

Introduzione

Fallout 32 è un'avventura grafica ambientata nell'universo post-apocalittico di Fallout. Il giocatore assume il ruolo di un abitante del Vault, un rifugio sotterraneo progettato per proteggere l'umanità dalla devastazione nucleare.

Ispirazione

Per lo sviluppo il team si è ispirato all'universo narrativo di Fallout, gioco di ruolo nato nel 1997.

In particolare sono stati presi in considerazione Fallout 3 e la serie televisiva "Fallout".

Capitolo 1

Trama

La vita nel Vault 32 è all'apparenza normale: gli abitanti sono in perfetta salute, cibo e acqua non scarseggiano e il morale sembra alto.

Eppure, poco a poco, qualcosa sembra rompersi: i residenti si fanno più diffidenti, si dividono in gruppi, l'aggressività aumenta.

Quello che sembrava essere un paradiso, si rivela essere un inferno.

Scoppiano le rivolte, un abitante viene ucciso mentre altri vengono feriti.

L'amministrazione del Vault si dimostra incapace nel gestire la situazione, il sovrintendente si trincerava nell'ufficio protetto da una scorta e le strutture sociali e di sicurezza iniziano a crollare.

Di fronte alla crescente anarchia l'unica speranza di sopravvivenza è la fuga dal Vault.

Obiettivi del gioco

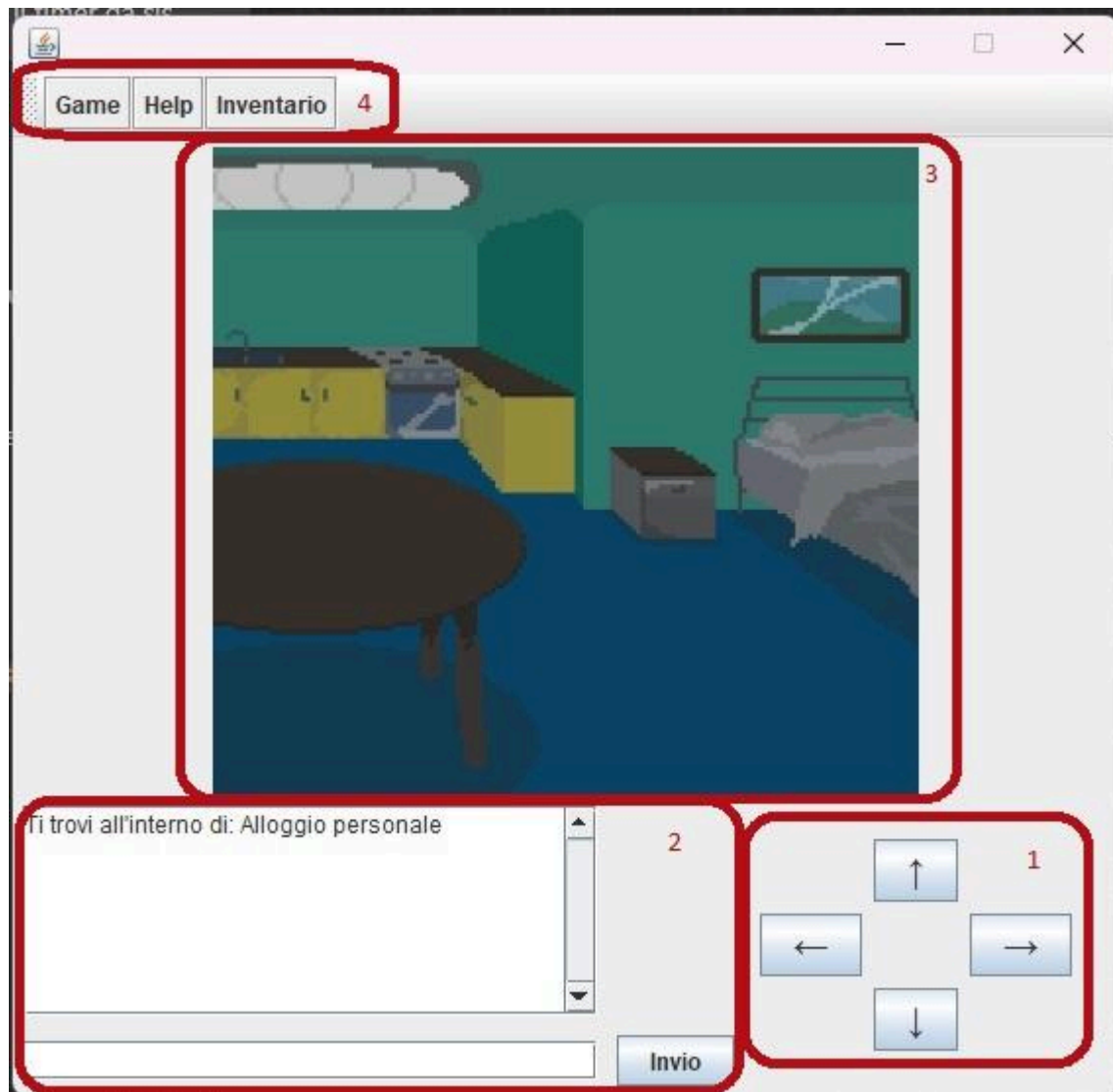
L'obiettivo del gioco è scappare dal Vault 32. Per farlo, il giocatore dovrà risolvere enigmi ambientali.

Sarà fondamentale osservare attentamente gli ambienti e scovare gli oggetti utili alla fuga.

Come si gioca

Per avviare il progetto è necessario inserire le proprie credenziali del DB H2 all'interno del file JDBC.java.

Il sistema è diviso in frontend e backend, è necessario avviare entrambe le parti per far sì che il sistema funzioni correttamente









La schermata di gioco è divisa in 4 zone:

1. **Navigazione nelle Stanze:** in basso a destra sono presenti i tasti per il movimento del personaggio. Permettono lo spostamento nelle diverse stanze del Vault. In prossimità di un ascensore compariranno a destra due tasti che permetteranno di salire o scendere di piano;
2. **Terminale:** in basso a sinistra è presente il terminale. Qui si potranno leggere le descrizioni degli ambienti. Consente inoltre di interagire con l'ambiente circostante e di eseguire azioni specifiche (es. Osserva Alloggio personale, Esamina Comodino);
3. **Immagine di gioco:** in questa sezione verrà visualizzata la stanza corrente;
4. **Menù di gioco:** il menù è formato da tre tasti:
 - a. Game: in cui si potrà salvare e caricare il gioco
 - b. Help: in cui si potranno consultare in ogni momento i comandi di gioco
 - c. Inventario: visualizza tutti gli oggetti raccolti con annessa una breve descrizione

Lista comandi

Fallout 32 usa un sistema di comando ibrido, ovvero che fa uso di un'interfaccia grafica per alcune azioni basilari mentre comandi testuali per azioni più complesse.

Per muoversi bisogna usare le frecce direzionali visualizzate a schermo:

- Click sulla freccia  per muoversi verso Nord;
- Click sulla freccia  per muoversi verso Sud;
- Click sulla freccia  per muoversi verso Est;
- Click sulla freccia  per muoversi verso Ovest;
- Click su : Quando sei in ascensore per salire di piano;
- Click su : Quando sei in ascensore per scendere;

Per interagire con l'ambiente circostante si dovranno digitare sul terminale dei comandi.

In particolare digitare:

- **Osserva** per avere una descrizione della stanza (Es. "Osserva alloggio personale");
- **Esamina** per ispezionare un mobile (Es. "Esamina armadietto");
- **Usa** per utilizzare un oggetto su qualcosa (Es. "Usa fusibile su ascensore");

Alla fine fare click su **Invio** per inviare il comando testuale.

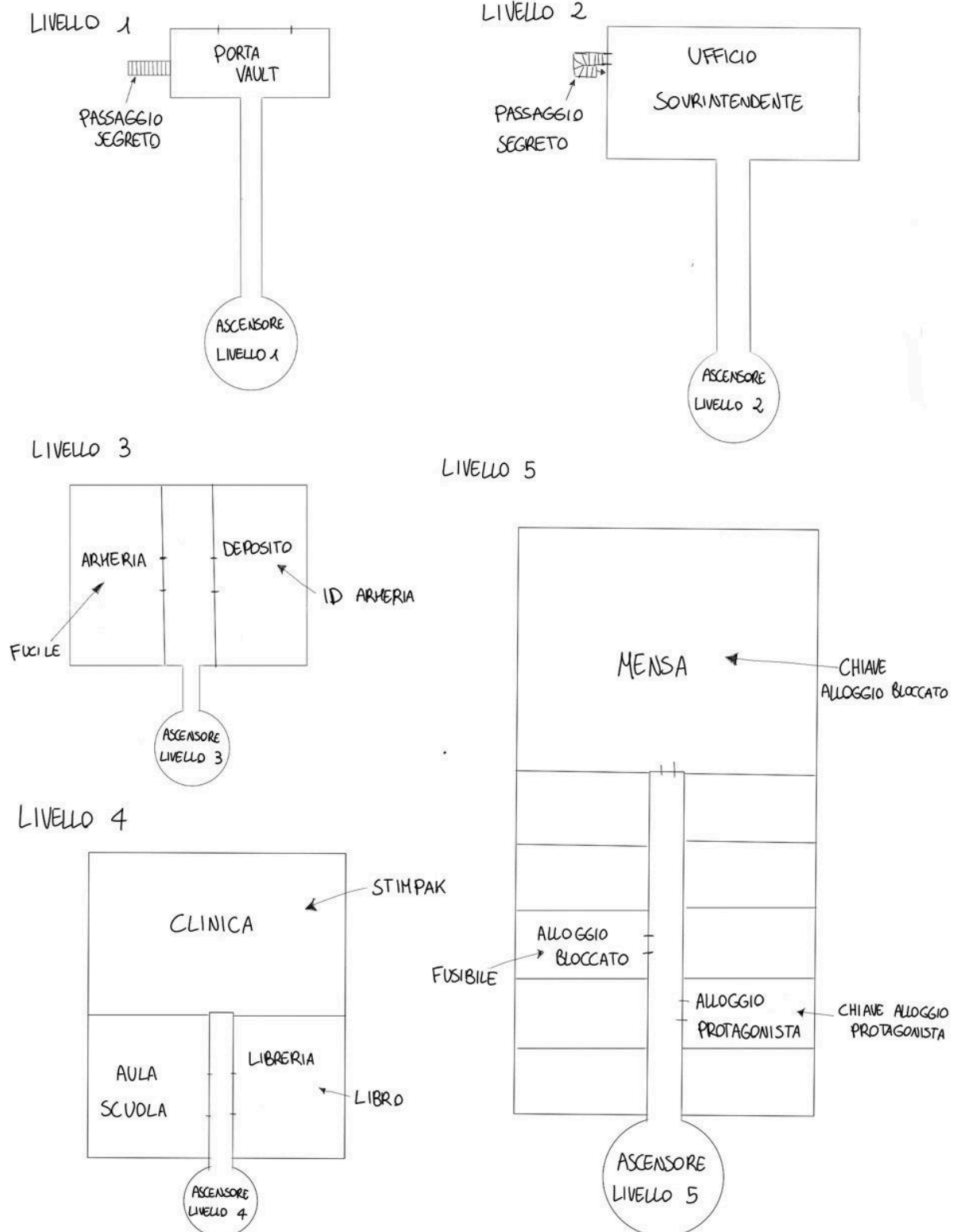
Per salvare o caricare una partita andare su "**Game**" nella toolbar e fare:

- Click su **Carica Partita** per caricare una partita;
- Click su **Salva Partita** per effettuare un nuovo salvataggio ;

Per visualizzare gli oggetti raccolti fare click su **Inventario**.

Fare click su **Help** per visualizzare tutti comandi di gioco.

Mappa del gioco



Capitolo 2

Specifiche Algebriche

Specifica Sintattica

Sorts: lista, boolean, item, posizione;

Operations:

- creaLista() \rightarrow Lista
- listaVuota(lista) \rightarrow boolean
- leggiLista(posizione, lista) \rightarrow item
- scriviLista(item, posizione, lista) \rightarrow lista
- primoLista(lista) \rightarrow posizione
- ultimoLista(lista) \rightarrow posizione
- fineLista(posizione, lista) \rightarrow boolean
- succeLista(posizione, lista) \rightarrow posizione
- predLista(posizione, lista) \rightarrow posizione
- insLista(item, posizione, lista) \rightarrow lista
- canclLista(posizione, lista) \rightarrow lista

Specifica Semantica

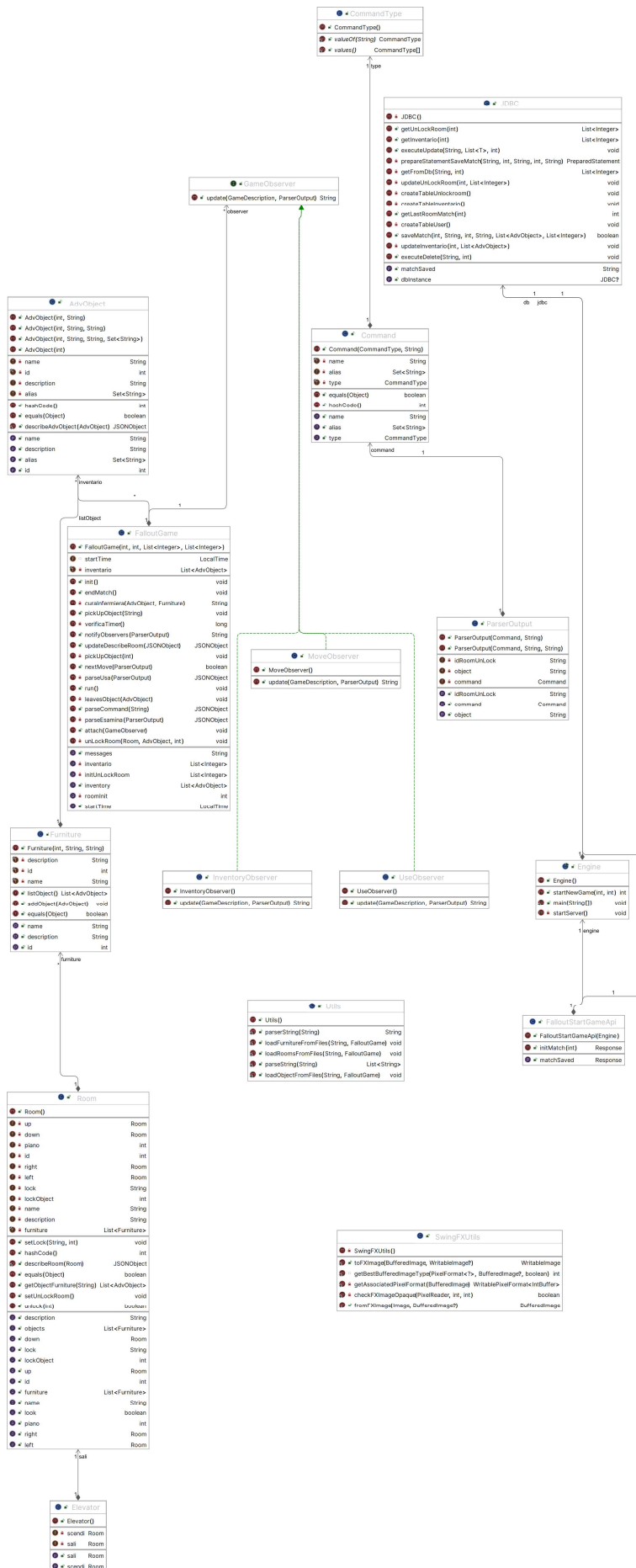
- **Declare** l: lista, i: item, p: posizione;
- **Operations**

| Osservazioni | Costruttore | Costruttore |
|-----------------------|-------------|-------------------|
| | creaLista | insLista(i, p, l) |
| listaVuota(l') | True | False |
| leggiLista(p, l') | error | item |
| scriviLista(i, p, l') | error | l |
| primoLista(l') | p | p |
| ultimoLista(l') | p | p |
| fineLista(p, l') | true | true |
| succeLista(p, l') | error | p |

| | | |
|---------------------|-------|---|
| predLista(i, p, l') | error | p |
| cancLista(p, l') | error | l |

Specifica di Restrizione

- leggiLista(creaLista()) = error
- scriviLista(creaLista()) = error
- succeLista(creaLista()) = error
- predLista(creaLista()) = error
- canclista(craLista()) = error



Dettagli di implementazione

FILE

I file hanno un ruolo cruciale all'interno del progetto dal momento che, attraverso essi, è possibile configurare la mappa del gioco, l'arredamento presente nelle singole stanze e gli oggetti presenti all'interno.

I file usati sono tutti di tipo json e sono 3:

- ROOM

Questo file contiene al suo interno tutte le stanze presenti all'interno della mappa. Ogni elemento del file è strutturato nel seguente modo:

```
"Ufficio sovrintendente":
{
  "type": 0,
  "id": 15,
  "descrizione" : "Questo è l'ufficio del sovrintendente",
  "isLock" : [{
    "objectLock": 1,
    "description Lock": "L'alloggio è bloccato utilizza la chiave per accedere"
  }],
  "piano": 2,
  "directions" : [{
    "down": 18
  }]
}
```

DESCRIZIONE

- **type** → indica il tipo di room (0 stanza normale, 1 ascensore)
- **id** → id della stanza;
- **descrizione** → descrizione della stanza (mostrata nel momento in cui usiamo il comando osserva);
- **isLock** → se è False la stanza è accessibile. In caso contrario (come sopra) la stanza è bloccata e viene specificato l'oggetto necessario a sbloccarla. Inoltre, viene fornita una descrizione del blocco, utile per dare degli indizi al giocatore;
- **piano** → indica il piano in cui è situata la stanza;
- **directions** → contiene una lista di elementi del tipo "direzione" : "id stanza destinazione". "Direzione" può assumere i seguenti valori --> UP, DOWN, RIGHT, LEFT, SALI, SCENDI;

- OBJECT

Questo file contiene al suo interno tutti gli oggetti presenti all'interno delle varie stanze. Ogni elemento del file è formato nel seguente modo:

```
"Fusibile": [{  
  "id": 2,  
  "furniture": 6,  
  "descrizione" : "Fusibile che ripara il guasto dell'ascensore"  
}]
```

DESCRIZIONE

- **id** → id univoco dell'oggetto;
- **furniture** → indica l'id dell'arredamento sulla quale è posto l'elemento;
- **descrizione** → contiene la descrizione dell'oggetto;

- FURNITURE

Questo file contiene al suo interno tutti gli elementi di arredamento presenti nelle varie stanze.

Ogni elemento del file è strutturato nel seguente modo:

```
"Letto Alloggio Protagonista": [{  
  "id": 0,  
  "stanza": 0,  
  "descrizione" : "Letto del protagonista"  
}]
```

DESCRIZIONE

- **id** → contiene l'id dell'arredamento;
- **stanza** → indica l'id della stanza in cui è situato;
- **descrizione** → descrizione dell'arredamento;

DATABASE

All'interno del progetto i database hanno lo scopo di salvare i progressi delle singole partite. I dati vengono divisi in varie tabelle:

- **match** → contiene i dati dei vari match, nella quale vengono salvati nome, idUltimaStanza, nomeUltimaStanza
- **inventario** → contiene l'inventario corrispondente ai vari match
- **unlockroom** → contiene la lista di tutte le stanze sbloccate nei vari match

THREAD

I thread all'interno del progetto ci danno la possibilità di avere più occorrenze del gioco, le quali essi sono completamente indipendenti tra di loro.

Ogni qual volta viene avviata una nuova istanza del frontend viene richiesto al server di avviare un nuovo thread, che a sua volta avrà il compito di gestire univocamente quell'istanza.

REST API

Le rest api hanno svolto un ruolo fondamentale, dal momento che esse hanno il compito di far comunicare il frontend con il backend.

Le due parti comunicano attraverso richieste get fatte su uno specifico path.

Nel momento in cui viene richiesta una nuova istanza del gioco viene fatta richiesta al server di avviare una nuova istanza, il quale se accetta comunica la porta sulla quale verranno fatte le successive richieste per gestire i comandi di gioco.

SWING

Le swing sono state utilizzate per creare l'interfaccia di gioco, che l'utente può utilizzare per giocare. Le swing contengono quindi vari pulsanti, ognuno di essi genera una richiesta al backend.

Importante evidenziare che la parte del frontend sviluppata con le swing non contiene la logica del gioco.

La logica del gioco è quindi gestita completamente dal backend.

LAMBDA EXPRESSION

Le lambda expression sono state inserite all'interno del frontend. Esse sono state utilizzate molto frequentemente per aggiungere ai vari pulsanti pezzi di codice generati da specifici eventi generati dall'utente