



UNIVERSITÀ DEGLI STUDI
DI NAPOLI FEDERICO II

I-eats

PROGETTAZIONE E SVILUPPO DI UN APPLICATIVO PER LA
CONSEGNA DI ALIMENTARI IN SISTEMA DI ISOLE

a.a. 2020 – 2021

Gruppo 3

Marotta	Vincenzo	N86003005
Muto	Emanuele Ciro	N86003440
Salernitano	Mattia	N86003385

I-eats

Sommario

Descrizione del progetto.....	3
1.1 Analisi del problema	3
1.2 Funzionalità di sistema.....	3
1.3 Mappa delle Isole	4
Strutture e Informazioni.....	5
2.1 Introduzione	5
2.2 Grafo	5
2.3 Driver	8
2.4 Alimenti	10
2.5 File e salvataggio dati	12
Guida d'uso	13
3.1 Comandi e Simulazione	13
3.2 Menu Principale	14
3.3 Menu del Driver	14
3.3.1 Programma Consegna	15
3.3.2 Storico consegna.....	17
3.3.3 Informazioni sull'account.....	18
3.3.4 Cancella account.....	18
3.4 Registrazione	19
3.5 Mappa.....	19

Capitolo 1

Descrizione del progetto

1.1 ANALISI DEL PROBLEMA

Si progetterà e implementerà un applicativo per l'azienda *I-eats*, che si occupa di trasporto e consegna di alimentari in un sistema di isole, tramite dei camion, che sono caratterizzati da un peso e sono guidati da driver.

Le isole sono tutte collegate tramite dei ponti che hanno un carico massimo che possono reggere. È possibile attraversare i ponti in entrambe le direzioni.

Ciascun driver può registrarsi alla piattaforma *I-eats*, specificando obbligatoriamente il peso del camion che utilizzerà per le consegne. Se già registrato, invece, potrà effettuare l'accesso tramite credenziali.

Una volta completato il login, il driver può effettuare una consegna selezionando la merce che intende trasportare, isola di partenza e isola di arrivo.

L'applicativo sarà in grado di fornire il percorso più breve che il driver, con il suo camion, è in grado di fare.

1.2 FUNZIONALITÀ DI SISTEMA

L'applicativo *I-eats* si occupa della gestione dei trasporti e consegna di alimentari in un sistema composto da dodici isole, collegate tra loro attraverso ponti caratterizzati da una portata massima (peso massimo supportabile).

Il trasporto di alimentari è effettuato tramite driver. I driver devono registrarsi sulla piattaforma, specificando il peso del loro mezzo di trasporto.

Una volta effettuato il login tramite username e password, il driver potrà programmare una consegna, cancellare il suo account, vedere lo storico delle consegne da lui effettuate e avere un riepilogo del suo account.

Programmando una consegna, il driver deve scegliere isola di partenza, isola di arrivo e i prodotti che desidera portare.

L'applicativo è in grado di trovare il miglior percorso viabile dal driver, proponendo l'attraversamento di ponti in grado di reggere il peso del mezzo di trasporto sommato al peso dei vari prodotti

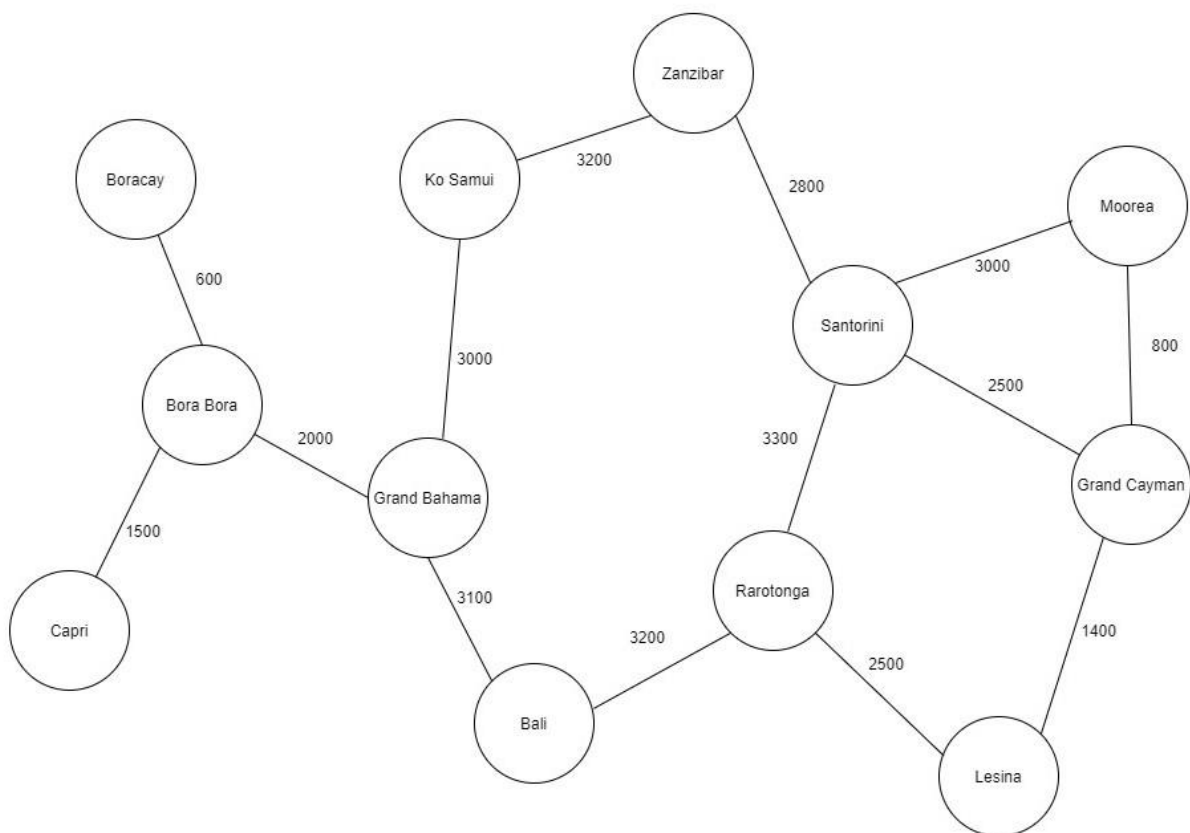
1.3 MAPPA DELLE ISOLE

Il sistema è composto da dodici isole collegate attraverso dei ponti.

Nella mappa, vicino ad ogni ponte, è presente un numero: quel numero indica la portata massima di un ponte, cioè il peso massimo che è in grado di reggere.

Per comodità, le isole sono state denominate con nomi di isole reali.

I ponti delle isole centrali, essendo i più utilizzati e quindi più importanti, avranno una portata massima maggiore rispetto a quelli che collegano le isole più esterne.



Capitolo 2

Strutture e Informazioni

2.1 INTRODUZIONE

In questo capitolo verranno analizzate le strutture utilizzate, divise in librerie; verranno motivate e spiegate le scelte prese durante la realizzazione di questo sistema.

2.2 GRAFO

Le informazioni relative alle isole sono state raccolte nelle seguenti strutture dati.

```
//Struttura per errori
typedef struct AdjacencyError
{
    int genericError;
} AdjacencyError;

//Struttura per arco
typedef struct AdjacencyArc
{
    int startingKey;
    int destinationKey;
    int pathLength;
    int supportedWeight;
} AdjacencyArc;

//Struttura del nodo della lista di adiacenza.
typedef struct AdjacencyListNode
{
    int destinationKey;
    int pathLength;
    int supportedWeight;
    struct AdjacencyListNode *next;
} AdjacencyListNode;

//Struttura del nodo del grafo.
typedef struct AdjacencyGraphNode
{
    int key;
    int isDisabled;
```

```

    int isVisited;
    int predecessorPosition;
    int distance;
    char *name;
    unsigned numberOfInArches;
    unsigned numberOfOutArches;
    AdjacencyListNode *adjacencyListHead;
} AdjacencyGraphNode;

//Struttura della testa del grafo.
typedef struct AdjacencyGraph
{
    unsigned arrayLength;
    unsigned numberOfNodes;
    AdjacencyGraphNode **nodeArray;
} AdjacencyGraph;

```

Si è scelto quindi di rappresentare il sistema di isole come un grafo non orientato con liste d'adiacenza.

La scelta di tale tipo di insieme, con le sue caratteristiche, è dovuta ai seguenti motivi:

- Facilità di rappresentazione di un sistema con più collegamenti;
- Facilità di navigazione tra i vari nodi del grafo in maniera bidirezionale;
- Risparmio di memoria grazie alla rappresentazione tramite liste;
- Richiesto dalla traccia.

Le funzioni per queste prime strutture sono:

```

/* Prototipi sugli errori */

AdjacencyError *graphlist_newError();
int graph_setErrorCode(AdjacencyError *error, int errorCode);
int graph_getErrorCode(AdjacencyError *error);

/* Prototipi sugli archi */

int graph_setArc(AdjacencyArc *arc, int startingKey, int destinationKey
, int pathLength, int supportedWeight);
int graph_getArcStartingKey(AdjacencyArc *arc, AdjacencyError *error);
int graph_getArcDestinationKey(AdjacencyArc *arc, AdjacencyError *error
);
int graph_getPathLength(AdjacencyArc *arc, AdjacencyError *error);

/* Prototipi sui nodi */

unsigned graph_findPositionByKey(AdjacencyGraphNode **array, unsigned
arrayLength, int keyToFind, AdjacencyError *error);
int graph_isNull(AdjacencyGraph *graph);

```

```

int graph_isEmpty(AdjacencyGraph *graph);
AdjacencyGraph *graph_newGraph(unsigned initialNumberOfNodes);
int graph_reallocNodeArray(AdjacencyGraph *graph, unsigned newLength);
int graph_addNode(AdjacencyGraph *graph, int key);
int graph_addNodeWithName(AdjacencyGraph *graph, int key, char *name);
int graph_deleteNode(AdjacencyGraph *graph, int keyToDelete);
int graph_disableNode(AdjacencyGraph *graph, int keyToDisable);
int graph_deleteGraph(AdjacencyGraph **graph);
void graph_printGraph(AdjacencyGraph *graph);
int graph_addDirectionalArc(AdjacencyGraph *graph, AdjacencyArc *arc);
int graph_addBidirectionalArc(AdjacencyGraph *graph, AdjacencyArc *arc)
;
int graph_deleteBidirectionalArc(AdjacencyGraph *graph, AdjacencyArc
*arc);
unsigned graph_getNumberOfGraphNodes(AdjacencyGraph *graph);
unsigned graph_getNodeGrade(AdjacencyGraph *graph, int key);
int graph_findDijkstraPath(AdjacencyGraph *graph, int startKey,
int destinationKey, int driverWeight);

/* Prototipi sui file */

FILE *graph_createNewDb(char *dbName);
FILE *graph_openDbToAppend(char *dbName);
FILE *graph_openDbToRead(char *dbName);
int graph_closeDb(FILE **dbFile);
int graph_deleteDb(char *dbName);
int graph_writeGraphOnDb(FILE *db, AdjacencyGraph *graph, unsigned
*numberOfWriteNodes, unsigned *numberOfWriteArches);
int graph_loadGraphFromDb(FILE *db, AdjacencyGraph *graph, unsigned
*numberOfLoadNodes, unsigned *numberOfLoadArches);

```

Le funzioni sono divise in quattro categorie: funzioni che operano sugli errori, funzioni che operano sugli archi del grafo, sui nodi e infine funzioni che operano con i file.

Di particolare importanza è la funzione *graph_findDijkstraPath* per la ricerca del percorso minimo.

2.3 DRIVER

Le informazioni relative al driver sono state raccolte nella seguente struttura dati:

```
// Struttura del Driver
typedef struct Driver
{
    char *username;
    char *password;
    char *cf;
    char *name;
    char *surname;
    char *email;
    char *telephone;
    char *address;
    int vehicleWeight;
    char *vehicleName;
    char *plate;
    FoodList *items;
    struct Driver *next;
} Driver;
```

I campi *username* e *password* sono fondamentali per il login del driver, mentre il campo *vehicleWeight* è necessario, dato che indica il peso del mezzo di trasporto. Gli altri campi sono stati inseriti per dare ulteriore realtà al progetto.

Nella struttura Driver è presente una lista di alimenti, che servirà nel momento in cui il driver dovrà effettuare una consegna.

Infine, nella struttura è anche presente un puntatore ad una struttura Driver, inserito qui per facilità di creazione di strutture dati dinamiche.

Si è scelto di rappresentare i Driver tramite una lista singolarmente concatenata allocata dinamicamente.

La scelta di tale tipo di insieme, con le sue caratteristiche, è dovuta ai seguenti motivi:

- Versatilità nell'inserimento di nuovi nodi;
- Migliore gestione dello spazio di memoria nel caso di liste aventi elevate dimensioni;
- Necessità del solo accesso sequenziale.

Per quanto riguarda le funzioni implementate per la struttura dati, sono le seguenti:

```
/* FUNZIONI LISTA */
Driver *driver_newDriver();
Driver *driver_append(Driver *list, Driver *newDriver);
Driver *driver_createList();
int driver_printDriverList(Driver *list);
int driver_deleteDriverByUsername(char *username);
```

```

/* FUNZIONI REGISTRAZIONE */
Driver *driver_initializeDriver();
int driver_checkUsernameExistence(char *username);
int driver_checkEmailExistence(char *email);
int driver_checkCfExistence(char *cf);
int driver_isAllAlpha(char *string, int len);
int driver_isAllDigit(char *string, int len);
int driver_checkUsername(char *username);
int driver_checkPassword(char *password);
int driver_checkCf(char *cf);
int driver_checkName(char *name);
int driver_checkEmail(char *email);
int driver_checkTelephone(char *telephone);
int driver_checkAddress(char *address);
int driver_checkVehicleName(char *vehicleName);
int driver_checkPlate(char *plate);
int driver_registration(Driver *newDriver);

/* FUNZIONI LOGIN */
int driver_login(char *username, char *password, Driver **driver);

/* FUNZIONI STORICO */
int driver_printDeliveryHistory(Driver *d, int startIsland, int destinationIsland, double totalPrice);
int driver_printDeliveryHistoryByUsername(char *username);

```

Le funzioni sono divise in quattro categorie: funzioni che operano sulla lista, funzioni di registrazioni, funzioni di login e funzioni di storico.

Da notare con particolare attenzione le funzioni riguardanti lo storico: le consegne effettuate dal driver, infatti, vengono memorizzate e ogni driver, dopo aver effettuato l'accesso nel proprio account, è in grado di ricontrollare i dettagli delle proprie consegne.

2.4 ALIMENTI

Le informazioni relative agli alimenti sono state raccolte nella seguente struttura dati:

```
typedef struct Food{
    int barcode;
    char *name;
    int available;
    int foodWeight;
    char *brand;
    double price;
    struct Food *next;
} Food;
```

Il campo *foodWeight* è fondamentale per il calcolo totale del peso del camion. Gli altri campi sono stati inseriti per dare ulteriore realtà al progetto.

Nella struttura Food è presente anche un puntatore ad una struttura Food, inserita qui per facilità di creazione di strutture dati dinamiche.

Un'altra struttura utilizzata è la seguente:

```
typedef struct FoodList{
    Food item;
    int quantity;
    struct FoodList *next;
} FoodList;
```

Questa struttura tiene conto degli alimenti che il driver decide di portare, insieme alla quantità scelta.

Anche qui, è presente un puntatore ad una struttura FoodList, per facilità di creazione di strutture dati dinamiche.

Si è scelto di rappresentare sia la struttura Food che la struttura FoodList con delle liste semplicemente concatenate allocate dinamicamente.

La scelta di tale tipo di insieme, con le sue caratteristiche, è dovuta ai seguenti motivi:

- Versatilità nell'inserimento di nuovi nodi;
- Migliore gestione dello spazio di memoria nel caso di liste aventi elevate dimensioni;
- Necessità del solo accesso sequenziale.

Le funzioni implementate per queste due strutture sono:

```
//FUNZIONI GENERICHE////////////////////////////////////
int listFood_isEmpty(FoodList *head);
void listFood_listPrint(FoodList *head);
void listFood_freeList(FoodList **head);
////////////////////////////////////
```

```

//FUNZIONI SU LISTA ALIMENTI DISPONIBILI IN MAGAZZINO//
FoodList *listFood_findPosition(int position, FoodList *head, int cnt);
FoodList *listFood_findFoodByPosition(int position, FoodList *head);
int listFood_listLenght(FoodList *head);
int listFood_loadConfirmedFoodListFromDb(FILE *db, FoodList **head);
int listFood_loadFoodListFromDb(FoodList **head);
int listFood_checkAvailabilityByFoodBarcode(FoodList **head,int barcode
, int quantityTaken);
int listFood_addNewFood(FoodList **head,int barcode, char *name, int av
ailable, int foodWeight, char *brand, double price);
int listFood_removeFoodFromListByBarcode(FoodList**head,FoodList *prev,
int barcode);//prev è passato come NULL inizialmente!
int listFood_writeOnDbFoodListWithoutFoodByBarcode(FoodList **head, int
barcode);
int listFood_updateDbFoodListBeforeCloseSystem(FoodList **head);
FoodList *listFood_findFoodByBarcode(FoodList *head,int barcode);
////////////////////////////////////

//FUNZIONI SU LISTA ALIMENTI TRASPORTATI DAL DRIVER//
int listFood_calculateTotalWeight(FoodList *items);
FoodList *listFood_insertToListItemsForDriver(FoodList *items, int quan
tityTaken, FoodList *cpyNodo);
int listFood_checkAvailability(int foodQta, FoodList *nodo);
int listFood_updateAvailability(FoodList *node, int quantityTaken, Food
List *head);
double listFood_calculateTotalPrice(FoodList *items);
////////////////////////////////////

```

Le funzioni sono divise in tre categorie: generiche, funzioni sulla lista di alimenti in magazzino e funzioni sulla lista di alimenti trasportati dal driver.

Da notare con particolare attenzione la funzione *listFood_calculateTotalWeight*, che calcola il peso totale di una serie di alimenti.

2.5 FILE E SALVATAGGIO DATI

Il salvataggio dei dati avviene su file.

I file contengono semplice testo, i dati sono organizzati in record in modo simil “.csv”.

I file sono i seguenti:

- **foodList.txt** – contiene i record degli alimenti. L’aggiornamento del file viene fatto a fine programma.
- **graph.txt** – contiene le informazioni delle isole per la creazione del grafo. Il file viene aperto ad inizio programma e chiuso alla fine, quindi la chiusura del programma in un altro modo può portare alla compromissione dei dati.
- **login_driver.txt** – contiene i record con le informazioni dei driver. Il file viene aperto e chiuso nelle funzioni in cui viene utilizzato.
- **order_history.txt** – contiene i record con le informazioni sulle consegne effettuate dai driver. Il file viene aperto e chiuso nelle funzioni in cui viene utilizzato.

Capitolo 3

Guida d'uso

3.1 COMANDI E SIMULAZIONE

La GUI è una semplice interfaccia grafica in console, strutturata in menu dove per la navigazione bastano i tasti:

- UP
- DOWN
- ENTER

La scelta della voce del menu è evidenziata dalla comoda freccetta al lato sinistro. Naturalmente, gli altri tasti saranno disponibili in caso di necessità.

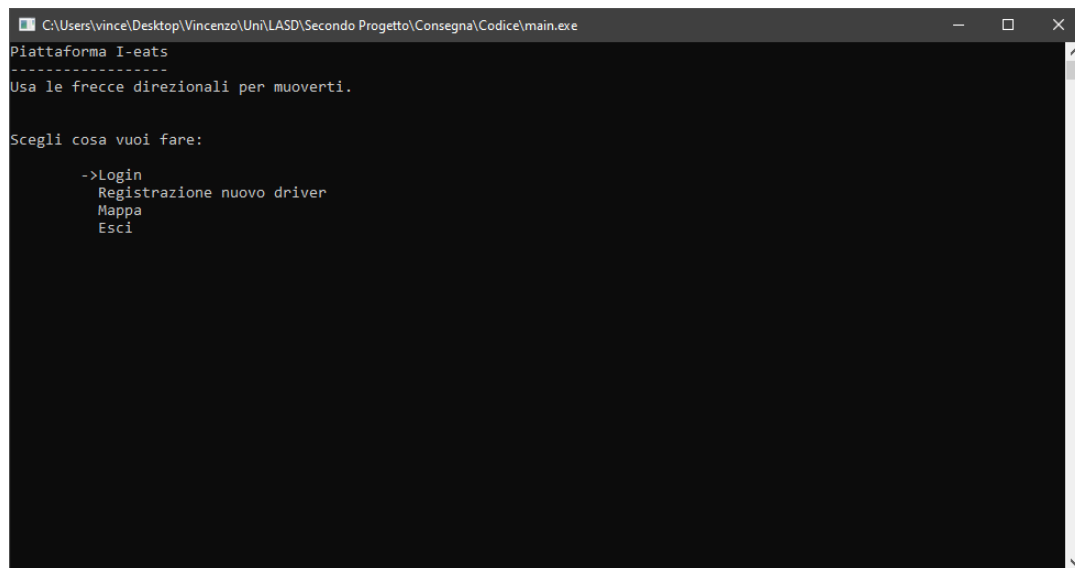
Per permettere una pratica simulazione, sono stati precaricati diversi driver con pesi del mezzo di trasporto differenti.

Si consiglia di provare le seguenti simulazioni:

- **Username** - lucamar
Password - napoli11
Peso del mezzo di trasporto (Kg) - 1200
Itinerario – Lesina [nodo 8] -> Boracay [nodo 1]: non verrà trovato alcun percorso poiché il passaggio del driver dall'isola 'Bora Bora' [nodo 4] a 'Boracay' è impossibile per l'elevato peso del veicolo.
- **Username** - mariorossi
Password - 123456
Peso del mezzo di trasporto (Kg) - 600
Itinerario – Lesina [nodo 8] -> Boracay [nodo 1]: peso complessivo permettendo, il driver è in grado di eseguire questo percorso grazie ad un mezzo di trasporto dal peso adeguato.
- **Username** – fran88
Password – musica18
Peso del mezzo di trasporto (Kg) – 1700
Itinerario – Moorea [nodo 10] -> Lesina [nodo 8]: peso complessivo permettendo, per arrivare a destinazione, il driver dovrà percorrere un cammino più lungo e quindi passare per 'Santorini' [nodo 3] e 'Raronga' [nodo 6], poiché i ponti 'Moorea' – 'Grand Cayman' [nodo 7] e 'Grand Cayman' – 'Lesina' non sono in grado di reggere l'elevato peso del mezzo di trasporto.

3.2 MENU PRINCIPALE

Il menu principale si presenta in questo modo:



```
C:\Users\vince\Desktop\Vincenzo\Uni\LASD\Secondo Progetto\Consegna\Codice\main.exe
Piattaforma I-eats
-----
Usa le frecce direzionali per muoverti.

Scegli cosa vuoi fare:

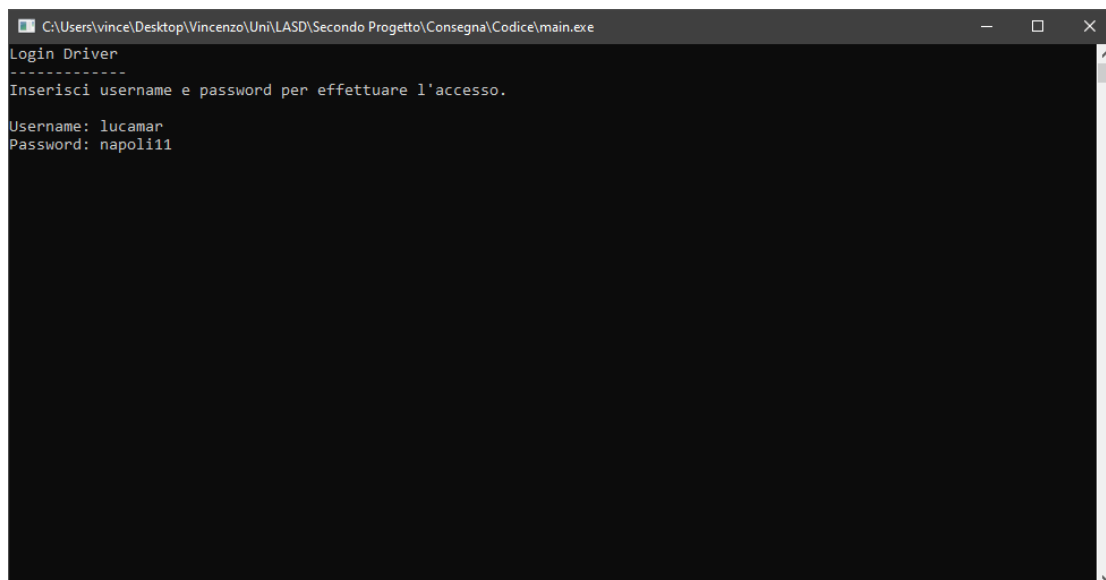
->Login
Registrazione nuovo driver
Mappa
Esci
```

Figura 1 - Menu principale

Da qui è possibile effettuare il login per accedere al menù del driver, registrare un nuovo utente, vedere la mappa delle isole o uscire dal programma.

3.3 MENU DEL DRIVER

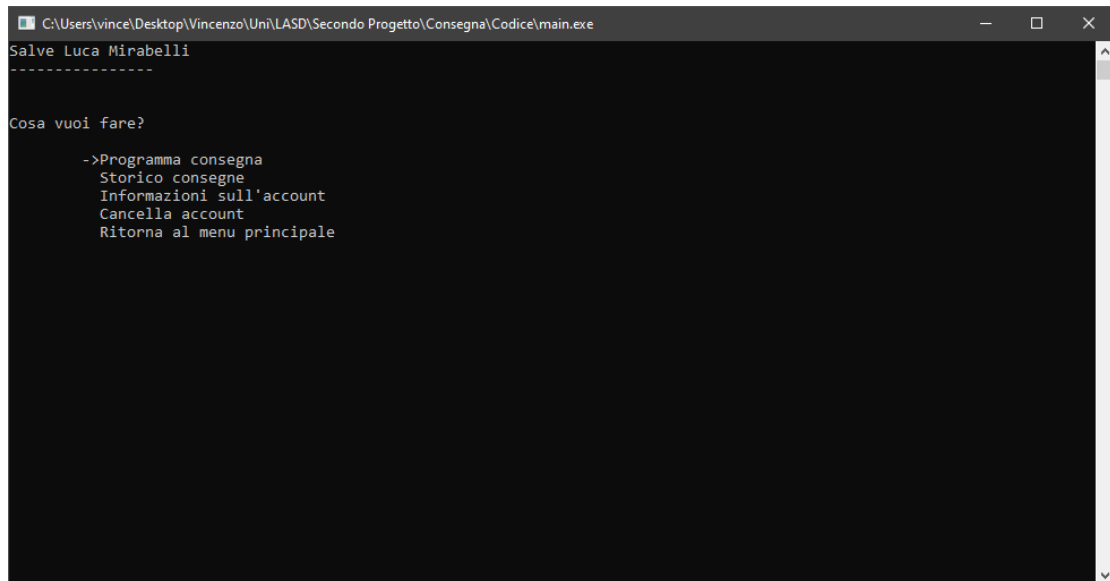
Il driver, per effettuare l'accesso, deve inserire username e password



```
C:\Users\vince\Desktop\Vincenzo\Uni\LASD\Secondo Progetto\Consegna\Codice\main.exe
Login Driver
-----
Inserisci username e password per effettuare l'accesso.
Username: lucamar
Password: napol111
```

Figura 2 - Login driver

Dopo aver effettuato correttamente l'accesso, il driver potrà programmare una consegna, controllare lo storico delle sue consegne, vedere le informazioni del proprio account, cancellare l'account o tornare al menu principale.



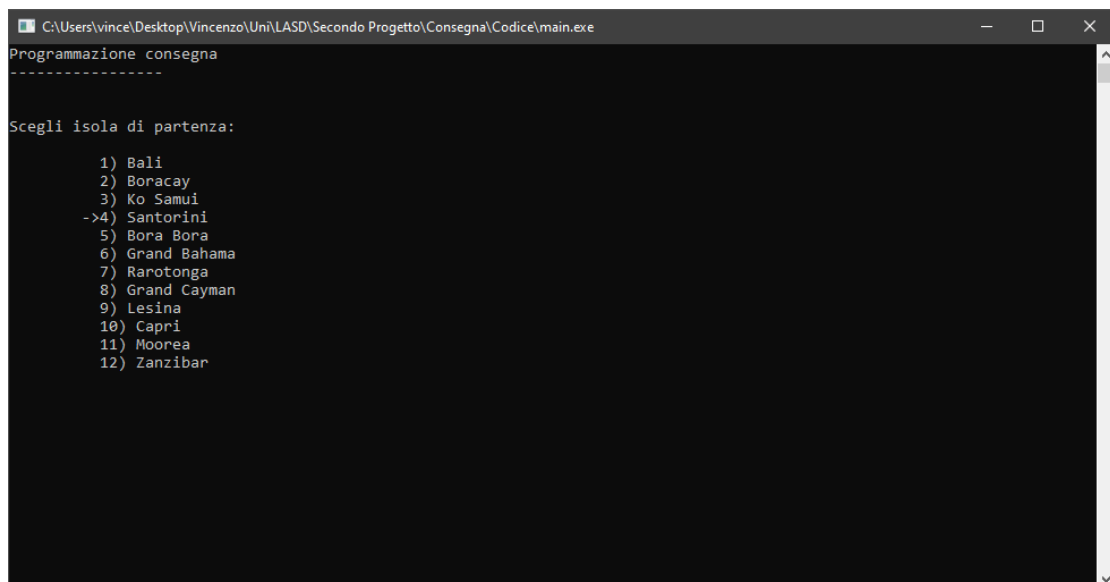
```
C:\Users\vince\Desktop\Vincenzo\Unì\LASD\Secondo Progetto\Consegna\Codice\main.exe
Salve Luca Mirabelli
-----
Cosa vuoi fare?

->Programma consegna
  Storico consegne
  Informazioni sull'account
  Cancella account
  Ritorna al menu principale
```

Figura 3 - Menu driver

3.3.1 Programma Consegna

Per programmare una consegna il driver dovrà selezionare come prima cosa l'isola di partenza e l'isola di destinazione dall'elenco:



```
C:\Users\vince\Desktop\Vincenzo\Unì\LASD\Secondo Progetto\Consegna\Codice\main.exe
Programmazione consegna
-----
Scegli isola di partenza:

  1) Bali
  2) Boracay
  3) Ko Samui
->4) Santorini
  5) Bora Bora
  6) Grand Bahama
  7) Rarotonga
  8) Grand Cayman
  9) Lesina
 10) Capri
 11) Moorea
 12) Zanzibar
```

Figura 4 - Isola di partenza, in questo caso Santorini

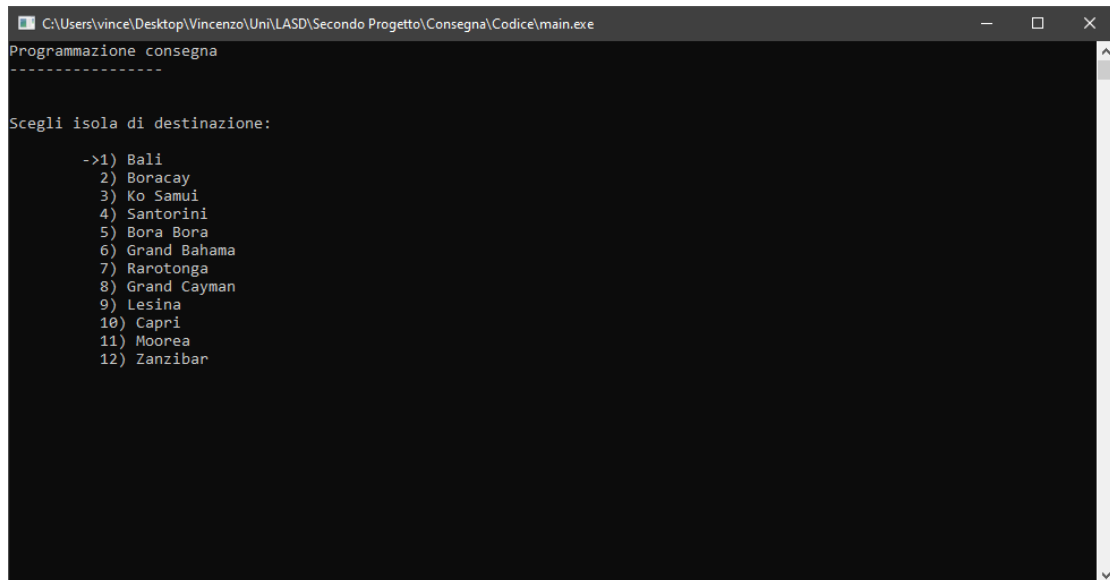


Figura 5 - Isola di destinazione, in questo caso Bali

Dopo aver selezionato partenza e destinazione, il driver dovrà scegliere che prodotti portare con sé da un elenco.

Da qui sarà possibile controllare la disponibilità dei prodotti e il loro peso unitario:

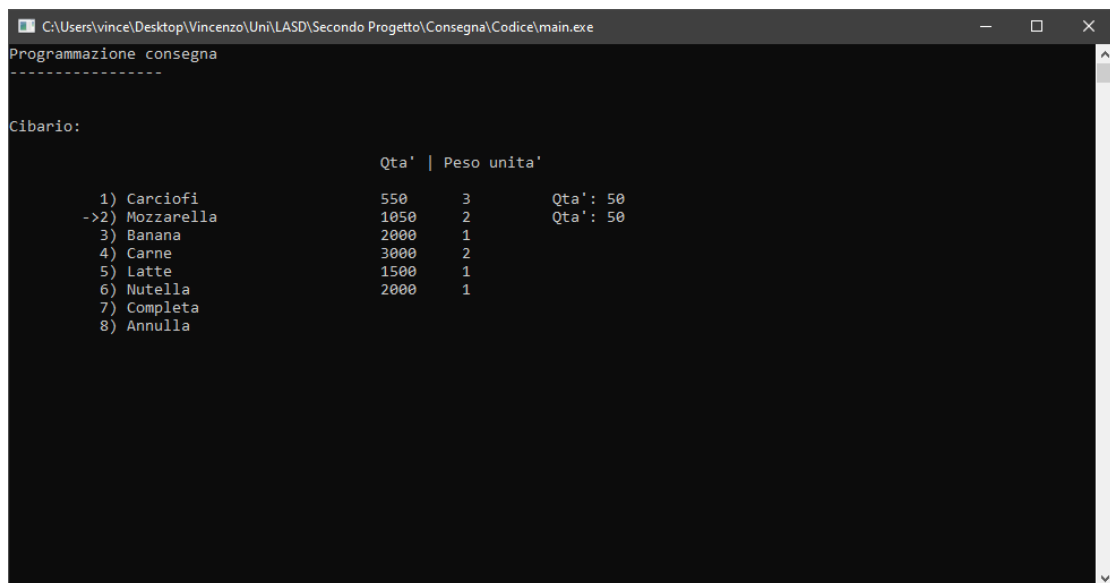


Figura 6 - Scelta dei prodotti

Dopo aver selezionato i prodotti, e aver scelto la relativa quantità, selezionando la voce del menu 'Completa' sarà possibile avere un riepilogo della consegna, con i prodotti, il prezzo totale, il peso totale dei prodotti, il percorso più breve che il driver può effettuare e una conferma per concludere l'ordine:

```

C:\Users\vince\Desktop\Vincenzo\Uni\LASD\Secondo Progetto\Consegna\Codice\main.exe
Riepilogo
-----
Merce da consegnare:

    Prodotto: Carciofi
    Brand: Sempreverde
    Prezzo: 5.60 x 50
    Peso ordine: 150

    Prodotto: Mozzarella
    Brand: Di Battipaglia
    Prezzo: 6.00 x 50
    Peso ordine: 100

Prezzo totale: 580.00
Peso totale: 250
Percorso:
    Santorini
    Rarotonga
    Bali

Sei sicuro/a [s/n]? s

```

Figura 7 – Riepilogo

Nel caso in cui il peso del mezzo di trasporto, sommato al peso di tutti i prodotti, sia maggiore della capacità massima che un ponte è in grado di reggere, allora non sarà possibile raggiungere l'isola in nessun modo. Dopo aver confermato l'ordine, quest'ultimo verrà memorizzato su file e il driver verrà riportato al proprio menu.

3.3.2 Storico consegna

Qui il driver potrà vedere lo storico di tutte le sue consegne.

Qui viene riportata la data, isola di partenza, isola di destinazione ed il prezzo totale della consegna:

```

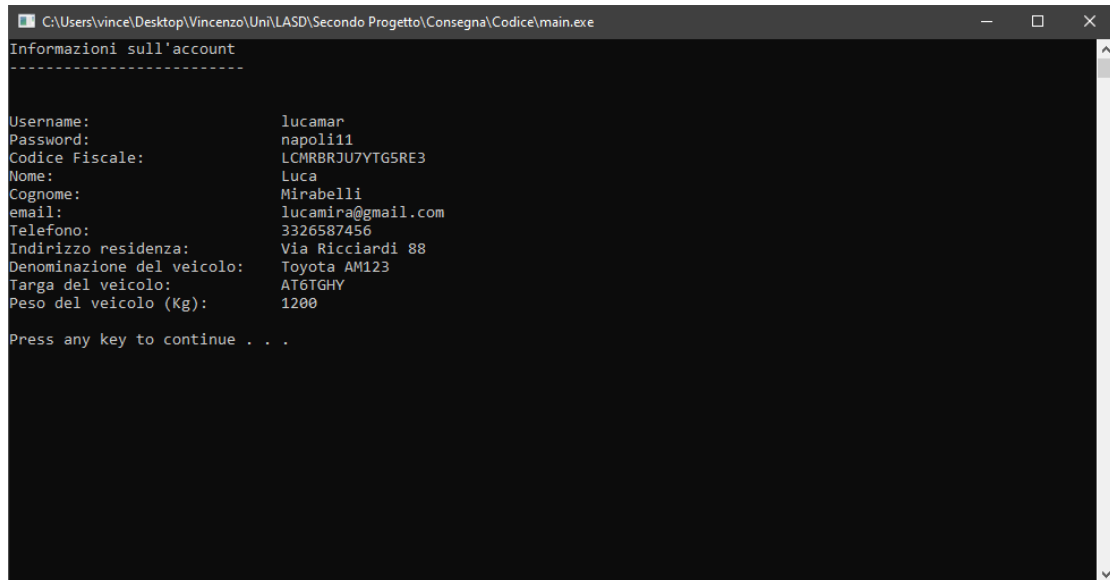
C:\Users\vince\Desktop\Vincenzo\Uni\LASD\Secondo Progetto\Consegna\Codice\main.exe
Storico consegne
-----
Data: 05/26/21 21:24:46
Isola di Partenza: Island 3
Isola di Destinazione: Island 0
Prezzo Totale: 751.00
Press any key to continue . . .

```

Figura 8 - Storico consegne

3.3.3 Informazioni sull'account

Qui sarà possibile avere un riepilogo di tutti i dati inseriti dal driver nel momento della registrazione, inclusi i dati riguardanti il mezzo di trasporto:

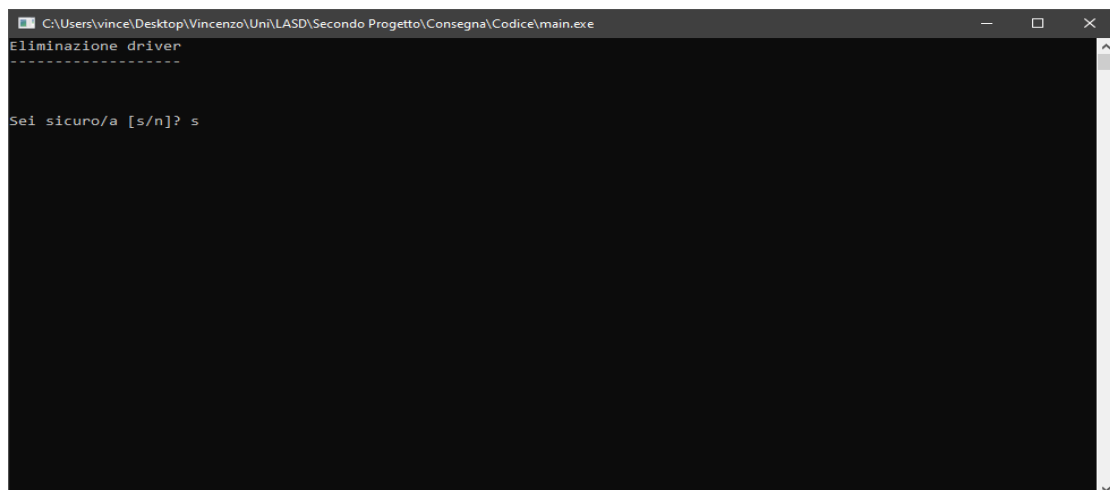


```
C:\Users\vince\Desktop\Vincenzo\Un\LASD\Secondo Progetto\Consegna\Codice\main.exe
Informazioni sull'account
-----
Username:          lucamar
Password:          napoli11
Codice Fiscale:    LCMRBRJU7YTG5RE3
Nome:              Luca
Cognome:           Mirabelli
email:             lucamira@gmail.com
Telefono:          3326587456
Indirizzo residenza: Via Ricciardi 88
Denominazione del veicolo: Toyota AM123
Targa del veicolo:  AT6TGHY
Peso del veicolo (Kg): 1200
Press any key to continue . . .
```

Figura 9 - Informazioni sull'account

3.3.4 Cancella account

Qui il driver, dopo una conferma, può cancellare il proprio account. Se conferma, verrà riportato al menu principale.



```
C:\Users\vince\Desktop\Vincenzo\Un\LASD\Secondo Progetto\Consegna\Codice\main.exe
Eliminazione driver
-----
Sei sicuro/a [s/n]? s
```

Figura 10 - Cancella account

3.4 REGISTRAZIONE

Per registrarsi il driver dovrà inserire tutti i dati richiesti.

I dati inseriti dovranno essere tutti del formato corretto; username, email e codice fiscale non devono essere già presenti nel file.

```

C:\Users\vince\Desktop\Vincenzo\Uni\LASD\Secondo Progetto\Consegna\Codice\main.exe
Registrazione nuovi driver
-----
Benvenuto/a sulla piattaforma.
Inserisci i seguenti dati:

Username: franassi
Password: 123456
Codice fiscale: FRNSSAXXXXXXXX
Nome: Francesco
Cognome: Assi
Email: franassi@gmail.com
Telefono: 3331234567
Indirizzo: Via Italia, 1
Nome del veicolo: TGT-AA-00
Peso veicolo: 1200
Targa del veicolo: ABC00A

Sei sicuro/a [s/n]? s

```

Figura 11 – Registrazione

3.5 MAPPA

Qui è possibile stampare la mappa dell'isola, mettendo in evidenza i collegamenti tra le isole e il peso che un ponte può reggere.

```

C:\Users\vince\Desktop\Vincenzo\Uni\LASD\Secondo Progetto\Consegna\Codice\main.exe
Mappa
-----

Isola: Bali
  Collegamento con: Rarotonga
    Peso max consentito: 3200
  Collegamento con: Grand Bahama
    Peso max consentito: 3100

Isola: Boracay
  Collegamento con: Bora Bora
    Peso max consentito: 600

Isola: Ko Samui
  Collegamento con: Grand Bahama
    Peso max consentito: 3000
  Collegamento con: Zanzibar
    Peso max consentito: 3200

Isola: Santorini
  Collegamento con: Zanzibar
    Peso max consentito: 2800
  Collegamento con: Rarotonga
    Peso max consentito: 3300
  Collegamento con: Moorea
    Peso max consentito: 3000
  Collegamento con: Grand Cayman
    Peso max consentito: 2500

Isola: Bora Bora
  Collegamento con: Boracay
    Peso max consentito: 600

```

Figura 12 – Parziale visione del menu della mappa