30/05/2021

I-eats Delivery Tool

Progetto di gruppo per Laboratorio di Algoritmi e Strutture Dati

Gianluca Viscardi
N86003309
Mattia Tarallo
N86003283
Mattia Rossi
N86003211

Sommario

1. INTRODUZIONE GENERALE

1. Descrizione del software

I-eats si propone come tool per la gestione di ordini relativi ad un arcipelago di isole chiamato *Alola*. Il software è diviso principalmente in *3 interfacce*: un'interfaccia *Cliente*, la quale permette di effettuare e cancellare ordini; un'interfaccia *Driver*, dalla quale i driver possono prendere in carico ordini e completarli partendo da una locazione variabile; infine, un'interfaccia di *Debug*, dalla quale si possono effettuare svariate operazioni di testing.

2. Approccio al problema

Tutte le funzionalità di I-eats sono state implementate tramite una suddivisione in *librerie* (raccolte all'interno della cartella *headers*) così da poter gestire interfacce e strutture dati in maniera *modulare*. Inoltre, è stato fatto un *vasto uso di algoritmi ricorsivi*.

Si può facilmente notare come entrambi i processi presentino molte similarità. Difatti, molte delle funzioni di Azalea, dal punto di vista implementativo, sono state sviluppate *tenendo ben presente l'aspetto modulare del programma* e quindi è stato possibile riutilizzare codice generico sia dal lato pazienti sia dal lato laboratorio.

Nello sviluppo di I-eats si è tenuta in considerazione la difficoltà di una tipica interfaccia da terminale e, di conseguenza, si è deciso di approcciare per un sistema di *GUI* sviluppato *ad hoc* il quale consente ai fruitori di interagire con il software *nella maniera più semplice possibile*.

2. ANALISI DEL CODICE SORGENTE

1. Analisi delle librerie

Molte delle funzioni di I-eats relative alle GUI sono state sviluppate tramite l'ausilio della libreria `conio.h` di MINGW.

La libreria `conio.h` permette di utilizzare funzioni come `_kbhit()` e `_getc()` che permettono di prendere valori in input per poi valutarli *senza fermare l'esecuzione del programma* come farebbe una `scanf` o una `gets`.

Oltre alle librerie di MINGW, come già specificato nell'introduzione generale, *ne sono state implementate altre ad hoc*:

A. `isle.h`
   Contiene tutti i prototipi relativi alle funzioni principali del software con qualche funzione di utility strettamente relativa al funzionamento di I-eats.
B. `cliui.h`
   Contiene tutti i prototipi relativi alle funzioni delle stampe della GUI, una struct definita come `Choice_t` (contenente una stringa ed un valore booleano), vari array extern di `Choice_t` che vengono utilizzati come base per riempire i menù delle scelte, le macro relative alla grandezza di tali array e le macro dei

codici ASCII che permettono di cambiare il colore dei valori stampati nel terminale.

C. `datastructures.h`
   Contiene tutti i prototipi relativi alle funzioni generiche delle strutture dati e una struct definita come `Node_t` (contenente un puntatore a void ed un puntatore alla struct stessa). All'interno di questa libreria vengono anche incluse tutte le librerie relative alle singole strutture dati.

D. `graph.h`
   Contiene tutti i prototipi relativi alle funzioni di un *grafo non orientato* assieme alle funzioni che *valutano i percorsi possibili all'interno di esso*. Inoltre, al suo interno sono presenti due struct `Vertex_t`, la quale definisce un *vertice*, e `Edge_t`, la quale definisce un *arco*.

E. `linkedlist.h`
   Contiene tutti i prototipi relativi alle funzioni di una *lista singolarmente linkata*, molte delle funzioni definite lavorano sui file di I-eats oltre che a permettere lo svolgimento di operazioni basilari. Include le librerie `utils.h` e `datastructures.h`.

F. `account.h`
   Contiene tutti i prototipi relativi alle funzioni di *login e registrazione di driver o clienti*.

G. `utils.h`
   Contiene tutte le dichiarazioni dei tipi utilizzati da I-eats (`String`, `IdPair_t`, `User_t`, `Vehicle_t`, `Resource_t`), una *macro* relativa alla grandezza massima di una `String`, una variabile *extern* costante che è la chiave di sicurezza per registrarsi dal lato driver e svariate funzioni di utility che vengono utilizzate in svariate sezioni del codice.

Tutte le librerie sopraelencate sono organizzate nel seguente modo (salvo eccezioni):

Organizzazione librerie

Tutti i nomi delle funzioni (così come quelli di variabili, parametri, tipi e macro) sono stati assegnati in modo da essere chiari ed esplicativi per il programmatore che si appresta ad esaminare il codice. Molte delle funzioni dichiarate, inoltre, fungono da wrapper. Le funzioni wrappate seguono una notazione del tipo `_nomefunzione(…)` mentre le funzioni wrapper e quelle che non necessitano di un wrap non presentano l'underscore all'inizio.

2. Analisi delle strutture dati

Come precedentemente anticipato, *le strutture dati utilizzate sono due*. È opportuno precisare che le funzioni relative a tali strutture dati sono state sviluppate prevalentemente

con un approccio di *tipo ricorsivo*. Inoltre, è stato fatto anche un ampio utilizzo dei *doppi puntatori*. Si procede dunque ad analizzare tali strutture dati dal punto di vista implementativo specificandone anche le varie responsabilità:

A. Single Linked List

Le liste singolarmente linkate funzionano sulla base di un tipo definito come `Node_t` il quale presenta due campi: `void *Data` e `struct node *next`. Tali campi si riferiscono rispettivamente ai dati del nodo ed al nodo successivo in lista. Si accede alla struttura tramite la testa e si scorre la lista utilizzando i puntatori a next dichiarati in ogni nodo. La definizione dei dati come puntatori a void *permette inoltre la generalizzazione della struttura* così da poterla usare con svariati tipi di informazioni.

B. Non-Oriented Graph

Tale struttura è costruita con un approccio *non orientato* ed è basata su due struct: `Vertex_t` ed `Edge_t`. La struttura `Vertex_t` contiene un `unsigned int` che corrisponde all'ID del vertice, una `String` corrispondente al nome del vertice (che in questo caso sta a rappresentare il nome di un'isola), due `bool` che determinano se il nodo è stato visitato/esplorato (utilizzato nella navigazione del grafo) e una lista di ID di vertici adiacenti definita come puntatore a `Node_t`. La struttura `Edge_t` invece contiene a sua volta una seconda struttura `IdPair_t` (contenente 2 ID di vertici) la quale identifica i due vertici collegati ed un double relativo al peso massimo che l'arco può sostenere. Il grafo è basato sulle *liste di adiacenza*, di conseguenza ogni vertice presente in esso conterrà a sua volta tutti gli ID relativi ai vertici con cui hanno un collegamento.

3. Analisi delle funzioni principali del software

Ogni header (salvo poche eccezioni) possiede il suo file c corrispondente, tali file contengono le inizializzazioni delle variabili costanti dichiarate nei rispettivi header e anche tutte le realizzazioni dei prototipi precedentemente discussi. Tutte le operazioni che richiedono modifiche più elaborate all'interno di un file si avvalgono, quasi in tutti i casi, delle strutture già esaminate. Si procede dunque ad analizzare le funzionalità di particolare rilievo nel software:

A. Terminal GUI

L'interazione con la GUI viene effettuata dalla funzione `getVerticalInput` la quale prende in input un array di `Choice_t`, un intero che corrisponde al numero totale di opzioni ed un puntatore void a funzione, essa restituisce *un intero che va da 0 ad n* in base a ciò che viene scelto a schermo. *Tale funzione è resa generica grazie all'utilizzo del puntatore a funzione*, difatti è grazie a quello che *è possibile stampare a schermo diversi tipi di interfacce senza dover cambiare il codice dell'input*. Di default il cursore si trova di fianco al primo valore `Choice_t` che ha il suo valore boolean impostato a true, ogni volta che viene premuta una delle due frecce direzionali (quelle che puntano rispettivamente in alto e in basso) *il frame corrente viene aggiornato* ponendo il valore boolean della precedente opzione a false e posizionando il cursore di fianco all'opzione che adesso risulta true. Assieme a questi aggiornamenti viene

sempre verificato che l'intero corrente si trovi *sempre all'interno del dominio delle opzioni disponibili*. Alla pressione di invio viene restituito il valore intero selezionato.

B. Menù principale

All'interno del menù possono essere effettuare tre operazioni: *accesso, registrazione e debug*. Nelle due prime opzioni sarà data la possibilità di scegliere il tipo di utente (cliente o driver). Per quanto riguarda l'accesso verrà richiesto all'utente di inserire un CF ed una password, verrà poi verificato che tali dati siano *corretti e presenti nel file* degli utenti standard o nel file degli utenti driver a seconda dell'opzione scelta in precedenza. Se l'accesso viene eseguito con successo allora si verrà reindirizzati al menù relativo al tipo di utente scelto altrimenti si verrà riportati al menù precedente. Per quanto riguarda la registrazione, il processo è analogo salvo per *due differenze principali*: la prima consiste in una verifica addizionale della password così da essere sicuri che sia corretta mentre la seconda si riscontra solo dal lato driver e consiste in uno step addizionale nel quale viene richiesta una *chiave di sicurezza*, tale controllo viene effettuato così da poter dare accesso al menù driver solo a chi possiede quest'informazione. Il menù di debug, invece, è un menù senza bisogno di login che permette di effettuare varie operazioni utili per il testing del software.

C. Cliente

Nel menù del cliente, l'utente ha la possibilità di effettuare due semplici operazioni: *ordinare delle risorse oppure cancellare un'ordinazione* fatta eventualmente in precedenza. Nel caso in cui l'utente effettui un ordine verrà fatto un controllo nel file `IsleOrders.isle` al fine di verificare se è ancora in corso un altro ordine a carico dell'utente corrente. Se così non fosse l'utente potrà *visualizzare tutte le risorse esistenti* e selezionare quelle di cui ha bisogno, *ogni risorsa ha un limite di 100 unità per ordine*.

Nel caso in cui l'utente cerchi di effettuare una cancellazione verrà controllato prima se esiste un ordine a carico dell'utente che richiede la cancellazione, *se l'esito è positivo allora l'ordine viene immediatamente cancellato*.

D. Driver

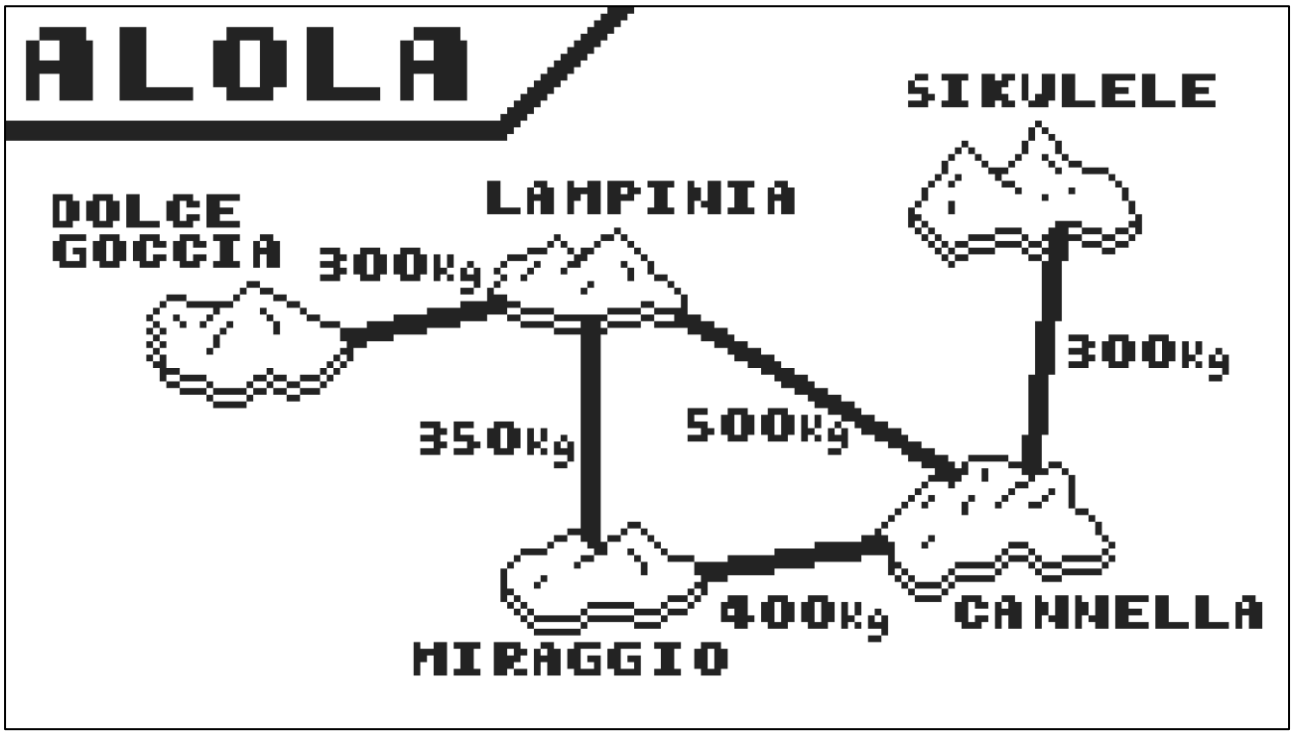
Il driver, nel suo menù personale, può effettuare *3 operazioni principali* ovvero *prendere in carico un ordine*, *cambiare il suo veicolo* con un altro veicolo aziendale oppure *visualizzare la mappa originale di Alola*.

Quando il driver prova a prendere in carico un ordine il peso degli elementi dell'ordine verrà *addizionato al peso della vettura associata al driver* e il *vertice di partenza viene deciso casualmente* (in modo da *simulare la possibile posizione corrente del driver*). A questo punto, la funzione `tryRoute(…)` cercherà *il percorso minimo* passando per i *ponti* definiti in `ExistingBridges.isle` (i quali hanno attribuito un *peso massimo* che possono sostenere) e segnalerà a schermo tutti i *vertici attraversati*. Se il vertice di destinazione è presente nel percorso trovato allora il driver *completerà l'ordine e consegnerà le merci con successo* altrimenti apparirà un messaggio a schermo che annuncia il fallimento dell'ordine.

Nel caso in cui l'utente selezioni l'opzione di cambio vettura si verrà reindirizzati ad un menù con i 3 veicoli generici dell'azienda: *Rotom-Apecar*, *Rotom-Furgone* e *Rotom-Autocarro*. Selezionandone uno automaticamente il veicolo dell'utente verrà *scambiato con il nuovo*

veicolo scelto e tale scelta verrà salvata in un file chiamato `VehiclesCFs.isle`. I tre veicoli pesano rispettivamente 100 kg, 200 kg e 300 kg.

Nel caso in cui si scelga di visualizzare la mappa delle isole verrà stampata a schermo una *copia della mappa originale dell'arcipelago in ASCII Art* con su scritti i *nomi delle isole* e il *carico massimo di ogni ponte*.



Mappa originale di Alola in Pixel Art

3. GUIDA ALL'INSTALLAZIONE

1. Contenuto del file ZIP

I-eats viene distribuito all'interno di un file ZIP. Tale file presenta, oltre che la cartella dell'eseguibile, una cartella src *contenente il codice sorgente del software* assieme al progetto di DevC++. Nell'archivio saranno anche presenti altre cartelle relative al funzionamento del software.

La cartella *"[SRC] I-eats Delivery Tool"* contiene i file C, i file .h (nella cartella *"headers"*) ed una cartella *"data"* contenente i vari file nel caso in cui si decida di compilare il progetto.

La cartella *"[BIN] I-eats Delivery Tool"* invece contiene l'eseguibile del software assieme alla sua icona e ad una cartella "data" che utilizza per il suo funzionamento.

2. Processo di installazione

Qui di seguito verranno esaminati i pochi e semplici step su come installare I-Eats sul proprio PC:

- Aprire il file ZIP "*Gruppo9*";

- Trascinare la cartella *"[BIN] I-eats Delivery Tool"* in una posizione esterna all'archivio;
- Aprire la cartella *"[BIN] I-eats Delivery Tool"* appena estratta;
- Avviare l'eseguibile *"I-eats Delivery Tool.exe"*.

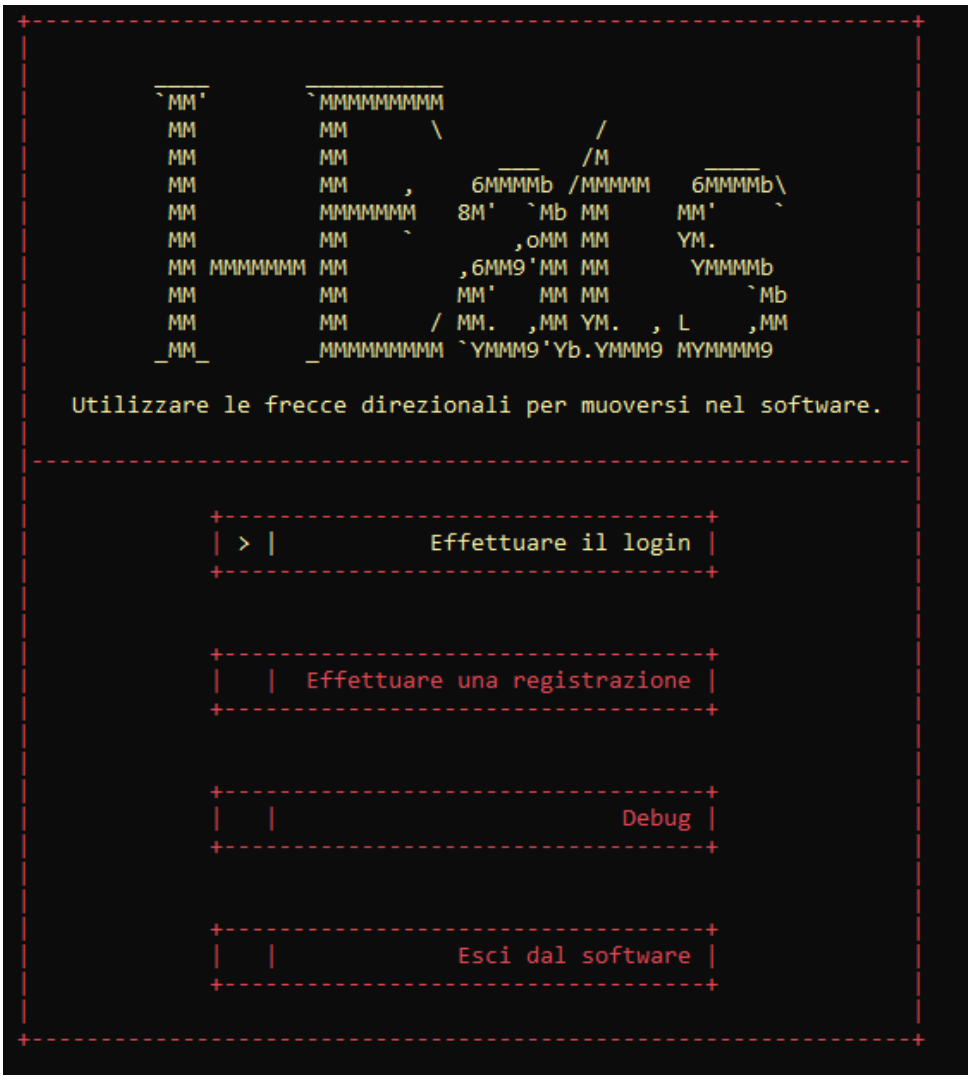
4. GUIDA ALL'UTENTE

1. Introduzione alla sezione

Al fine di rendere l'utilizzo di I-eats *semplice per ogni tipo di utente* verranno esaminate svariate funzionalità del software. Si partirà dunque dall'analisi della schermata principale del software procedendo ad effettuare l'accesso in un account di tipo cliente così da effettuare un ordine. A questo punto, si passerà ad un account di tipo driver che proverà ad eseguire tale richiesta.

2. Schermata principale

La schermata principale offre 3 opzioni oltre all'opzione che permette l'uscita dal software: *login, registrazione e debug*.



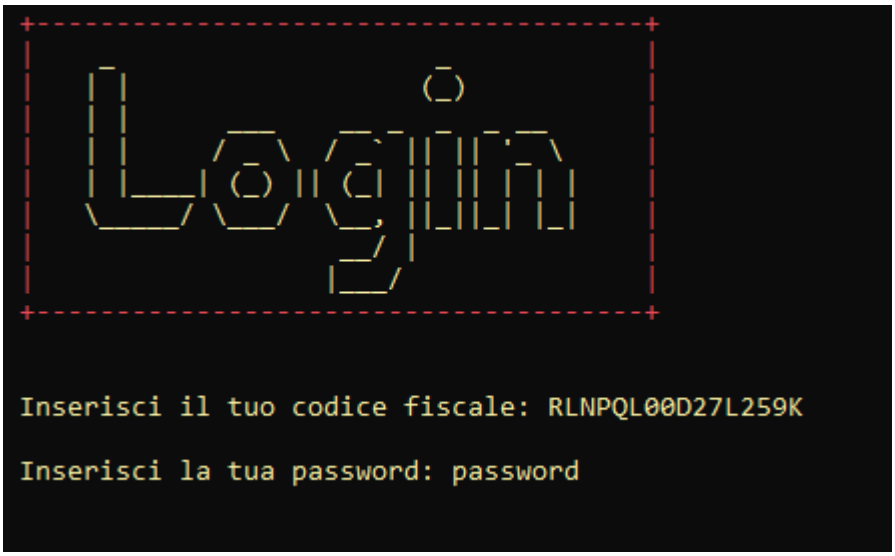
Tramite la schermata di registrazione abbiamo la possibilità di registrare il nostro CF per il lato cliente e/o per il lato driver, nel caso del secondo *sarà anche necessario inserire una chiave di sicurezza* per essere abilitati alla registrazione.

La schermata di login funziona in modo analogo per entrambe le tipologie di utente, è possibile accedere ad uno dei due tipi semplicemente selezionando l'opzione ad esso relativa.

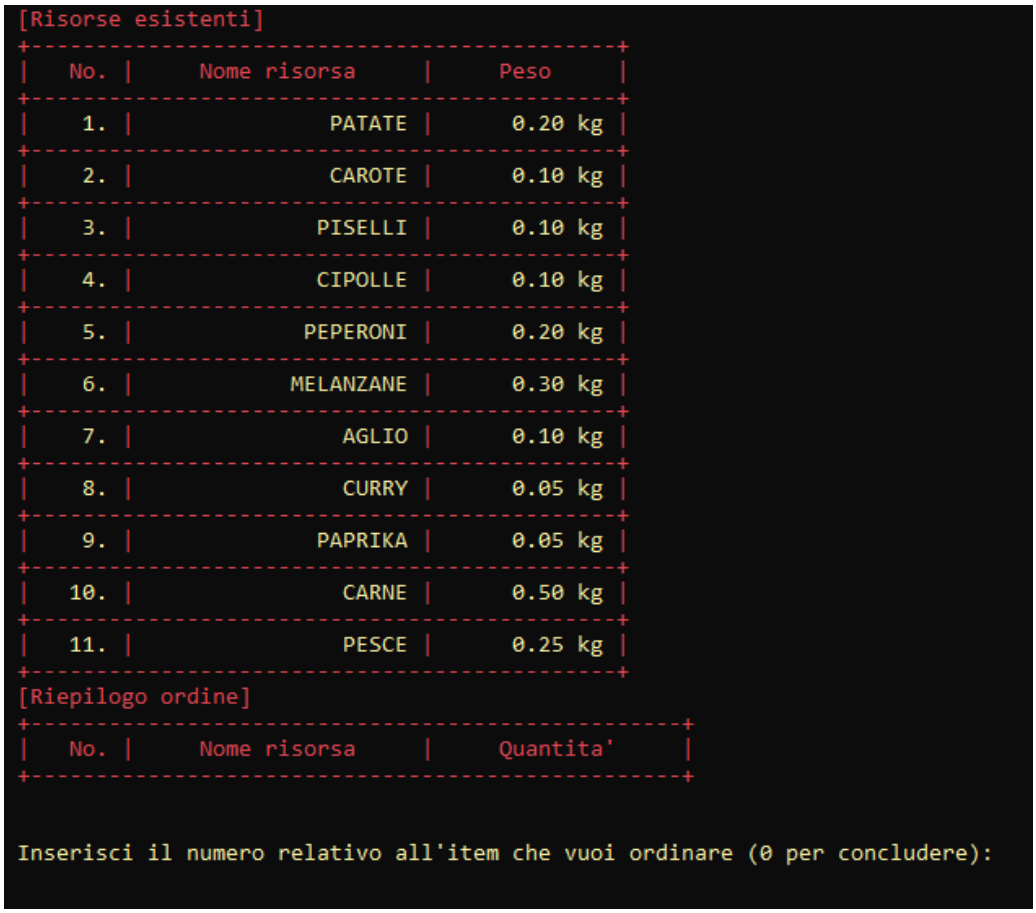
La schermata di debug è pensata per utente esperti, da essa possono essere effettuate svariate operazioni di tipo avanzato sul grafo e sulle risorse presenti nel sistema.

3. Lato cliente

Si presuppone dunque di procedere per un accesso di tipo cliente utilizzando delle credenziali di testing (*CF: RLNPQL00D27L259K; Password: password*).

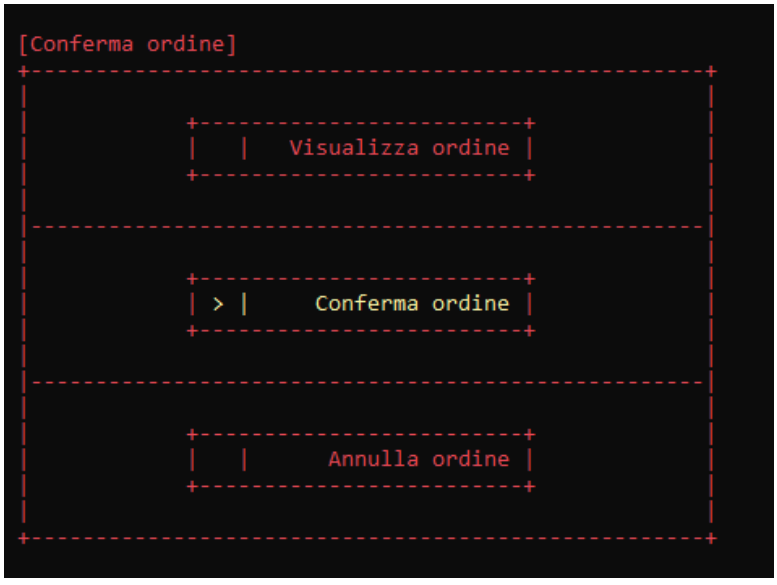


Alla pressione di invio avremo l'accesso al pannello cliente attraverso il quale effettueremo un ordine. Basterà infatti selezionare la prima voce disponibile e, *se non esistono già degli ordini relativi al CF con cui si è effettuato l'accesso*, verrà avviata la schermata dell'ordinazione.



A questo punto verrà effettuato un ordine seguendo le indicazioni a schermo selezionando tramite scelta numerica gli alimenti presenti nell'elenco ed il numero di unità per un determinato alimento. Al termine della scelta verrà richiesto dove consegnare l'ordine e infine apparirà un messaggio di conferma che permette di visualizzare nuovamente l'ordine, annullarlo o confermarlo.

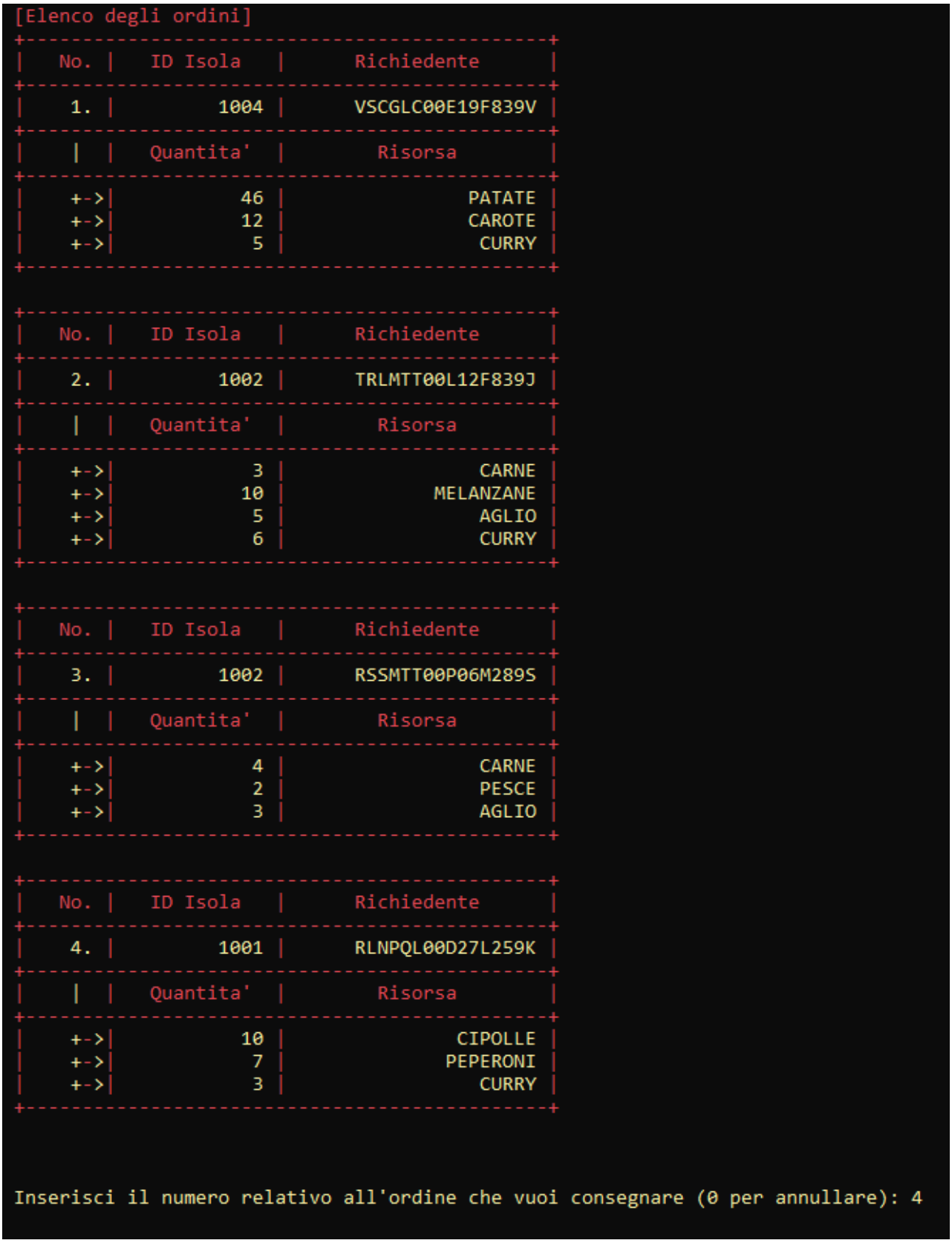
L'ordine viene dunque *salvato all'interno di un file* ed è *pronto ad essere preso in carico successivamente da un driver*.



4. Driver

Anche in questo caso procediamo utilizzando delle credenziali di *testing* (*CF: VSCGLC00E19F839V; Password: password*). Successivamente al login l'utente sarà reindirizzato al pannello driver.

Ai fini dell'esempio, l'utente accede al pannello di selezione ordine e procede a scegliere un ordine da completare.


[Elenco degli ordini]
No. | ID Isola | Richiedente
1. | 1004 | VSCGLC00E19F839V
| | Quantita' | Risorsa
+->| 46 | PATATE
+->| 12 | CAROTE
+->| 5 | CURRY
No. | ID Isola | Richiedente
2. | 1002 | TRLMTT00L12F839J
| | Quantita' | Risorsa
+->| 3 | CARNE
+->| 10 | MELANZANE
+->| 5 | AGLIO
+->| 6 | CURRY
No. | ID Isola | Richiedente
3. | 1002 | RSSMTT00P06M289S
| | Quantita' | Risorsa
+->| 4 | CARNE
+->| 2 | PESCE
+->| 3 | AGLIO
No. | ID Isola | Richiedente
4. | 1001 | RLNPQL00D27L259K
| | Quantita' | Risorsa
+->| 10 | CIPOLLE
+->| 7 | PEPERONI
+->| 3 | CURRY
Inserisci il numero relativo all'ordine che vuoi consegnare (0 per annullare): 4

L'utente quindi seleziona dall'elenco degli ordini quello *fatto da noi in precedenza* (l'ultimo della lista) così da *provare* a portarlo a termine. Tale ordine va consegnato al vertice con ID 1001 ovvero l'isola *Lampinia*. Prima di passare al completamento verrà richiesta una conferma dall'utente che, ai fini dell'esempio, accetteremo.

A questo punto apparirà a schermo il peso complessivo del mezzo utilizzato dall'utente considerando anche le nuove risorse acquisite e, partendo da un *vertice casuale* (in questo caso *Dolcegoccia*, ID 1000), si prova a raggiungere *Lampinia*. Infine, verrà stampato a schermo *il percorso effettuato* e *l'esito del viaggio* che in questo caso *risulta positivo*.