

Presentazione Elaborato Esame di Stato 2021

Sartini Matteo

Parte di informatica:

Obiettivo: Implementare il lavoro svolto nell'esperienza PCTO aggiungendo una funzionalità di salvataggio tramite database sviluppato per SQL Server.

Problemi affrontati:

- Ideazione e produzione schema ER e traduzione di esso in schema logico.
- Creazione DB tramite script TSQL utilizzando SQL Server Management Studio (SSMS).
- Ideazione e implementazione dei metodi per il Download/Upload dei dati nel progetto PCTO.

Il Progetto PCTO:

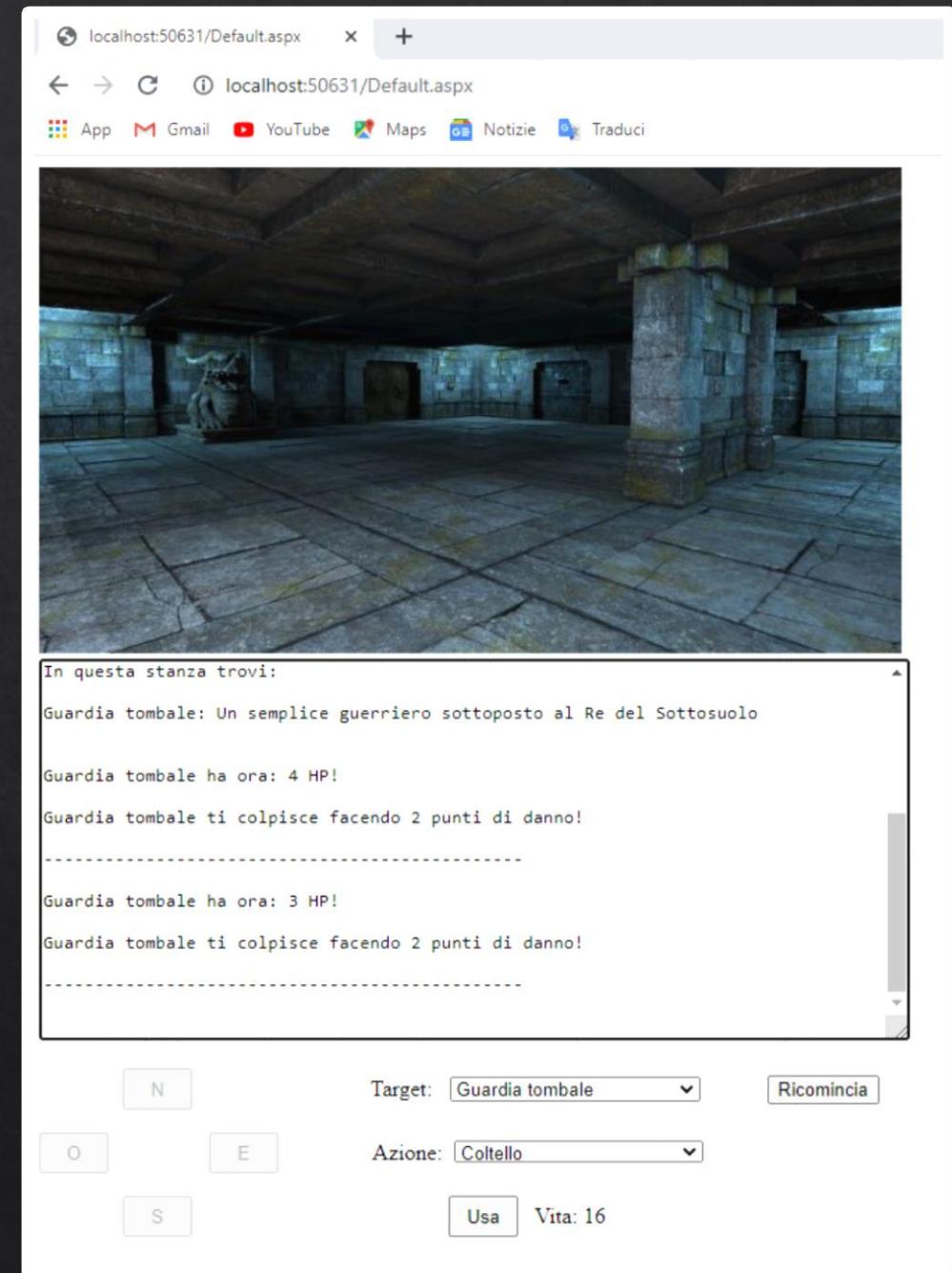
Progettato e sviluppato un' avventura testuale sotto forma di applicazione WEB ASP.NET, con l'ausilio dell'ambiente di sviluppo Visual Studio.

L'applicazione vede l'inclusione di features come:

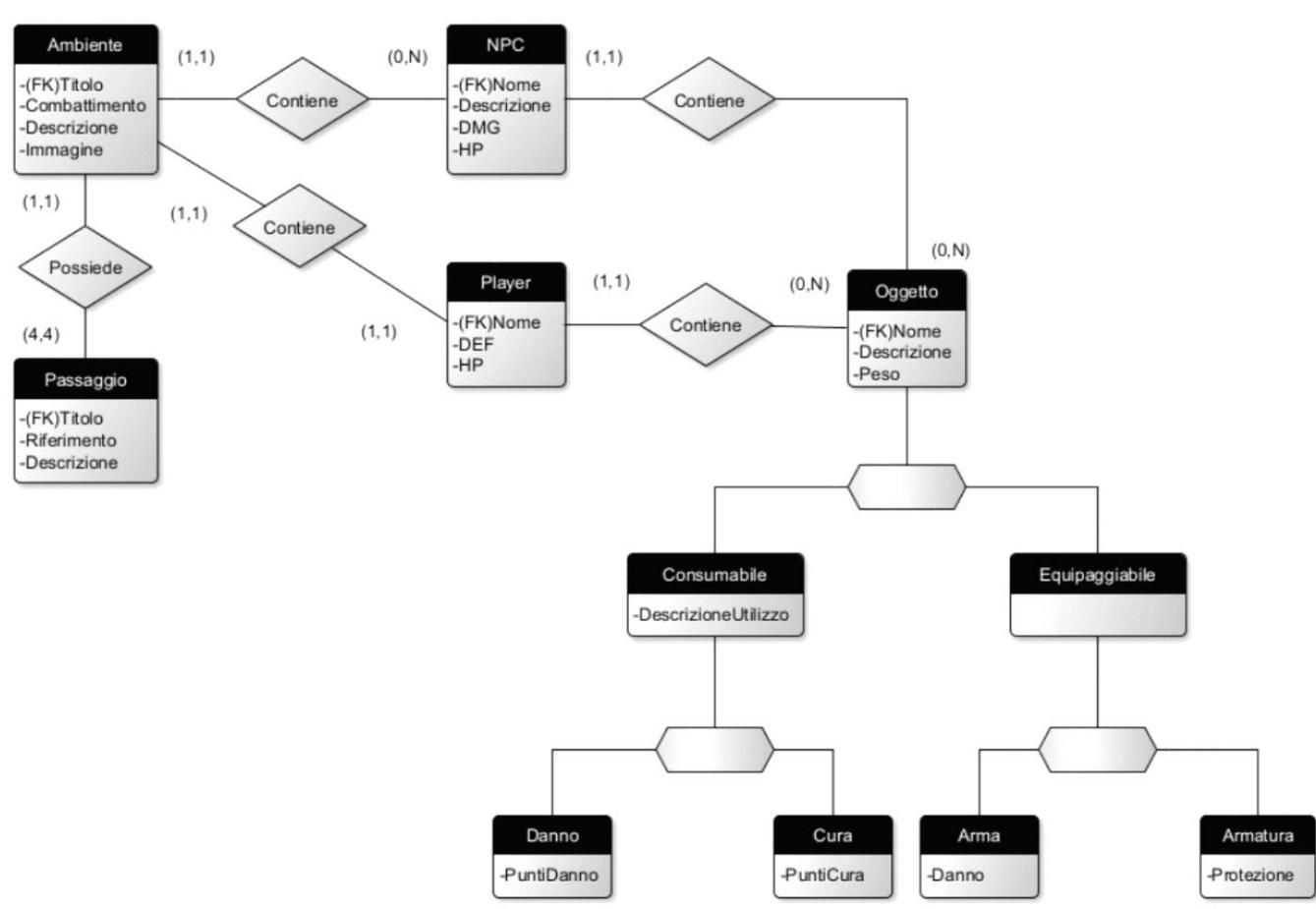
- Combattimento a turni
- Movimento tra ambienti prestabili
- Gestione vittoria/sconfitta con obiettivo l'uccisione di un 'BOSS'

I controlli del gioco prevedono 4 tasti per il movimento fra le stanze (N, S, O, E), una serie di comandi per gestire il combattimento (con la possibilità di scegliere l'arma e l'obiettivo dell'azione) e un bottone ricomincia per 'resettare' la partita.

Prevede infine uno sfondo e una listbox per la descrizione delle azioni e l'ambiente.



Schema ER



Lo schema ER prevede tutti i dati necessari al funzionamento dell'applicazione, che corrispondono alle proprietà della struttura di classi su cui si basa il gioco.

Ristrutturazione:

La ristrutturazione dell'ER ha toccato solo la gerarchia degli oggetti che è stata accorpata nell'entità padre 'Oggetto'.



Schema logico

Legenda:

- Grassetto + sottolineatura = Chiave primaria (PK)
- Corsivo + sottolineatura = Chiave esterna (FK)

Schema Logico:

Ambiente (**Titolo**, Combattimento, Descrizione, Immagine)

Passaggio (**Titolo**, Riferimento, Descrizione, NomeAmbiente)

NPC (**Nome**, Descrizione, DMG, HP, NomeAmbiente)

Player (**Nome**, DEF, HP, NomeAmbiente)

Oggetto (**Nome**, Descrizione, Peso, DescrizioneUtilizzo, PuntiDanno, PuntiCura, Durabilità, Danno, Protezione, Proprietario)

```

CREATE DATABASE LostDungeons
GO

USE LostDungeons
GO

CREATE TABLE Ambiente(
    Titolo NVARCHAR(50) PRIMARY KEY NOT NULL,
    Combattimento BIT,
    Descrizione NVARCHAR(100),
    Immagine NVARCHAR(50)
)

CREATE TABLE Passaggio(
    Nome NVARCHAR(50) PRIMARY KEY NOT NULL,
    Direzione INT CHECK(Direzione >= -1 AND Direzione <= 4),
    Riferimento INT,
    Descrizione NVARCHAR(100),
    Ambiente NVARCHAR(50) REFERENCES Ambiente(Titolo) ON DELETE CASCADE
)

CREATE TABLE NPC(
    Nome NVARCHAR(50) PRIMARY KEY NOT NULL,
    Descrizione NVARCHAR(100),
    DMG INT,
    HP INT,
    Ambiente NVARCHAR(50) REFERENCES Ambiente(Titolo) ON DELETE CASCADE
)

CREATE TABLE Player(
    Nome NVARCHAR(50) PRIMARY KEY NOT NULL,
    Descrizione NVARCHAR(100),
    HP INT,
    DEF INT,
    Ambiente NVARCHAR(50) REFERENCES Ambiente(Titolo) ON DELETE CASCADE
)

CREATE TABLE Oggetto(
    Nome NVARCHAR(50) PRIMARY KEY NOT NULL,
    Descrizione NVARCHAR(100),
    Peso INT,
    Tipo NVARCHAR(10),
    DescrizioneUtilizzo NVARCHAR(100),
    PuntiDanno INT,
    PuntiCura INT,
    Durabilità INT,
    Danno INT,
    Protezione INT,
    NPCOwner NVARCHAR(50) REFERENCES NPC(Nome) ON DELETE CASCADE,
    PlayerOwner NVARCHAR(50) REFERENCES Player(Nome) ON DELETE NO ACTION
)

```

Creazione database

Lo script qui riportato è responsabile della creazione del database su SSMS.

Segue la struttura designata negli schemi ER e logico e deve essere eseguito prima dell'avvio del programma ASP.NET poiché questo è totalmente dipendente dal DB per reperire i dati per il corretto funzionamento del gioco.

Per questo motivo, oltre alla creazione del DB, lo script si occupa anche di inserire i dati di partenza nelle tabelle in modo che siano immediatamente raggiungibili al primo avvio della soluzione.

Di seguito è riportato uno dei comandi INSERT che si occupano di questa operazione.

```

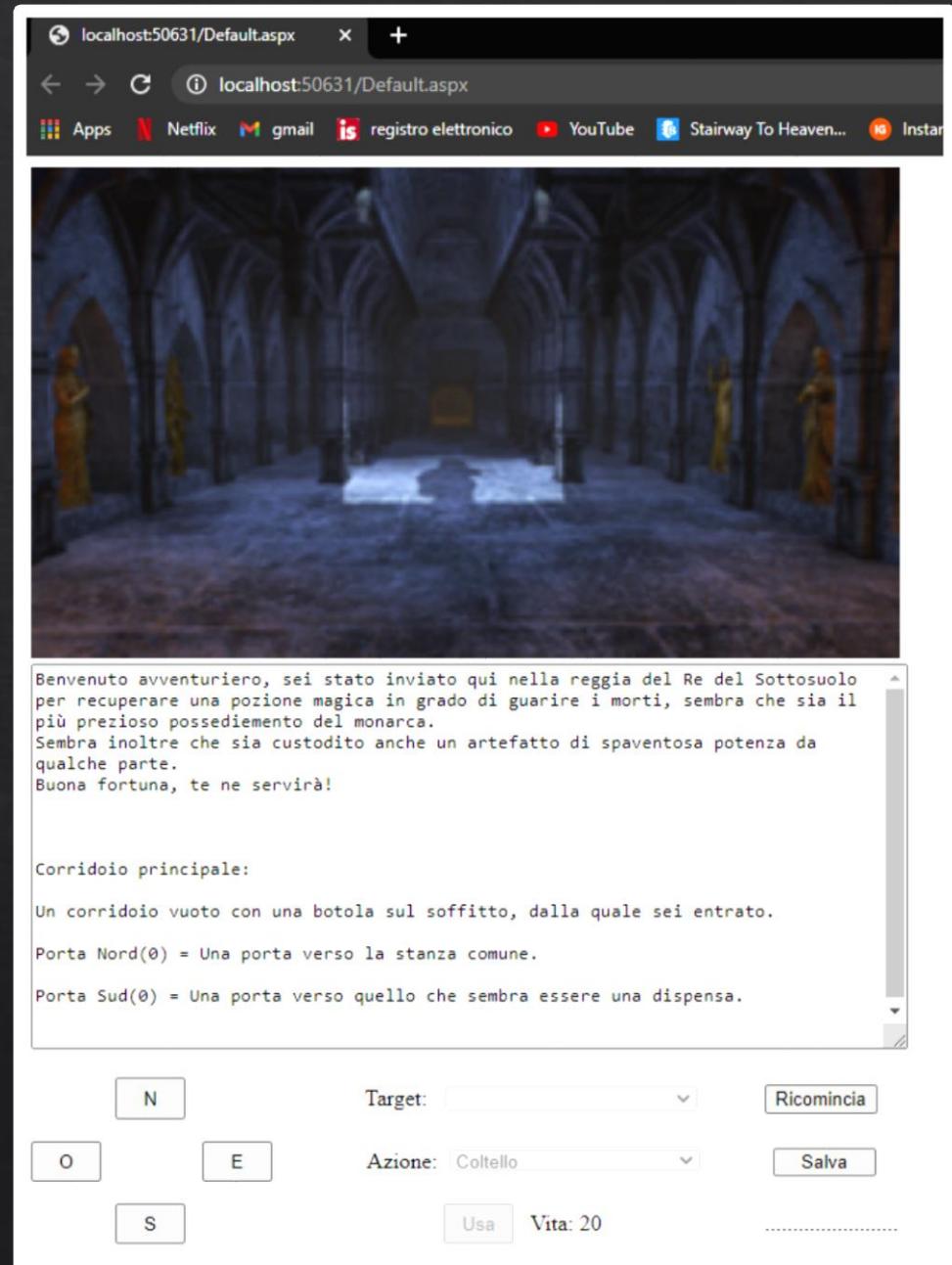
INSERT INTO Ambiente VALUES
    ('Corridoio principale', 0, 'Un corridoio vuoto con una botola sul soffitto, dalla quale sei entrato.', '~/Images/Room0.png'),
    ('Cucina', 0, 'La cucina personale del Re, non trovi nulla di interessante se non cibo e pentole sporche.', '~/Images/Room4.jpg'),
    ('Dispensa', 1, 'Una dispensa che contiene ben poco di valore.', '~/Images/Room1.png'),
    ('Forgia reale', 1, 'La forgia personale del re, molti dicono che armi leggendarie siano state forgeate in questo luogo.', '~/Images/Room6.jpg'),
    ('Stanza del trono', 1, 'La stanza più sfarzosa del complesso, e al centro di essa un trono dorato.', '~/Images/BossRoom.jpg'),
    ('Zona comune', 1, 'Una stanza comune, con diverse porte verso varie parti del complesso.', '~/Images/PreBossRoom.jpg')

```

La soluzione ASP.NET

Modifiche apportate alla soluzione di partenza:

- Aggiunto un bottone Salva per l'avvio del salvataggio da parte del player.
- Aggiornato il bottone Ricomincia per fare in modo che resetti anche il DB.
- Aggiunto metodo DownloadFromDB per gestire il download dei dati dal DB.
- Aggiunto metodo UploadData per gestire il caricamento dei dati nel DB.



Fase di avvio del programma

Durante il primo avvio del programma questo richiama il metodo DownloadFromDB per dare inizio alla lettura e il download dei dati.

Eseguita questa operazione vengono restituiti la mappa di gioco (sotto forma di array di Ambienti), i dati del player e la location di partenza del player. Questi dati vengono poi inseriti nei rispettivi Session per permetterne l'utilizzo al programma.

Il resto del metodo di apertura risulta identico a quello originale in quanto è servito solo cambiare l'input dei dati da inserimento manuale da codice a input attraverso il DB.

```
inserimento
protected void Page_Load(object sender, EventArgs e)
{
    //Primo ciclo del programma
    if (Session["firstTime"] == null)
    {
        //Creazione del player
        Player player = null;
        Ambiente location = null;

        //Introduzione
        txtout.Text = "Benvenuto avventuriero, sei stato inviato qui nella reggia del Re del Deserto. La tua missione è trovare il tesoro nascosto nel castello. Per farlo, devi superare diversi ostacoli e combattere mostri. Buona fortuna!";
        txtout.Text += "Iniziamo!";

        //Download dati dal DB
        Ambiente[] mappa = DownloadFromDB(ref player, ref location);

        //Label per HP
        lblVita.Text = "Vita: " + player.HP;

        //Salvataggio delle mappa nei vari session
        Session["Mappa"] = mappa;
        Session["Location"] = location;
        Session["Player"] = player;

        //Metodi di aggiornamento della componente grafica
        AggiungiContenuto();
        AggiornaAzioni();

        //Scrittura descrizione della stanza attuale
        txtout.Text += Session["Location"].ToString();

        //Preparazione movimento
        btnN.Enabled = true;
        btnS.Enabled = true;
        btnO.Enabled = true;
        btnE.Enabled = true;

        //Fine del primo ciclo
        Session["firstTime"] = false;

        //Variabile per riconoscere la fine del gioco
        Session["GameOver"] = false;
    }
}
```

Download dei dati

A occuparsi del download dei dati è il metodo DownloadFromDB.

Questo provvede a eseguire le query necessarie al database per restituire i dati che sono successivamente assegnati alle rispettive variabili.

Vista la necessità di eseguire più query contemporaneamente, ho trovato necessario includere l'opzione MultipleActiveResultSets nella stringa di connessione al server.

Di fianco un esempio in cui viene eseguito il download dell'inventario del giocatore dal DB.

```
//Estrazione inventario
//

//Variabile temporanea per inventario
List<EntitàOggetto> inventario = new List<EntitàOggetto> { };

//Creazione comando
sqlCmd = new SqlCommand(@"SELECT O.Nome, O.Descrizione, O.Peso, O.Tipo, O.DescrizioneUtilizzo, O.PuntiDanno, O.PuntiCura, O.Durabilità, O.Danno, O.Protezione
    FROM Player AS P INNER JOIN Oggetto AS O ON P.Nome = O.PlayerOwner", connection);

//Esecuzione comando
p = sqlCmd.ExecuteReader();

//Lettura oggetti nell'inventario
while (p.Read())
{
    //Tipo di oggetto
    string tipo = p[3].ToString().TrimEnd(null);

    //Inserimento oggetto nell'inventario
    switch (tipo)
    {
        case "Danno":
            inventario.Add(new Danno(p[0].ToString().TrimEnd(null), p[1].ToString().TrimEnd(null), (int)p[2], p[4].ToString().TrimEnd(null), (int)p[5]));
            break;
        case "Cura":
            inventario.Add(new Cura(p[0].ToString().TrimEnd(null), p[1].ToString().TrimEnd(null), (int)p[2], p[4].ToString().TrimEnd(null), (int)p[6]));
            break;
        case "Arma":
            inventario.Add(new Arma(p[0].ToString().TrimEnd(null), p[1].ToString().TrimEnd(null), (int)p[2], (int)p[7], (int)p[8]));
            break;
        case "Armatura":
            inventario.Add(new Armatura(p[0].ToString().TrimEnd(null), p[1].ToString().TrimEnd(null), (int)p[2], (int)p[7], (int)p[9]));
            break;
    }
}

//Fine lettura
p.Close();
```

Upload dei dati

A regolare invece l'upload dei dati sul DB è il metodo UploadData.

Questo metodo elimina tutti i dati presenti all'interno delle tabelle del DB e si occupa poi di inserire i nuovi dati tramite una serie di INSERT.

Il metodo è richiamato alla pressione del bottone SALVA che è attivo solo al di fuori dei combattimenti.

```
//  
//Upload dati player e location  
  
insertCmd = new SqlCommand(@"INSERT INTO Player VALUES (@Nome, @Descrizione, @HP, @DEF, @Ambiente)", connection);  
  
Player player = (Player)Session["Player"];  
Ambiente location = (Ambiente)Session["Location"];  
  
insertCmd.Parameters.AddWithValue("@Nome", player.Nome);  
insertCmd.Parameters.AddWithValue("@Descrizione", player.Descrizione);  
insertCmd.Parameters.AddWithValue("@HP", player.HP);  
insertCmd.Parameters.AddWithValue("@DEF", player.DEF);  
insertCmd.Parameters.AddWithValue("@Ambiente", location.Titolo);  
  
insertCmd.ExecuteNonQuery();
```

Parte di Sistemi e Reti

Obiettivo: Ideare una soluzione che ci permetta di poter erogare il servizio descritto nella parte di informatica.

Premessa: per avere un punto di vista più ‘realistico’ le considerazioni sono basate sulla realtà analizzata nel corso del PCTO di GPOI.

Questa realtà vedeva il videogioco rilasciato da parte di un team di due persone, senza un’infrastruttura affermata e un budget non elevato.

Il simbolo dato al prodotto nel corso del progetto di GPOI:



Fattori critici

Studiando la situazione descritta nella premessa ho stabilito alcuni punti da tenere conto nel formulare una soluzione per soddisfare l'obiettivo.

I fattori da considerare sono:

- Un budget limitato.
- Ambiente e infrastruttura locale non sviluppate.
- Mancanza di personale, interno all'azienda, specializzato nell'organizzazione e la gestione di sistemi server.
- Possibilità di espansioni future (legate al successo del prodotto).

Soluzione cloud



La soluzione che trovo più pertinente è l'istituzione di un servizio cloud.

Questo si appoggerebbe sull'infrastruttura di un provider, ovviando al problema della mancanza di infrastrutture e personale all'interno dell'azienda e permetterebbe di garantire delle prestazioni minime di rete e hardware.

Il server sarebbe virtualizzato, il che abbasserebbe ulteriormente i costi della soluzione, e senza grandi ripercussioni sulle prestazioni viste le scarse risorse richieste dal programma.

Questa soluzione permetterebbe un alto livello di scalabilità in quanto per aumentare le risorse nostra disposizione non basterebbe che richiedere un 'upgrade' al fornitore, con conseguente aumento dei costi e un aggiornamento dei SLA.

Il server si comporrebbe dunque di una server application che permetterebbe l'accesso a una risorsa SaaS (Software as a Service) che consiste nel videogioco sopra descritto.

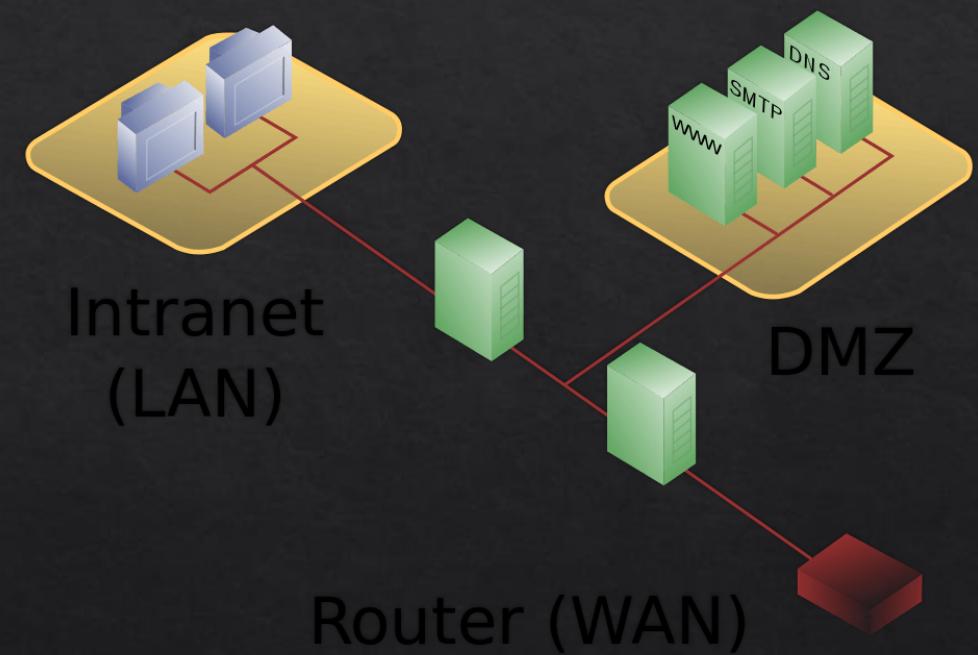
Soluzione alternativa locale

Una soluzione alternativa a quella cloud è la creazione di un Data Center di proprietà aziendale.

Questa soluzione risulta molto più costosa di quella cloud, data la necessità di fornirsi di personale competente nella gestione di sistemi informatici e dei componenti hardware per la realizzazione di esso, e trovo che sia da evitare almeno in una prima fase del progetto.

Penso che questa, però, sia una soluzione finale ottima, nonostante le maggiori responsabilità che comporta, in quanto permetterebbe un maggiore controllo nei confronti dei dati aziendali e degli utenti.

Questa soluzione dovrebbe prevedere la creazione di una DMZ aziendale tramite l'utilizzo di un router e due firewall per permettere l'erogazione sicura del servizio.



Fine

grazie per l'attenzione,
Sartini Matteo