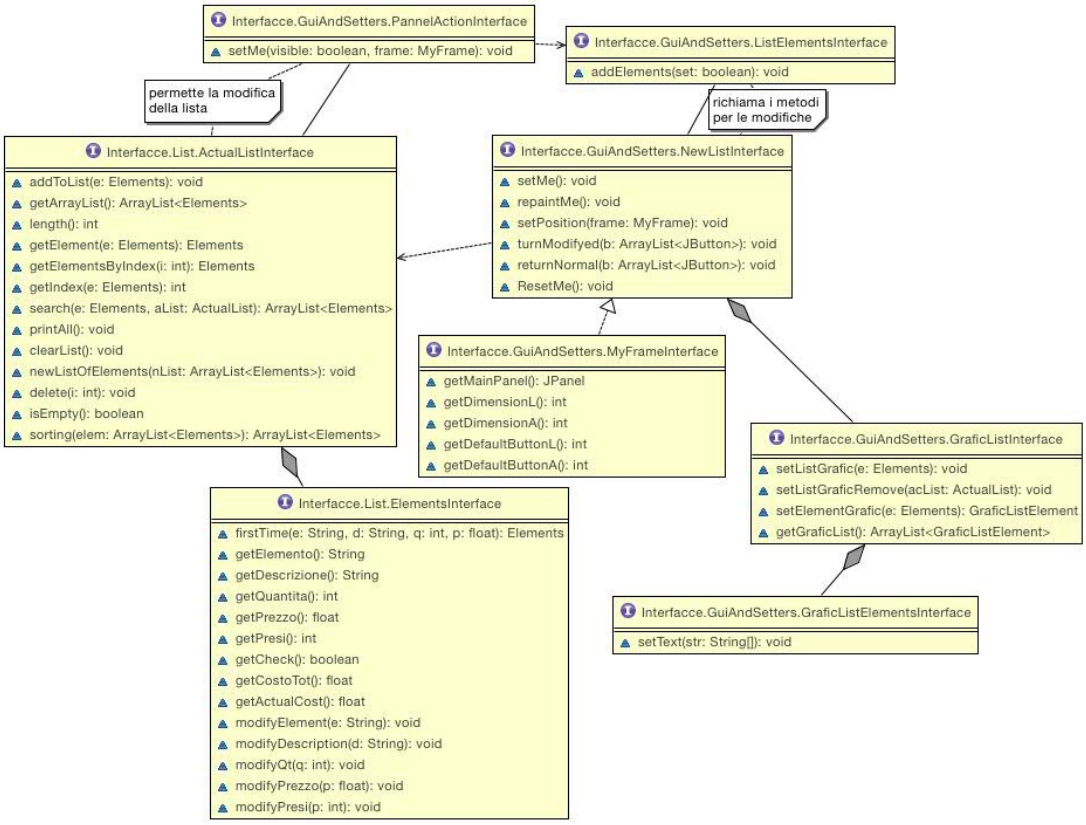


il programma è compatibile sia con Mac, Ubuntu e Windows. Abbiamo testato il programma su computer che hanno una risoluzione da 2560 x 1440 fino ad una risoluzione pari a 1280 x 720, su schermi più piccoli non siamo riusciti a testare la qualità grafica, in quanto non ne possediamo.

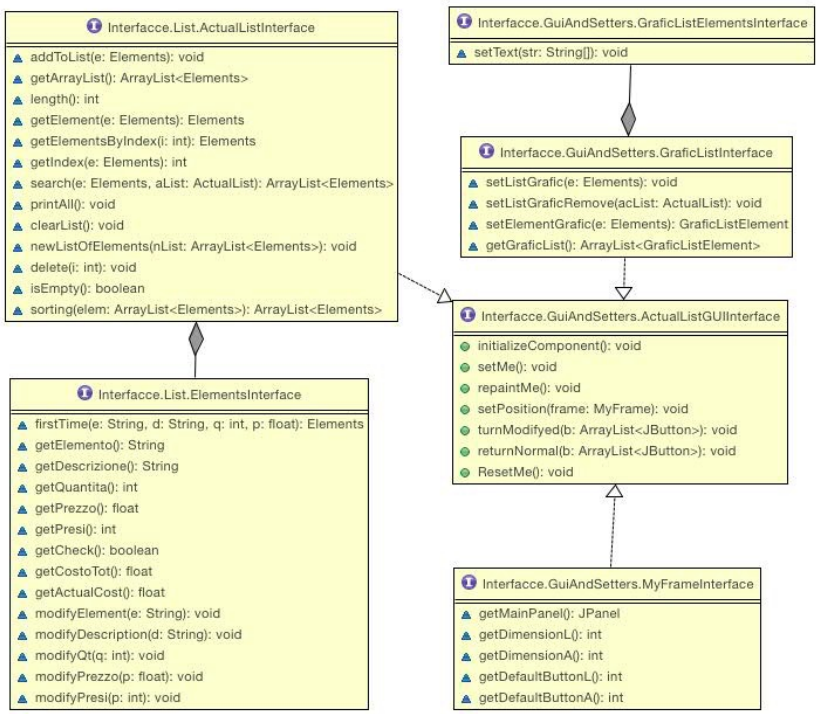
Design

Architettura

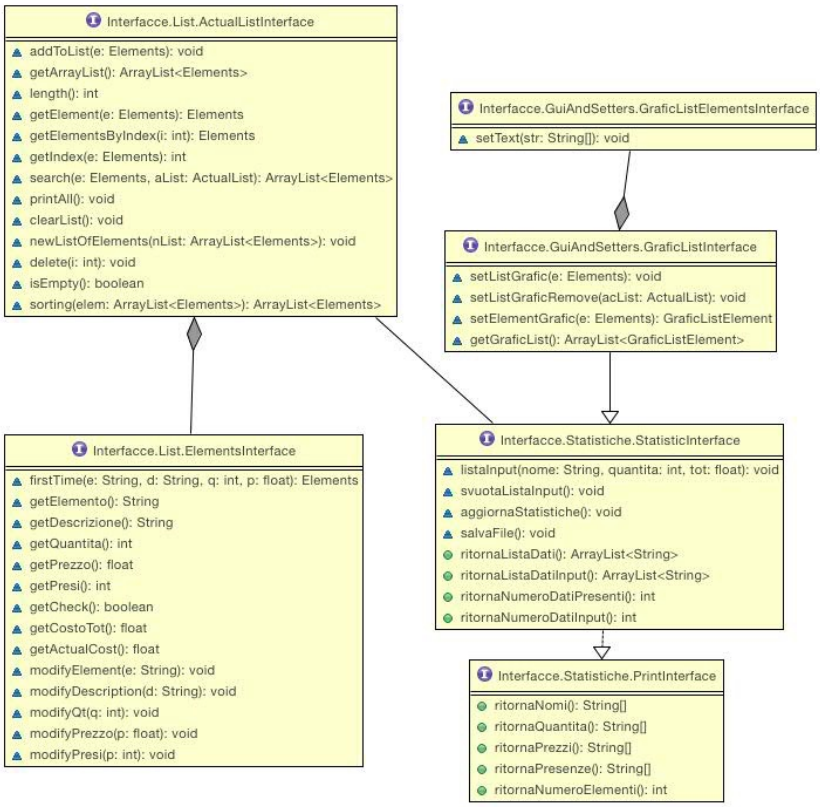
Il seguente schema indica la soluzione ad uno dei tre problemi principali, ovvero come si relaziona con le parti principali del codice la creazione della lista da noi utilizzata:



Il seguente schema indica la soluzione del secondo dei tre problemi principali, ovvero come si relaziona con le parti principali del codice il salvataggio della lista e il suo riutilizzo da noi utilizzata:



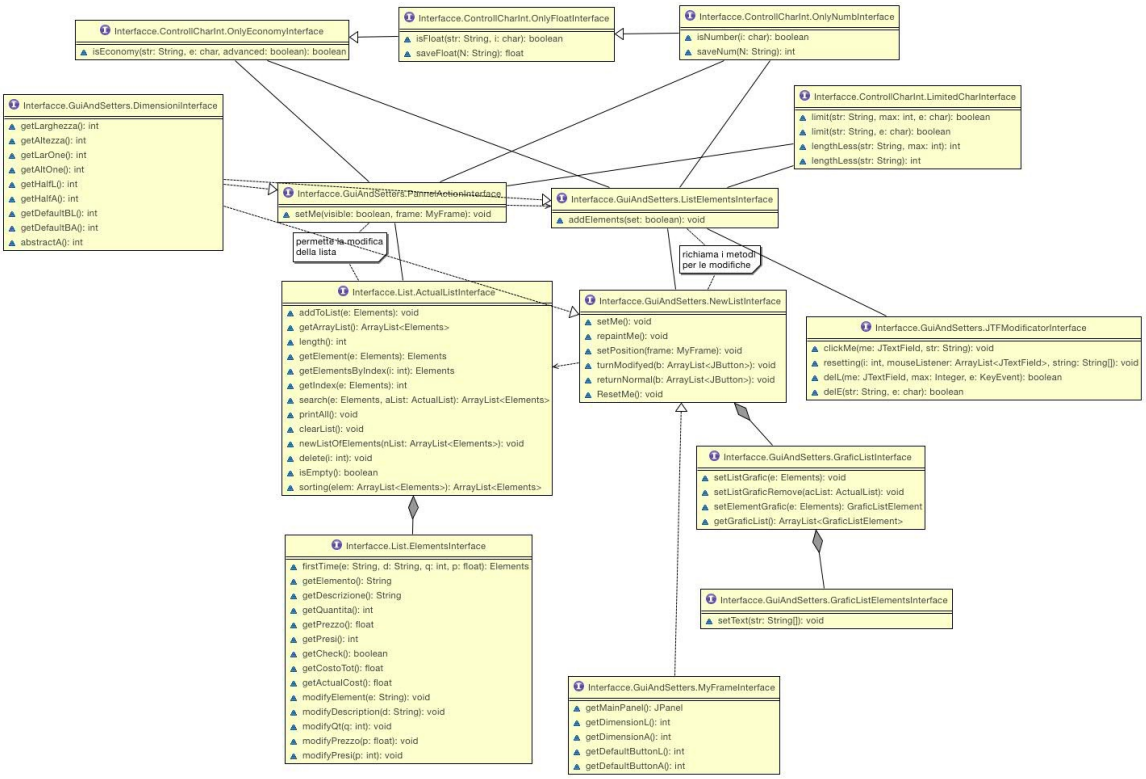
Il seguente UML descrive le relazioni e soluzione del problema riguardante la lista dei preferiti sviluppata da parte nostra:



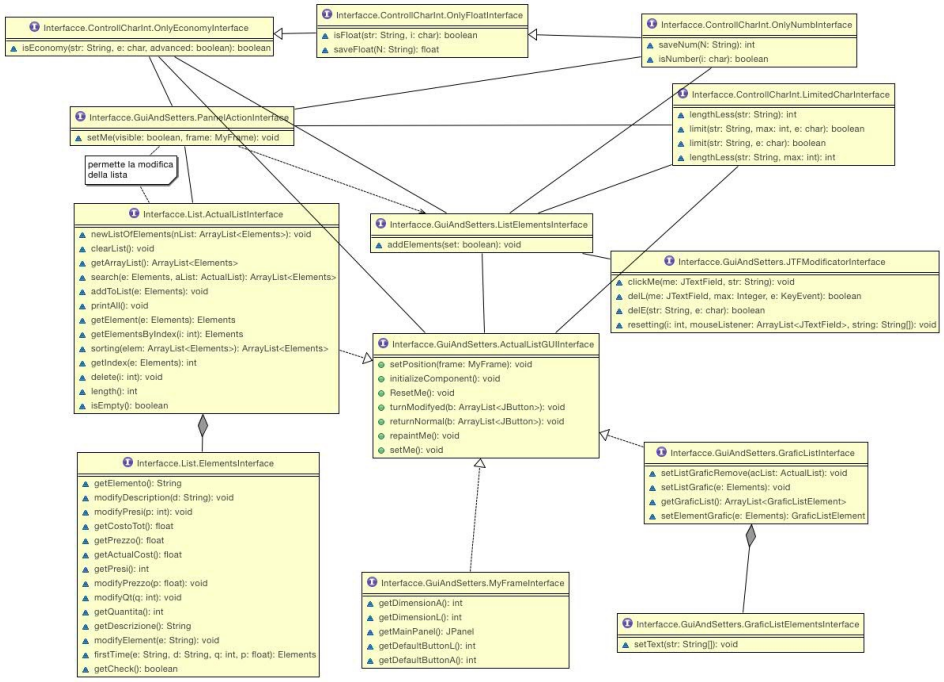
Sostituire la view nella maggioranza dei casi non comporta perdita di funzioni, naturalmente però in assenza di riutilizzo adeguato delle funzioni sarà impossibile far funzionare i diversi metodi (servono punti di input per costruire la lista)
La maggioranza dei metodi possono essere riutilizzati per applicazioni differenti da quella appena sviluppata, ma che comunque ne mantengano l'obbiettivo (creare una lista della spesa).

Design Dettagliato

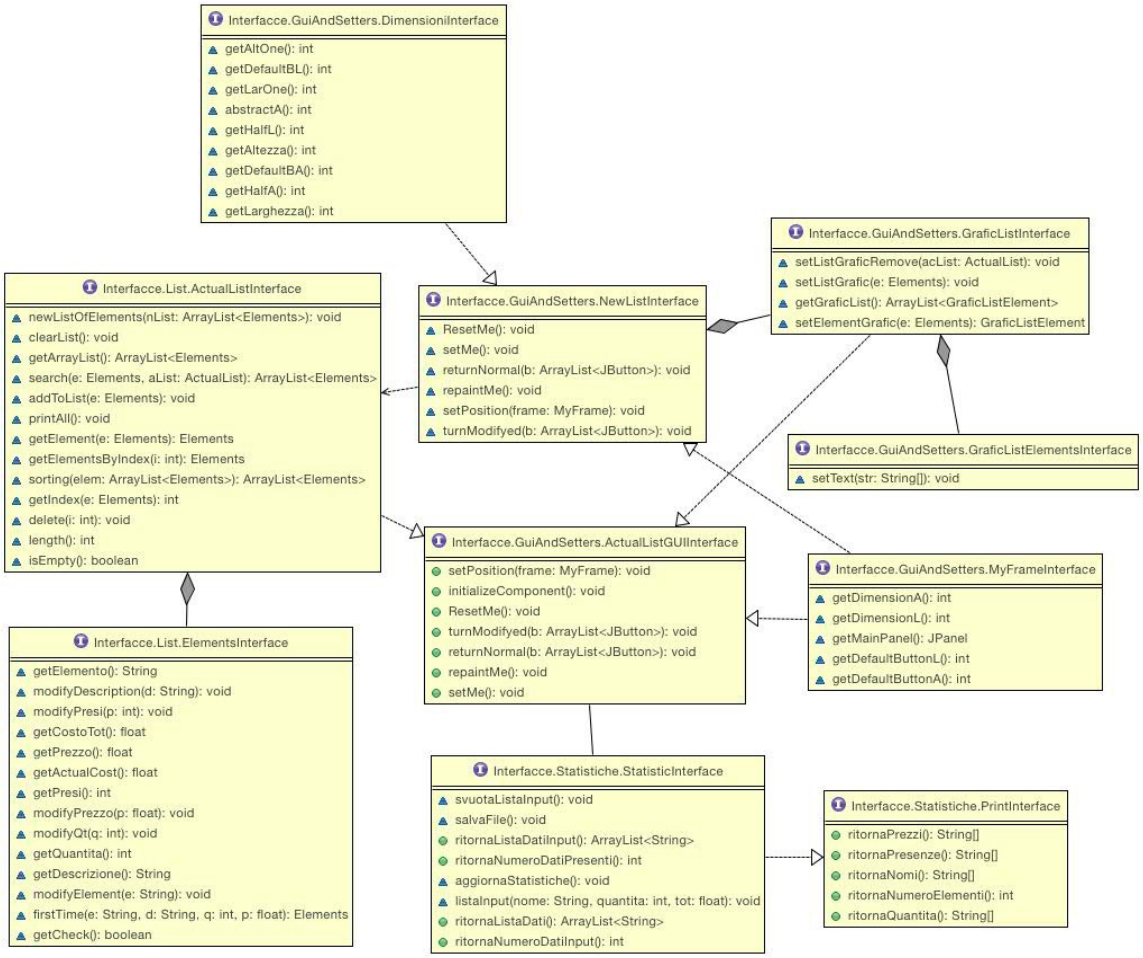
Questo è il design UML delle classi relative al problema della creazione di una lista che ne permettono la visualizzazione e altre funzioni (primo problema descritto nel precedente paragrafo)



Questo è il design UML delle classi relative al problema del salvataggio della lista e il suo riutilizzo che ne permettono la visualizzazione e altre funzioni (secondo problema descritto nel precedente paragrafo)



Questo è il design UML delle classi relative al problema della lista dei preferiti che ne permettono la visualizzazione e altre funzioni (terzo problema descritto nel precedente paragrafo)



Sviluppo

Testing automatizzato

Il Testing è stato effettuato sulle seguenti classi:

TestElements
TestActualList
TestRicerca
TestListElementsPrint

tali classi servono a controllare il cuore del programma, in quanto le altre classi sono una loro implementazione, nell'ordine le seguenti classi eseguono i seguenti test:

- controlla che gli elementi che registriamo si modifichino e che immagazzinino i dati desiderati
- controlla che la lista degli elementi possa essere effettivamente modificata e che la lista non contenga errori
- controlla che la ricerca dei suggerimenti avvenga correttamente
- controlla che la stampa del file sia corretta e controlla che prenda il file corretto

Divisione dei compiti

Leonardo Grotti si è occupato di implementare la lista e le varie funzioni di modifica che si possono svolgere su di essa, come ad esempio aggiunta, eliminazione, ricerca e modifica.
Inoltre si è occupato di dare una impostazione grafica al progetto (creare la lista grafica, che non sia una banale lista scritta in un editor di testo).

Lorenzo Nanni si è occupato di implementare una lista per i suggerimenti che permette di aggiungerne di nuovi e di eliminare quelli inseriti dall'utente ma non quelli presenti di default, tale lista viene utilizzata durante l'inserimento dei nomi degli oggetti da inserire in lista. Inoltre si è occupato di elaborare i dati per le statistiche riguardanti gli elementi inseriti in ogni lista, in modo tale da creare una lista dei preferiti.

Michele Proscia si è occupato del salvataggio della lista su file e il suo futuro riutilizzo, in modo tale che rimanga in memoria anche dopo che il programma stesso viene spento e poi riacceso) si è inoltre occupato di rendere graficamente la parte della lista dei preferiti. Infine ha generato il metodo per la stampa su supporto fisico delle liste create, lista utente e lista preferiti pre-creata da noi, con anche un settaggio grafico.

Le dipendenze dei compiti sono relativamente minime, dato che creata la lista principale (parte abbastanza corta) tutti hanno potuto lavorare singolarmente senza dover aspettare che un altro completasse la sua parte, in quanto conoscendo la struttura della lista tutti potevano sviluppare il proprio codice.

Le parti di ognuno sono state in seguito integrate tra loro in modo semplice, perché le funzioni di ognuno erano relative all'utilizzo di un bottone in cui si inseriva il richiamo alla propria classe.

Le parti di codice che hanno avuto una lieve difficoltà integrativa, quindi è stato necessario lavorare in due o tutti e tre per poterle applicare sono state quella della implementazione di eventi di altre classi rilevati da una classe esterna e la ricerca in lista è stata visionata da tutti per poter controllare eventuali errori. Leonardo Grotti e Michele Proscia hanno elaborato il sistema di ridimensionamento dell'interfaccia per varie risoluzioni di schermo.
Leonardo Grotti ha testato il programma per Mac OS, Lorenzo Nanni ha testo il programma su Windows e Michele Proscia ha eseguito il test su Linux.

La divisione sostanzialmente è stata equa data la complessità delle procedure che ognuno doveva svolgere.

Note di sviluppo

Elementi che hanno interferito con una rapida elaborazione della soluzione al problema proposto sono state la grafica dinamica, la stampa compatibile sui vari sistemi operativi, la risoluzione dei vari schermi e la gestione degli eventi di classi da parte di classi esterne a queste ultime.

La lista dei suggerimenti la avevamo costruita tempo fa per altre motivazioni (un gioco che necessita di vocabolario) di conseguenza abbiamo pensato di riutilizzarla. Tale lista, però era stata costruita tramite un database che avevamo trovato in internet, purtroppo abbiamo smarrito la fonte (in ogni caso l'avevamo riscritta tutta a mano, il copia incolla non ci era stato possibile, eseguita da Lorenzo Nanni).
Il metodo di stampa è stato suggerito da StackOverFlow, rielaborato poi da Michele Proscia per poter funzionare sulla nostra lista, e su vari sistemi operativi.

Abbiamo avuto problemi con BitBucket in quanto uno dei membri (Lorenzo Nanni) non riusciva ad accedere al repository da shell nonostante avesse i permessi necessari, abbiamo provato svariate

volte, assieme a sistemare tale problema, ma senza successo, abbiamo, quindi, ripiegato su altre piattaforme per poter continuare lo sviluppo senza interruzioni.

Commenti Finali

Conclusioni e lavori futuri

I punti di forza del nostro progetto sono gli elementi dinamici che possono portare un utente all'acquisto della nostra applicazione (se fosse monetizzata), altro punto di forza sono la riutilizzabilità della maggior parte delle classi scritte (naturalmente se una classe serve per creare un pannello potrà essere riutilizzata per quella funzione in particolare, con i settaggi da noi implementati).

I punti di debolezza è che certe funzioni sono fini a se stesse, per esempio il metodo che permette la stampa grafica dinamica a video, necessita di una lista che sia relativa alla classe Elements da noi elaborata, questa però tramite una semplice modifica (aumentare la dimensione del pannello, cambiare il numero di JTextField creati - è un ciclo for che li crea - e cambiare il tipo di elementi che si deve passare ad esso - da Elements a un altro qualsiasi oggetto che contenga stringhe -) è possibile effettuarne un riutilizzo.

In futuro si potrebbe pensare ad una estrazione e generalizzazione ulteriore delle classi proposte, oltre a nuove implementazioni (feature) e una versione mobile.

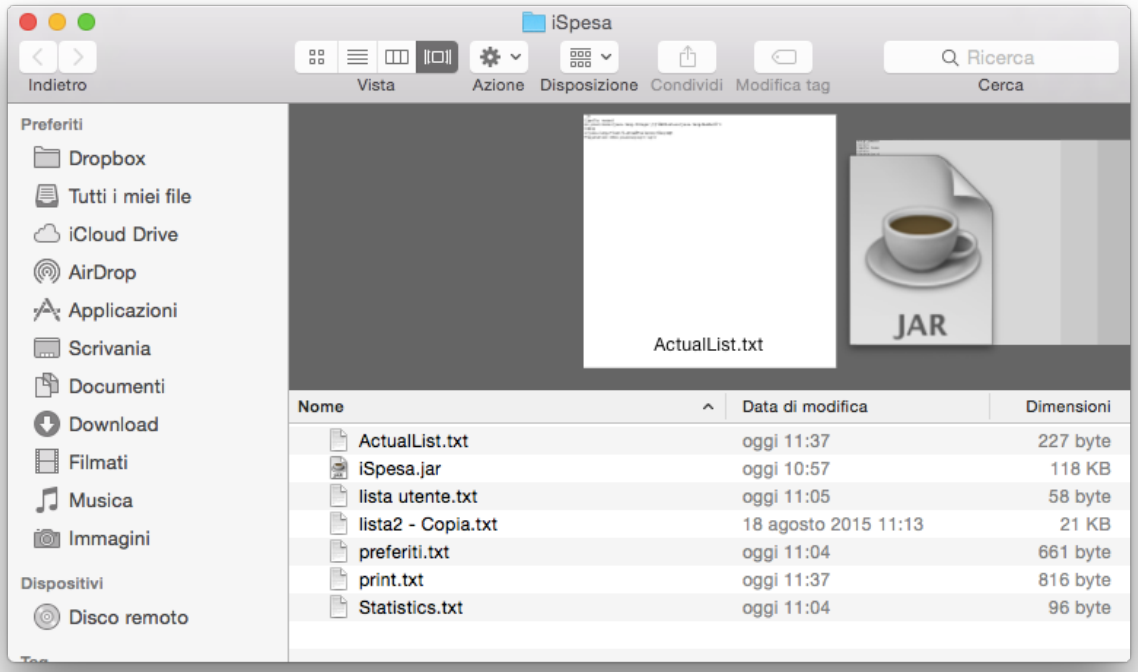
Infine sarebbe interessante far si che installi il nostro progetto in automatico in una cartella con dentro i file necessari e che chiedesse se l'utente vuole creare un alias all'applicazione su desktop, in modo tale da non doverlo far eseguire a quest'ultimo manualmente

Guida Utente

Pre-configurazione

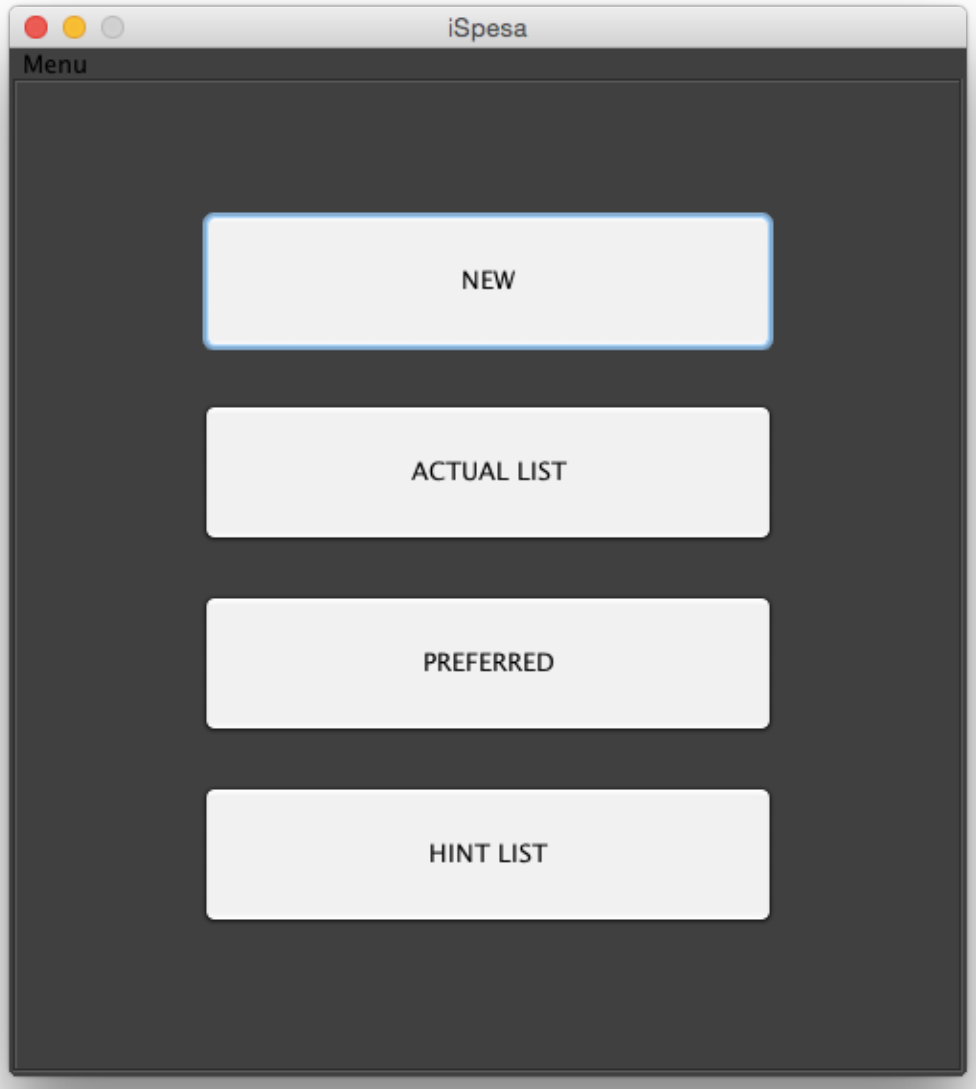
Per poter utilizzare l'applicazione iSpesa sarà necessario avere una cartella in cui è sistemato il file .jar assieme a tutti i file di seguito elencati:

ActualList.txt
lista utente.txt
lista2 - Copia.txt
preferiti.txt
print.txt
Statistics.txt



se si vuole avere l’applicazione all’esterno della cartella è necessario creare un alias

Esecuzione
La seguente è la schermata principale che rappresenta il menù della nostra applicazione



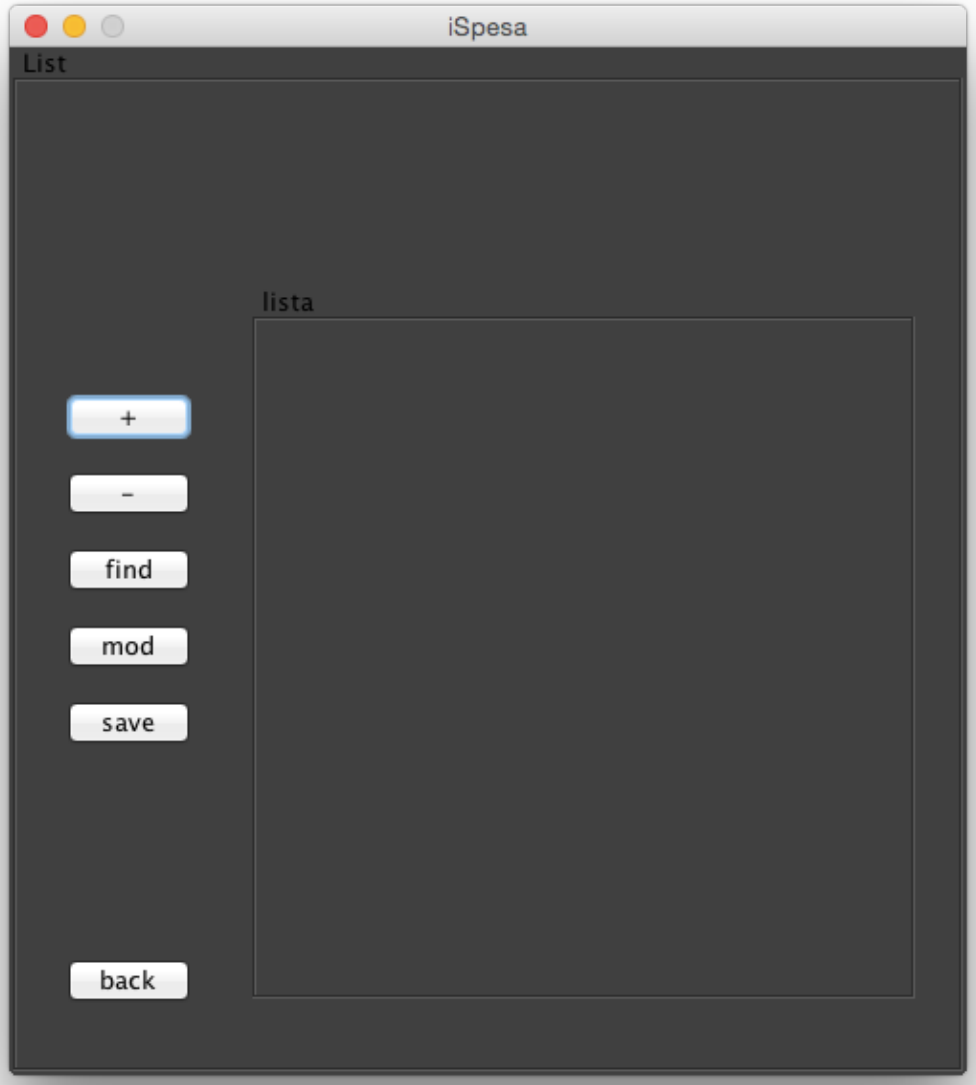
Il primo pulsante (con la scritta NEW) permette di creare una nuova lista, permettendo anche la sovrascrizione della lista precedentemente salvata.

Il secondo pulsante (ACTUAL LIST) permette di fare modifiche lievi alla lista slavata in precedenza e poterla rivisualizzare e stampare.

Il terzo pulsante (PREFERRED) permette di visualizzare una lista di preferiti (più che altro dati statistici) calcolata da noi e poterla stampare

Il quarto pulsante (HINT LIST) permette di controllare quali sono le parole presenti nei suggerimenti, eventualmente se ne possono aggiungere o eliminare

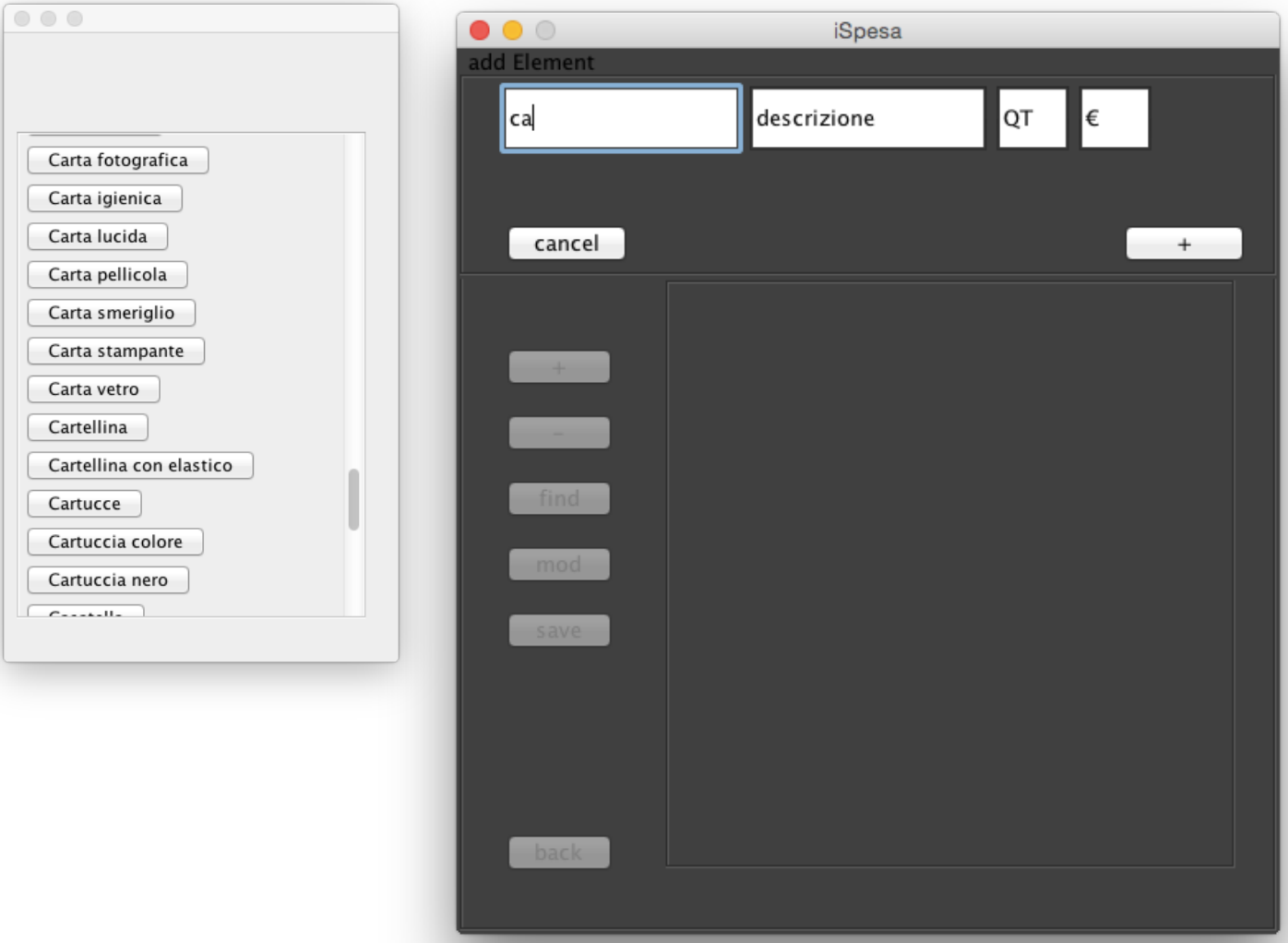
La seguente schermata presenta il contenuto del primo pulsante descritto in precedenza:



i pulsanti sulla sinistra permettono varie operazioni (in ordine dall'alto verso il basso):

- aggiungere un elemento
- rimuovere un elemento presente nella lista
- cercare uno o più elementi in lista
- modificare uno o più elementi della lista
- salvare la lista su file
- permette di tornare indietro al menù principale (N.B. se la lista non è stata salvata in precedenza viene persa irrimediabilmente)

Le seguenti immagini sono riproposte per ogni pulsante (da + a mod) cambiando solamente la funzione che verrà eseguita alla pressione del pulsante azione.

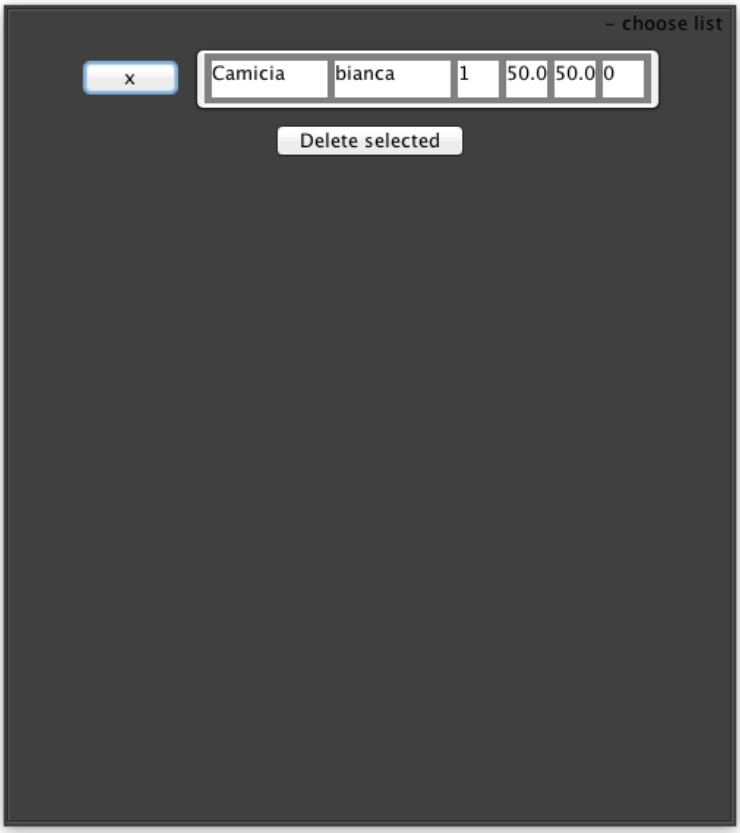


Il pannello di destra permette di inserire i campi dell'oggetto da comprare (in questo caso, per gli altri pulsanti dipende dalla funzione scelta)

I pannelli dove si può scrivere da sinistra verso destra indicano il nome, la descrizione, la quantità che si vuole comprare e il prezzo unitario del prodotto da comprare

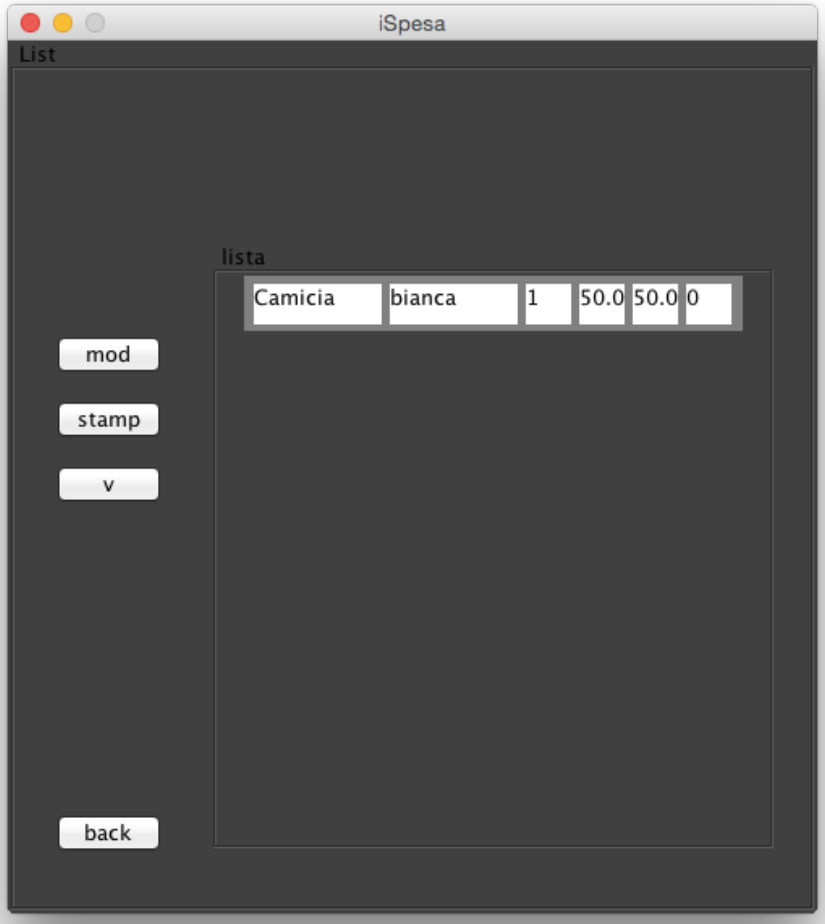
Il pannello di sinistra si aggiorna con ciò che scriviamo nel nome del prodotto, eseguendo una ricerca che ci darà suggerimenti correlati. Alla pressione di uno di tali suggerimenti verrà immesso come nome del prodotto il suggerimento selezionato.

In questa schermata si può vedere come l’oggetto è stato inserito in lista



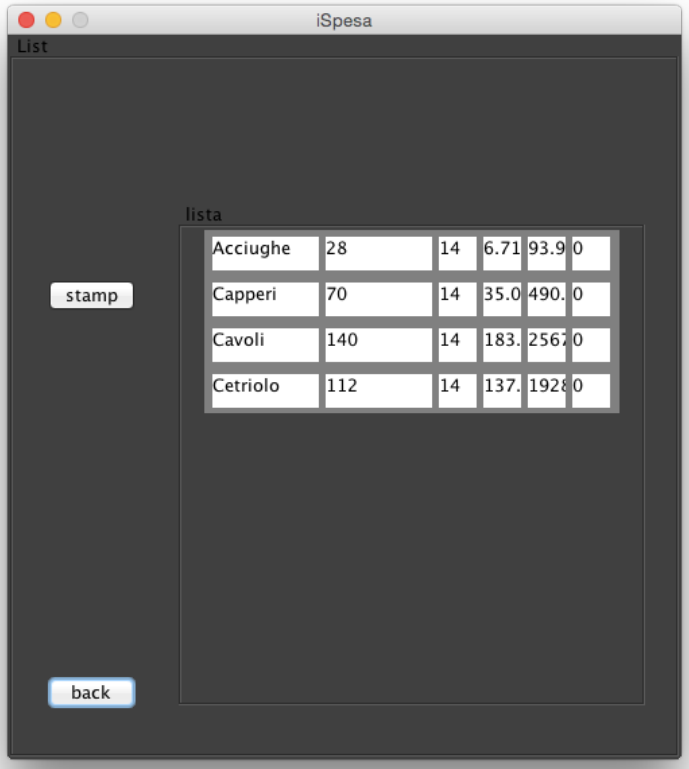
Tale schermata è ciò che si presenta per l'eliminazione, per modifica e cerca sono similari.
La schermata si può chiudere col pulsante x, se vi fossero stati più elementi con lo stesso nome sarebbero comparsi nella lista.
Cliccando sui pulsanti che contengono i dati si esegue la selezione di elementi da eliminare, premendo poi il pulsante Delete selected si procede all'eliminazione vera e propria e si torna alla schermata precedente con l'assenza degli elementi eliminati (nel caso di modifica con gli elementi modificati presenti ma con le nuove specifiche date)

Questa invece è la schermata che si presenta all cliccare sul secondo pulsante del menù.
I pulsanti sulla sinistra permettono in ordine:
La modifica del prezzo e degli elementi acquistati (il funzionamento è similare all'eliminazione mostrato in precedenza)
La stampa permette di stampare la lista, permettendo la selezione della stampante
La v è il check out della lista (per ora avviene solo una stampa da terminale)
Il back come nel caso precedente torna al menù



La seguente schermata rappresenta la Preferred List che indica le statistiche delle liste create col tempo dall'utente.
In ordine è presente il nome, quantità, frequenza nelle liste, prezzo totale nella singola lista, soldi spesi in totale per poterli comprare (somma di tutte le liste) e numero di presi.

Il pulsante stampa permette di stampare tutta la lista
Il pulsante back fa tornare al menù



La seguente lista rappresenta la gestione della lista dei suggerimenti.

In tale lista è possibile aggiungere nuovi elementi che non siano già presenti
ed eliminarne altri presenti nella lista (N.B. possono essere eliminati solo quelli elementi aggiunti dall'untene, quelli di default non possono essere rimossi)

La lista rappresenta prima gli elementi aggiunti dall'utente in ordine alfabetico e poi quelli di default in ordine alfabetico