

Gianluca Andretta mat. n.0327000510

**Università degli Studi
di Napoli Parthenope**



Corso di Ingegneria Informatica, Biomedica e delle Telecomunicazioni
Esame di Programmazione dei Calcolatori Elettronici

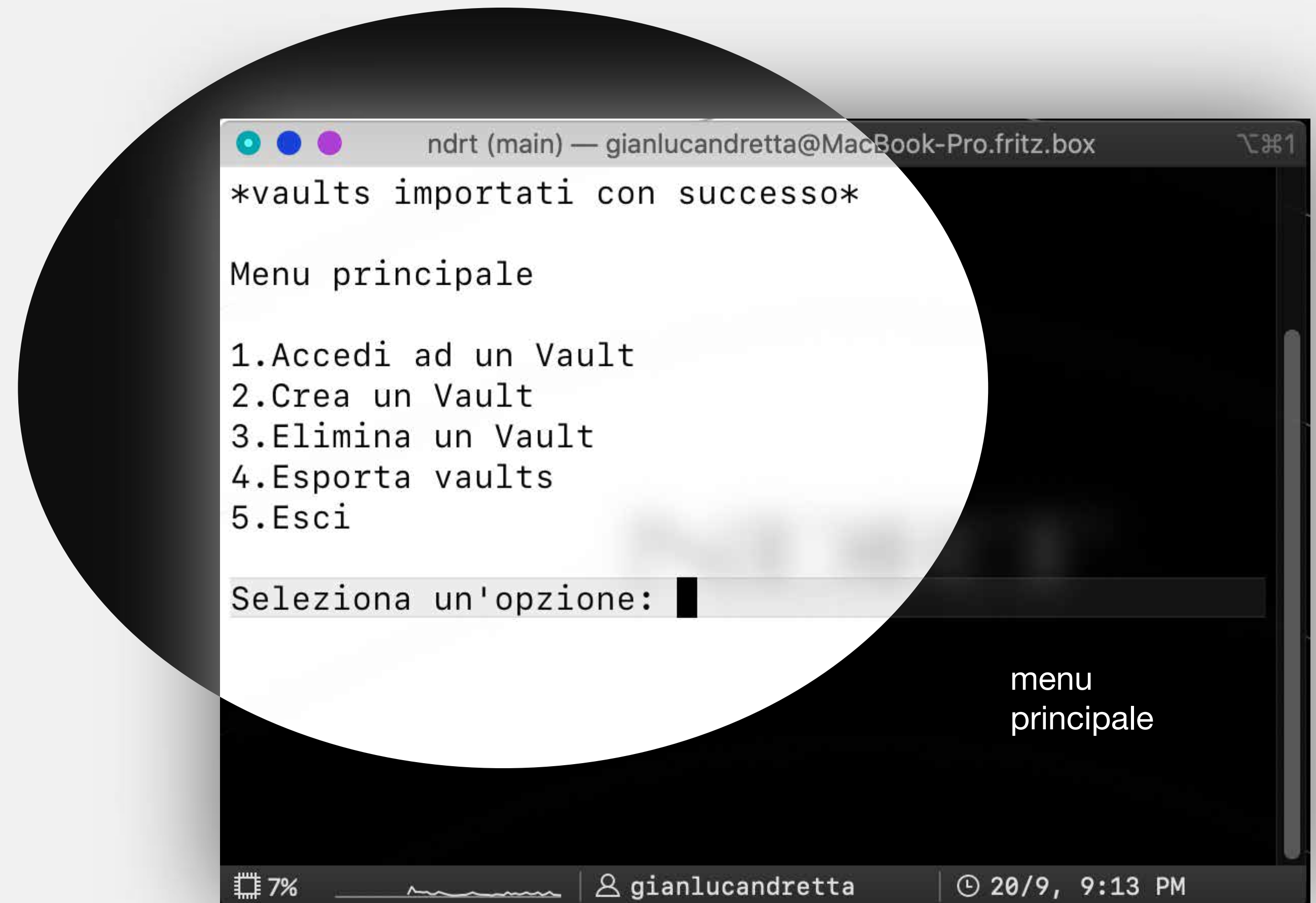
Gestore di password 

Descrizione del progetto

Il programma sviluppato è un gestore di password che registra sotto un'unico account più credenziali di accesso. L'obiettivo è quello di sottrarre all'utente il carico di informazioni sicure da ricordare e di migliorare la complessità delle password scelte. L'utente quindi agevolato dal programma sceglierebbe password più lunghe, meno prevedibili e ricorrenti.

Il programma quindi è stato arricchito con queste funzioni :

- Creazione di più Vault
- Importazione/esportazione automatica
- Generare password di lunghezza variabile
- autenticazione Vault



Test / fasi del progetto

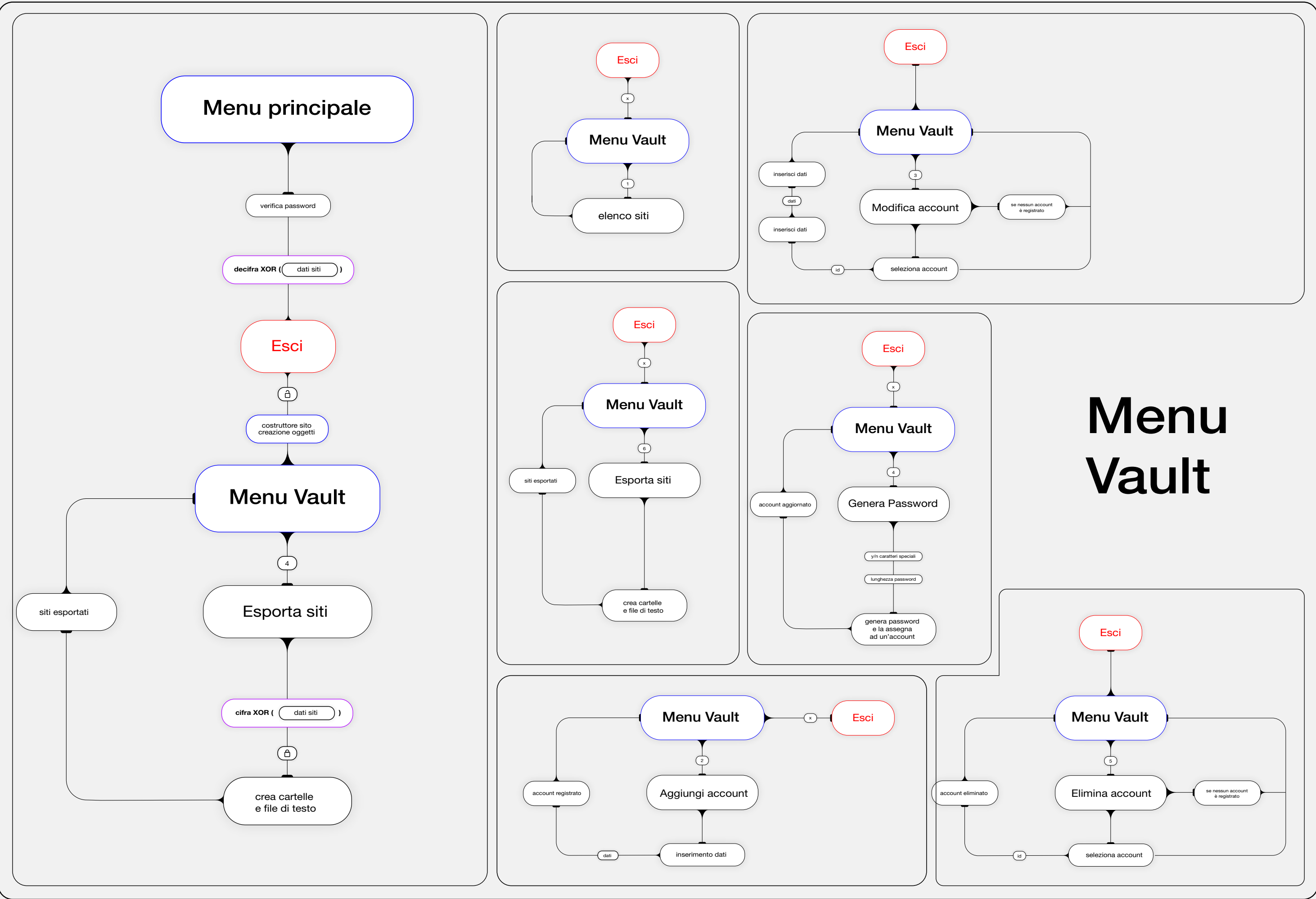
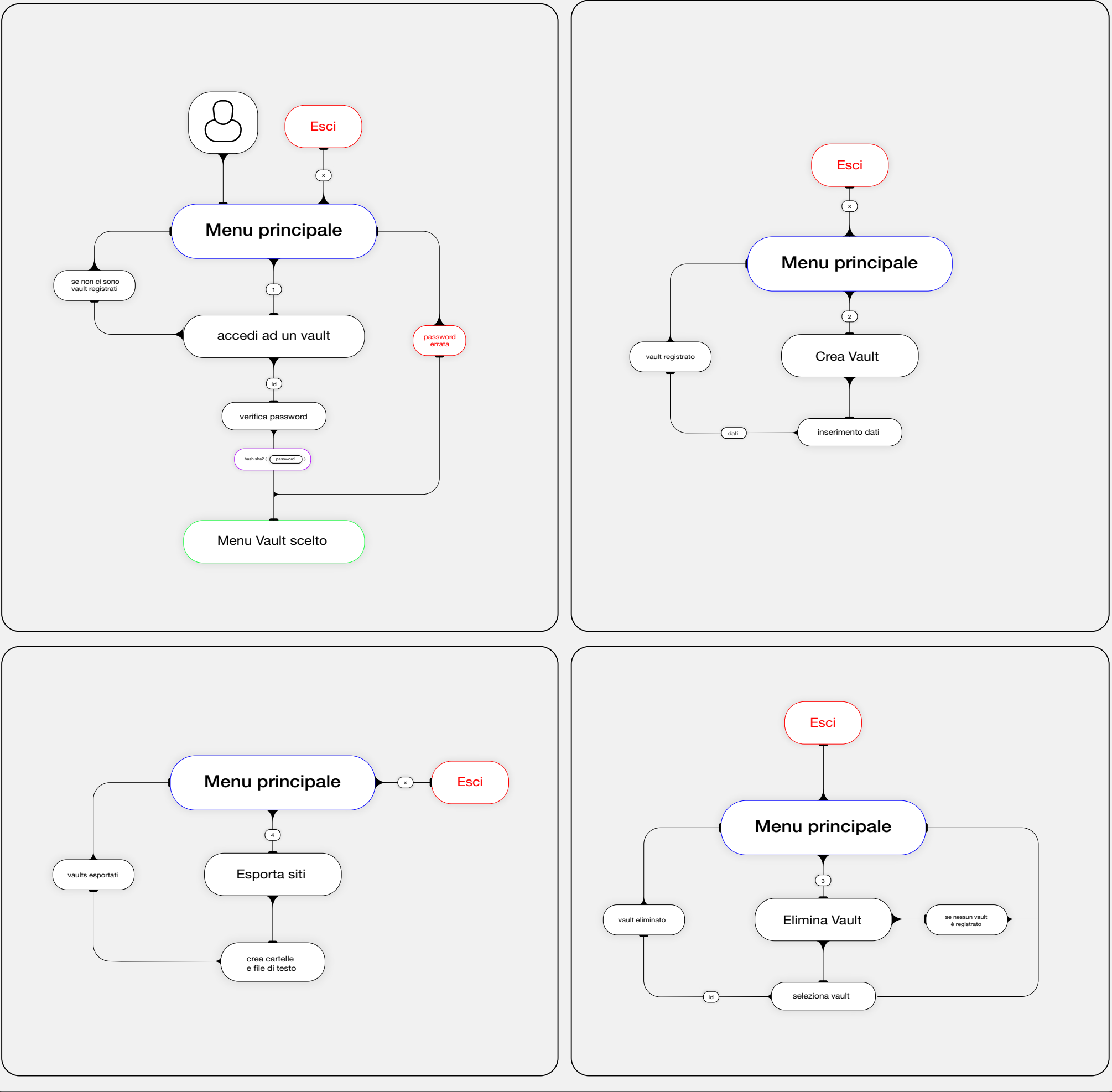


Azione		Risultato atteso	Risultato ottenuto
1	Inserimento dati	corretta istanziatura della classe	creazione di costruttori utili all'inserimento ✓
2	Salvataggio dati	esportazione di dati in file di testo	metodo per esportare i vault e i relativi siti in cartelle distinte ✓
3	Cifratura file	scelta dell'algoritmo di cifratura	corretta implementazione della cifratura XOR ✓
4	Importazione vaults	lettura del file e creazione degli oggetti	metodo per importare i vault da un file di testo ✓
5	Autenticazione utente	verifica degli accessi ai vaults	metodo con verifica dell'hash della password ✓
6	Generatore password	metodo per generare caratteri randomici	metodo per generare stringhe con lunghezza variabile e caratteri speciali ✓
7	Eliminazione account	metodo per eliminare gli account	metodo per eliminare gli account e per la selezione dell'account ✓

Diagramma use-case



Menu Principale



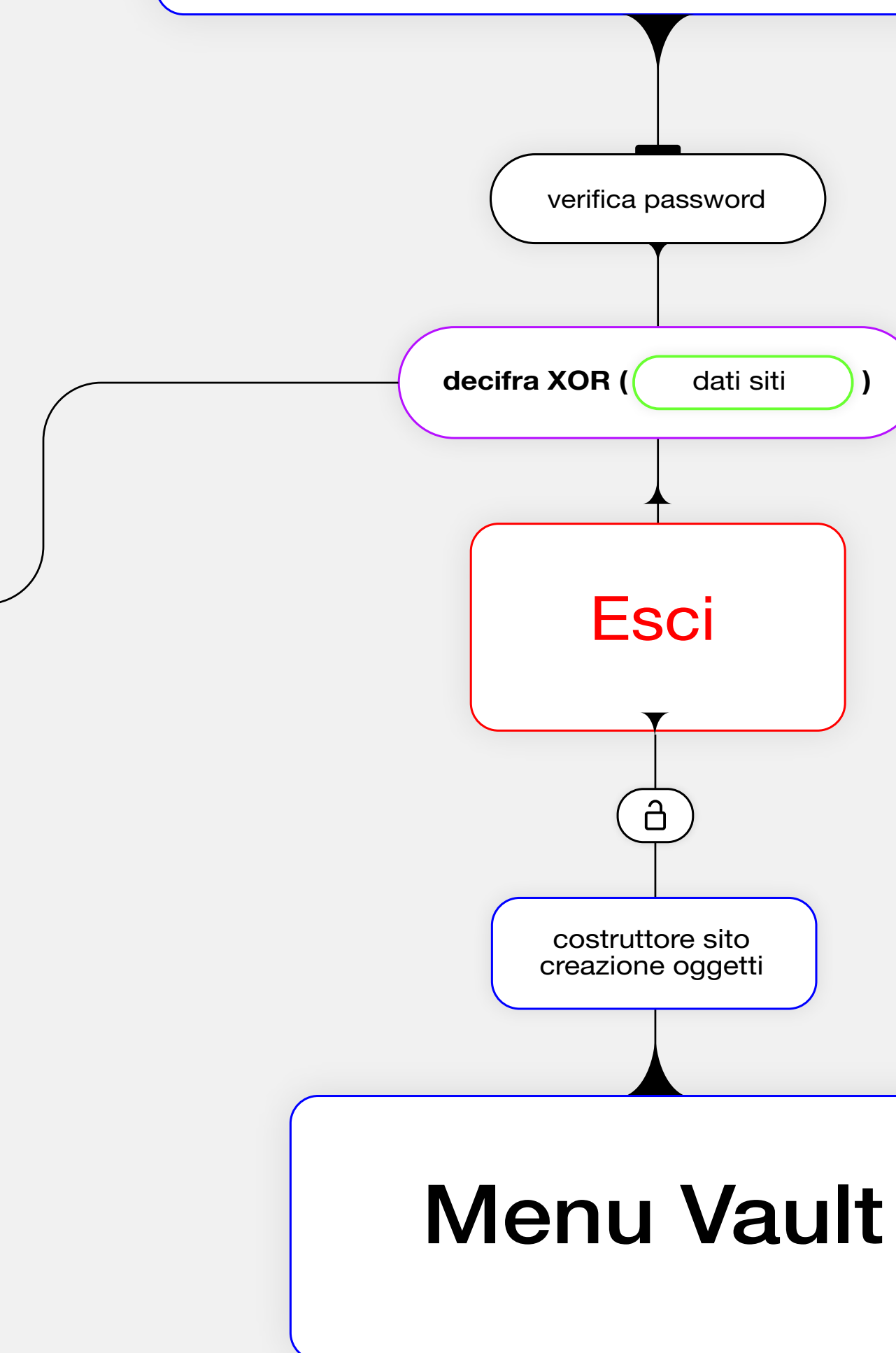
XOR cipher

Come algoritmo per crittografare i file di testo ho scelto XOR, un'algoritmo di cifratura simmetrica che usa il principio della porta logica da cui ne deriva il nome. L'algoritmo è simmetrico quindi è necessaria un'unica funzione sia per la cifratura che per l'operazione opposta

G	01000111	carattere da cifrare
p	01110000	chiave
7	00110111	carattere ascii risultante

Dunque prima di esportare le credenziali di accesso dei siti registrati, questi ultimi vengono prima formattati e poi passati in ingresso alla funzione. In uscita il contenuto è cifrato e pronto per essere salvato su un file di testo. In oltre la chiave utilizzata per crittografare il contenuto del file varia per ogni vault e dipende dalla password di accesso del vault stesso di cui solo l'hash viene conservato.

Menu principale





Password Generator

Il metodo scritto permette all'utente di generare una password di lunghezza variabile e con caratteri speciali.

Il numero di ogni tipo di carattere utilizzato dalla funzione è calcolato in base alla lunghezza scelta dall'utente.

Le varie sequenze di caratteri ottenuti randomicamente vengono poi unificate in un'unica stringa i cui caratteri vengono infine rimescolati.

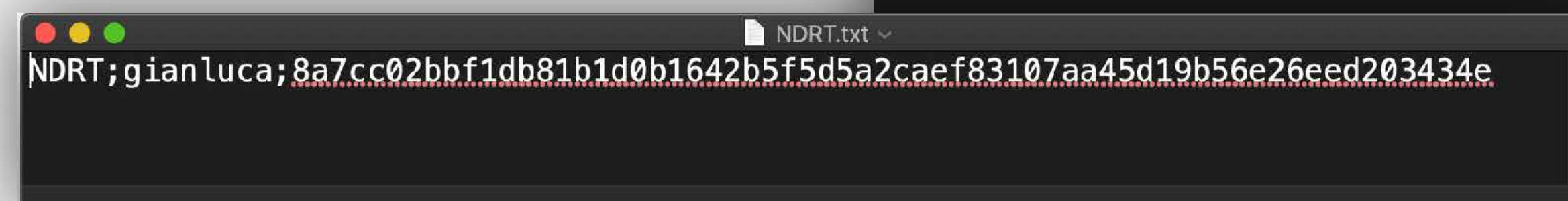
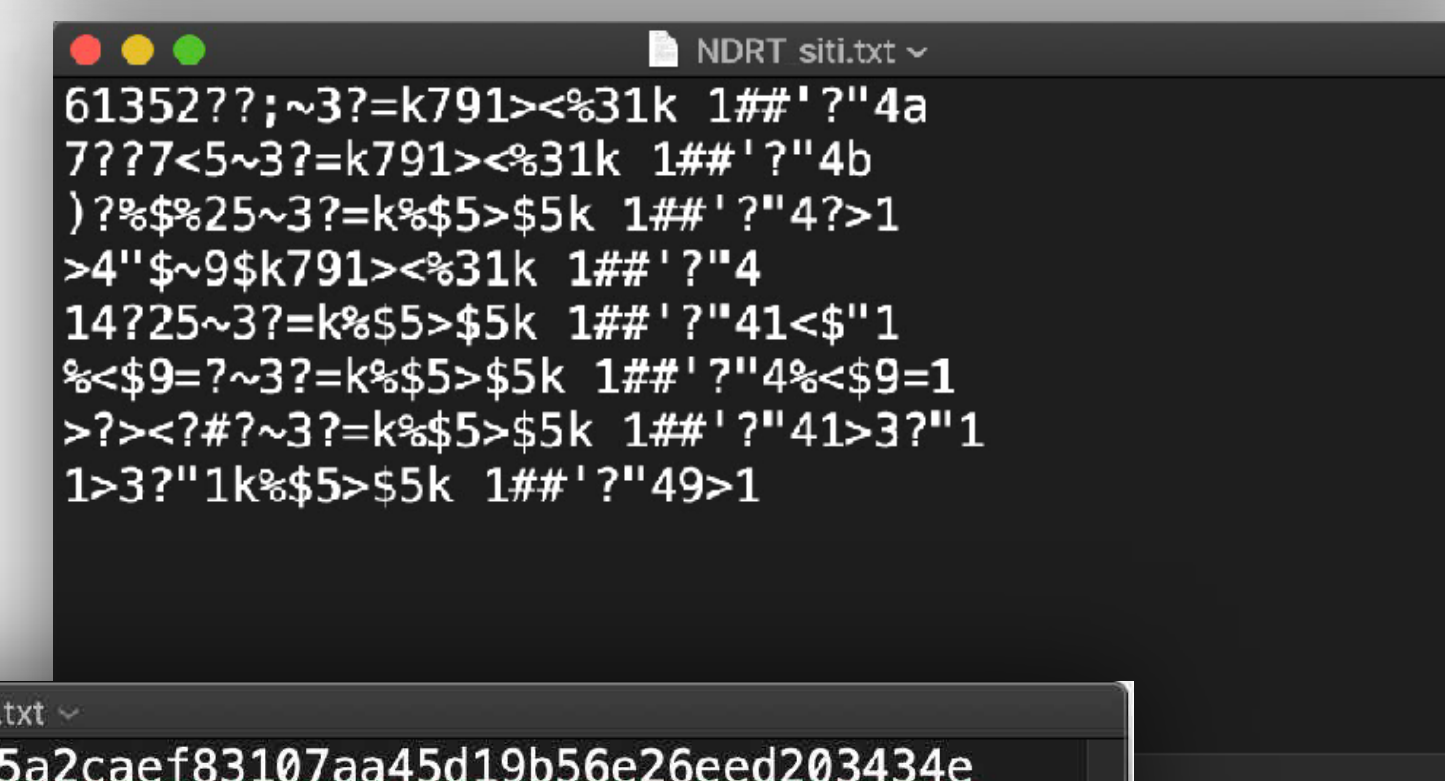
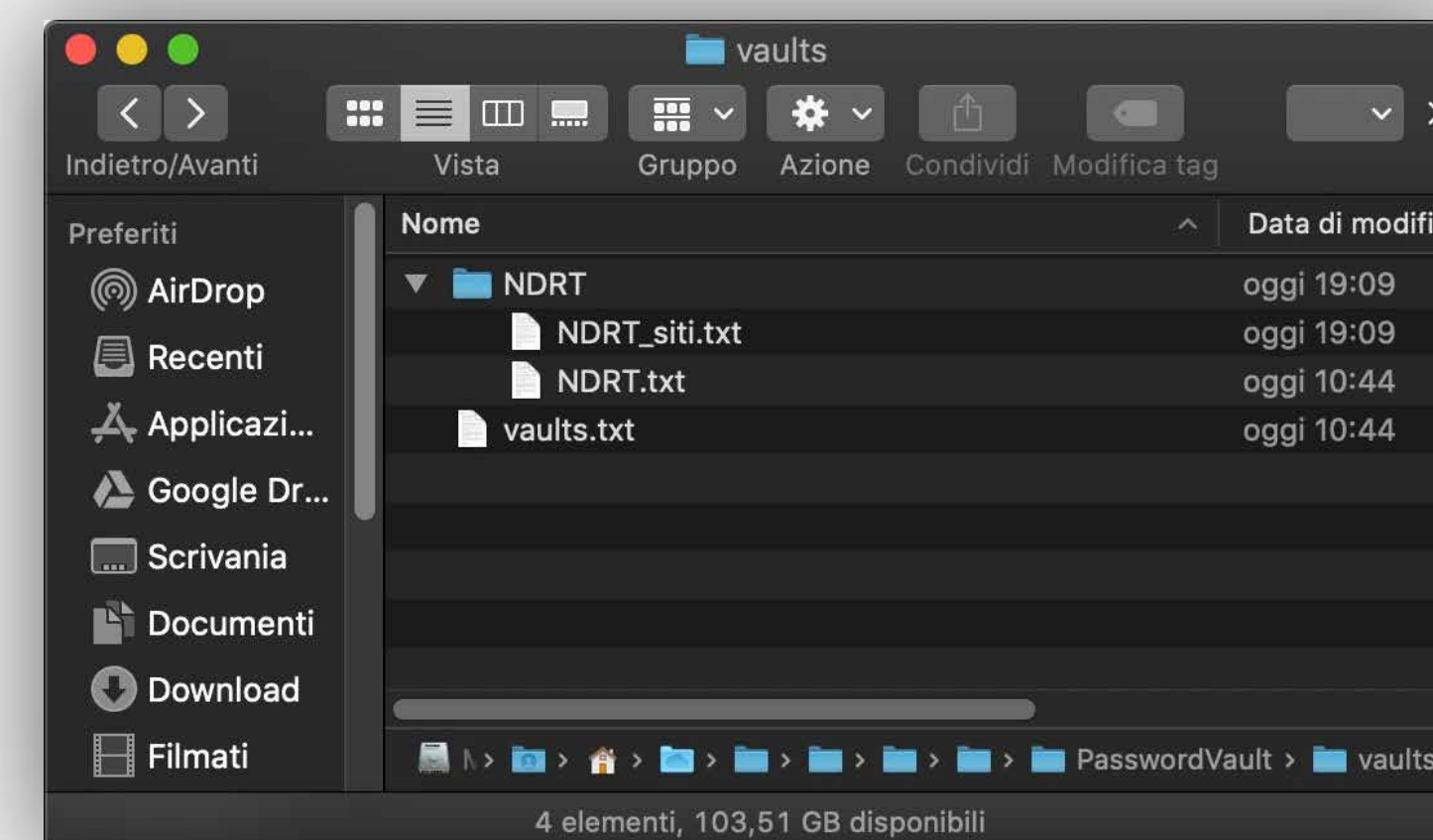
```
string caratteri[4] = {  
    "abcdefghijklmnopqrstuvwxyz",  
    "ABCDEFGHIJKLMNOPQRSTUVWXYZ",  
    "0123456789",  
    "!@#$%&- ",  
};
```

Struttura file

Ad ogni modifica i dati vengono salvati e scritti sui file di testo. I nomi dei vault salvati vengono conservati in chiaro in un file nella cartella vaults. Per ogni vault viene creata una cartella con due file:

- NomeVault.txt
- NomeVault_siti.txt

Il primo contiene i dati per la verifica di accesso e il secondo gli account registrati nel vault cifrati con XOR





Legenda
▲ provenienza/classe base
■ arrivo/classe derivata

classe Vault:

string vaultName;
vector<Sito*> siti;

vector alla classe Siti*

```
Vault();  
Vault(string*, string*, string*);  
~Vault();  
  
void menuVault();  
  
void set_vaultName(string );  
string get_vaultName();  
virtual void set_password(string);  
int get_numeroSiti();  
void get_siti();  
  
void riepilogo();  
int seleziona_sito();  
void aggiungi_sito();  
void modifica_sito();  
void elimina_sito(int );  
  
void importa_siti();  
void esporta_siti();
```

classe Sito:

string dominio;

```
Sito();  
Sito(string*, string*, string*);  
~Sito();  
  
void set_dominio(string s);  
string get_dominio();  
  
void riepilogo();
```

vector di puntatori alla classe Vault

classe Account:

email: string;
password: string;

```
Account();  
Account(string, string);  
~Account();  
  
void set_email(string);  
string get_email();  
void set_password(string);  
string get_password();  
  
template<class arr>  
struct arrfuncs{  
    void disordinarr(arr*);  
    void selezionarr(arr*, arr*, int);  
};  
  
func: arrfuncs<string> func;
```

classe Menu:

vaults: vector<Vault*>;
pass_key: static char;

```
Menu();  
~Menu();  
  
void mainMenu();  
void accedi_Vault();  
int select_Vault();  
bool check_password(int);  
void aggiungi_Vault();  
void elimina_Vault(int);  
int get_numeroVaults();  
void get_vaults();  
void esci_Vault();  
  
static vector<string> read_file(string );  
static void write_file(string , string*);  
void importa_Vault(vector<string> )  
void esporta_Vault();  
static bool riprova();  
static void pulisciSchermo();  
static void make_dir(string);  
static string sbarrett();  
Astatic void XOR(string*);  
static void encrypt(string*);  
static void decrypt();a
```