Ministero dell'Istruzione, dell'Università e della Ricerca

Istituto di Istruzione Superiore "Benedetto Castelli"
Istituto Tecnico Settore Tecnologico, Scuola in Ospedale
Via Cantore, 9 25128 Brescia tel. 030/3700267 fax 030/395206 e-mail *segreteria@itiscastelli.it*
cod. fiscale 80048510178 - cod. unico fatturazione UFE3MI - cod. ipa istsc_bsis037004 - cod. mecc. BSIS037004
PEC: bsis037004@pec.istruzione.it - SITO: www.iiscastelli.gov.it

Elaborato Esame di Stato
A.S 2020/2021

Informatica - Sistemi e Reti

Docenti di riferimento:
Laura Di Natale
Giorgio Zicari

Studente:
Singh Karanbir

Maggio, 2021

Indice

1 Introduzione

1.1 Tema

"In un **fastfood** tutti i tavoli sono stati isolati in cubicoli di plexiglas per ridurre i rischi di contagio da Coronavirus, in modo che ogni **cliente** possa mangiare in modo separato dagli altri. Per ridurre il disagio ad ogni postazione si vuole inserire un **tablet**, che permetta ai clienti di guardare dei **video di intrattenimento**, che vengono proposti in base al **profilo** di ogni utente che si **registra**. Discutere come si potrebbe organizzare un tale servizio, sia dal punto di vista hardware che software, prendendo in considerazione le tematiche che si ritengono più rilevanti."

1.2 Considerazioni

Il tema proposto ci pone ad una realtà di riferimento attuale, in quanto il Coronavirus ha condizionato la vita di tutti e soprattutto i ristoranti. Sono vari i modi per affrontare il tema, pertanto sarà proposta una soluzione adatta al servizio richiesto.

Di seguito sono elencate le parti, che considero fondamentali, esaminate nella sezione "Soluzione":

- Fastfood
- Cliente
- Tablet
- Profilo
- Sede centrale

La soluzione proposta dal sottoscritto è stata ragionata in base alle conoscenze acquisite durante l'anno scolastico e messe a punto in modo da connettere reciprocamente tutti gli argomenti affrontati ed opportuni per organizzare il servizio.

1.3 Premessa

Tengo a precisare che il sottoscritto ha cercato di descrivere la soluzione in modo da non entrare troppo nel dettaglio, in quanto lo scopo di questa relazione è quello di spiegare in generale i vari elementi principali ed esporre le ragioni per le quali sono state prese determinate scelte.

2 Soluzione

In questa sezione, è riportata la soluzione al servizio richiesto nella traccia dell'elaborato.

Ho deciso di creare a bella posta un marchio per l'applicativo e per i fastfood.

Figura 1: Logo della catena di fastfood

2.1 Analisi punti fondamentali

2.1.1 Fastfood

Si tratta del nocciolo dell'argomento da trattare.

La traccia riporta "in **un** fastfood", facendo pensare che è richiesto di organizzare il servizio per un unico fastfood.

Ho deciso di sofisticare la consegna e di trattare una catena di fastfood.

Ogni fastfood:

1. sarà connesso ad **Internet**, la rete **pubblica** (e quindi ***non sicura***) per eccellenza;
2. è suddiviso in due reti private:
   - una per i **dipendenti**;
   - una per i **clienti**.

2.1.2 Cliente

È il consumatore che ha la possibilità di utilizzare il tablet per accedere all'applicativo messo a disposizione dal fastfood stesso.
Bisogna prevedere le diverse tipologie di cliente che possono usufruire del tablet:

- clienti comuni;
- clienti di età avanzata (l'applicativo deve essere **intuitivo** e non complicato da utilizzare);
- clienti stranieri (deve esserci un modo per tradurre l'applicativo in **lingua inglese**).

2.1.3 Tablet

È il dispositivo che fornisce il servizio di **streaming** di video in base al profilo dell'utente. Di conseguenza, l'applicativo in esecuzione sul tablet deve provvedere:

1. una pagina di **registrazione**;
2. una pagina di **login**;
3. una pagina che mostra le categorie;
4. una pagina che elenca i video della categoria selezionata;
5. (extra):
   - una lista dei menù del fastfood;
   - una lista dei prodotti del fastfood.

2.1.4 Profilo

Ogni utente, per vedere i video di intrattenimento, può creare un profilo fornendo: **email** e **password**. Una volta effettuata la registrazione, viene inviata un'email di conferma nella casella postale associata all'email fornita dal cliente.
Confermata la registrazione, il cliente può effettuare il login e visualizzare i video di intrattenimento.

2.1.5 Sede centrale

Tutti i fastfood hanno come riferimento un'unica **sede centrale**, che gestisce il software applicativo installato sui tablet.
La sede suddivisa in diversi reparti, quali: l'**amministrazione**, **la zona tecnici**, **una segreteria** e una zona dedicata per **le conferenze** o per **gli ospiti**.

Detto ciò, si può partire con la descrizione più tecnica della soluzione.

2.2 Schema realtà di riferimento

È stato riportato lo schema che rappresenta in generale la realtà di riferimento.

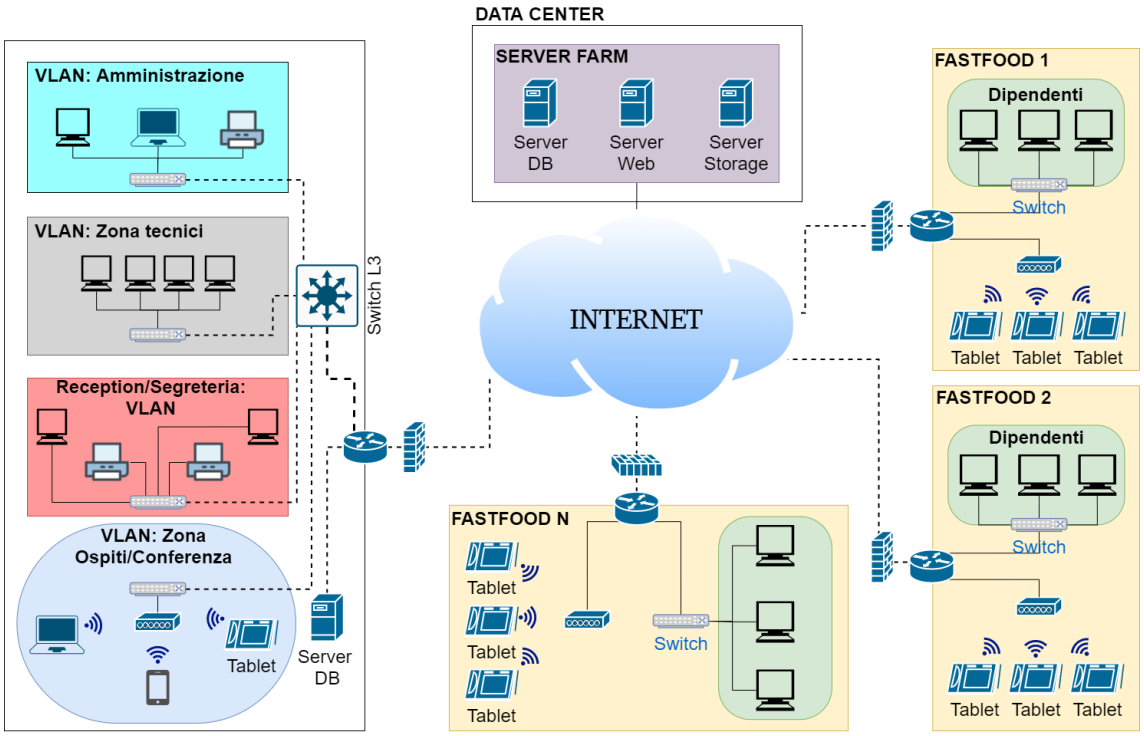

Figura 2: Schema

Si può notare subito la presenza di una catena di diversi fastfood.

Sono presenti la **sede centrale** suddivisa in reparti, precendentemente elencati, e una **server farm**, in cui sono situati tutti i server che si dedicano al mantenimento del software in esecuzione.

2.2.1 Analisi

La **sede centrale** è suddivisa in 4 **VLAN (*Virtual Local Area Network*)**:

- Amministrazione - 192.168.1.0/24 (classe C)
- Zona tecnici - 192.168.2.0/24 (classe C)
- Reception/Segreteria - 192.168.3.0/24 (classe C)
- Zona Ospiti/Conferenza - 192.168.4.0/24 (classe C)

Ogni VLAN viene gestita da uno **switch L3**, un dispositivo che, oltre ad operare a livello di **collegamento dati** (gestione degli indirizzi **MAC**), lavora a livello di **rete**, quindi gestisce gli indirizzi **IP**.
Per questo motivo, è stato utilizzato il **DHCP *(Dynamic Host Configuration Protocol)***, un protocollo per assegnare l'indirizzi IP, il Gateway ed l'indirizzo del server **DNS *(Domain Server Name)*** in maniera dinamica ai vari client della sede.

Ho deciso di suddividere la sede centrale in reti virtuali, in modo tale da prevedere un'eventuale divisione dei singoli reparti in più zone diverse (per esempio, a seguito di un'espansione della sede centrale).

Inoltre si può scorgere un server database, con indirizzo IP 192.168.100.2/30 (subnet, con massimo 2 host disponibili perché è supposto che ci sia solo il server DB in questa rete), utile alla sede per memorizzare le informazioni e i dati.

Per quanto riguarda i fastfood, si possono osservare le due reti locali:

- quella dedicata ai tablet - subnet di rete di classe C variabile (es 192.168.1.0/25)
- quella dedicata ai dipendenti - subnet di rete di classe C variabile (es. 192.168.1.128/26)

Gli schermi dei dipendenti sono collegati direttamente allo switch.

I tablet sono collegati ad un **access point** mediante una connessione wireless che, ha come **SSID**, il nome del fastfood in cui si trova: per esempio nel FASTFOOD 2 l'SSID sarà FastFood2, ecc. La **password** è fornita dalla sede centrale.

Tengo a precisare che, per i fastfood, sono state utilizzate delle subnet variabili in base alla grandezza del fastfood stesso: maggiore è il numero di clienti e dipendenti, maggiore dovrà essere il numero degli indirizzi IP disponibili.

La scelta degli indirizzi di rete di classe C (subnettate o meno) è stata data dal fatto che, sono comunemente quelli più adoperati nelle reti private. Di seguito è rappresentata una tabella contenente gli indirizzi disponibili per le reti locali:

Blocco di indirizzi	Intervallo indirizzi	Numero di indirizzi disponibili	Campo di utilizzo
10.0.0.0/8 (255.0.0.0)	10.0.0.0 10.255.255.255	16.777.216	Rete privata
172.16.0.0/12 (255.240.0.0)	172.16.0.0 172.31.255.255	1.048.576	Rete privata
192.168.0.0/16 (255.255.0.0)	192.168.0.0 192.168.255.255	65.536	Rete privata

Sarà il router, di ogni fastfood e della sede centrale, ad occuparsi nella conversione dell'indirizzo IP privato a quello pubblico, utilizzando il **PAT *(Port Address Translation)***, una particolare tecnica (estensione del ***Network Address Translation***) che permette di effettuare una "mappatura" tra più indirizzi IP di una rete privata e un singolo indirizzo IP pubblico.
Questo comporta ad avere un unico IP pubblico per ogni dispositivo che comunica con l'esterno. Pertanto, ogni client verrà distinto in base ad una **porta** associata alla connessione (max. 64 000 connessioni gestibili contemporaneamente).

La parte che invece considero particolarmente interessante è il **data center**, in cui si trova la **server farm**, una serie di server dedicati alla memorizzazione dei dati e esecuzione dell'applicazione.

Come si può vedere dall'immagine, abbiamo 3 server:

- un **server storage**, per salvare tutti i file relativi al al software (immagini dei menù e dei prodotti del fastfood, i video di intrattenimento);
- un **server web**, per gestire le richieste di pagine web (in questo caso quelle dell'applicativo) da parte del client. In pratica, quando una persona accede a un browser Web e digita un URL, il browser invia una richiesta (**HTTP o HTTPS**) al server web per ottenere la risorsa collegata a quell'URL. Il server risponde restituendo il contenuto cercato o con un messaggio di errore se non esiste (ad esempio, "Errore 404 - pagina non trovata"). La differenza fra HTTP e HTTPS è che il secondo si tratta sempre di un **protocollo di trasporto a livello applicativo**, ma utilizza la **porta 443** e la comunicazione avviene all'interno di un canale criptato (tramite **crittografia asimmetrica**) dal **TLS** (***Transport Layer Security***) o dal suo predecessore, **SSL** (***Secure Sockets Layer***). In questo modo il sito web in questione è **autenticato**, protegge la **privacy (riservatezza o confidenzialità)**, mantiene l'integrità dei dati scambiati dalle parti comunicanti. Nella soluzione proposta, è stato optato per l'utilizzo dell'HTTPS.
- un **server database**, per salvare i dati inerenti all'applicativo (informazioni riguardanti il cliente, descrizione delle categorie di video, riferimenti ai file nello storage, ecc.)

Il fatto di noleggiare (**hosting**) dei server di un data center comporta ad avere una serie di vantaggi e svantaggi:

- **vantaggi:**
  - possibilità di non avere dei spazi dedicati ai server fisici all'interno del proprio edificio;
  - aggiornamenti e gestione della sicurezza affidata allo staff del supporto sistemistico della server farm o dell'azienda che rivende il servizio;
  - nessun costo di acquisto iniziale della macchina e di installazione e configurazione iniziale;
  - costo di affitto dei server noto a priori e distribuito in pagamenti mensili.
- **svantaggi:**
  - la delega del funzionamento allo staff della server farm richiede fiducia e competenza in caso di applicazioni critiche;
  - alto costo di affitto mensile dei server in caso di applicazioni di rete critiche;
  - impossibilità di ricevere fin da subito macchine personalizzate, se non con costi aggiuntivi.

Prima di procedere nella descrizione dei firewall e dei privilegi tra le VLAN della sede centrale, sono elencati alcuni fra i più grandi provider di servizi su cloud: **Alibaba Cloud, Amazon Web Services, Microsoft Azure, Google Cloud Platform, Oracle Cloud**.

2.2.2 Firewall

Per definire quali pacchetti possono viaggiare tra i fastfood, sede centrale e Internet sono stati installati dei **firewall** sugli apparati router, detti **firewall perimetrali**. Essi permettono di controllare il contenuto dei pacchetti in entrata e uscita sulle due interfacce di rete: una affacciata sulla **rete sicura** e l'altra connessa alla **rete insicura**, ovvero Internet.
I firewall regolano il traffico circolante sulle interfacce in base alle cosiddette **ACL *(Access Control List)***: si trattano di liste di regole che definiscono quali pacchetti possono passare e a quali viene negato il transito.

Elenco delle principali regole:

1. dipendenti dei fastfood
    - possono accedere ad Internet;
    - possono accedere al server web dell'applicativo;
    - non possono comunicare con i tablet;
2. tablet
    - possono accedere **solo** al server web dell'applicativo;
3. sede centrale
    - tutte le VLAN possono accedere ad Internet;
    - **solo** la zona tecnici accede ai server nel data center;
    - **solo** l'amministrazione può comunicare con i dipendenti dei fastfood.

Per quanto riguarda la sede centrale, ci sono delle regole interne per gestire gli accessi fra le VLAN:

- la zona ospiti non può comunicare con le altre VLAN, eccetto la segreteria;
- la segreteria può comunicare solo con l'amministrazione e la zona ospiti;
- la zona tecnici comunica solo con l'amministrazione e il server database interno;
- l'amministrazione ha accesso a tutto.

2.3 Progettazione del Database

È il processo di formulazione di un modello dettagliato del database.
Questo modello contiene tutte le scelte progettuali a livello logico e fisico, e i parametri fisici di memorizzazione necessari per la generazione del **DDL *(Data Definition Language)***, che può essere usato per l'implementazione del database. Un modello dei dati completamente specificato contiene i dettagli caratteristici di ogni singola entità. Per la manipolazione dei dati (*lettura, inserimento, modifica, eliminazione, ecc.*) del database si utilizza il **DML *(Data Manipulation Language)***.

2.3.1 Modello concettuale

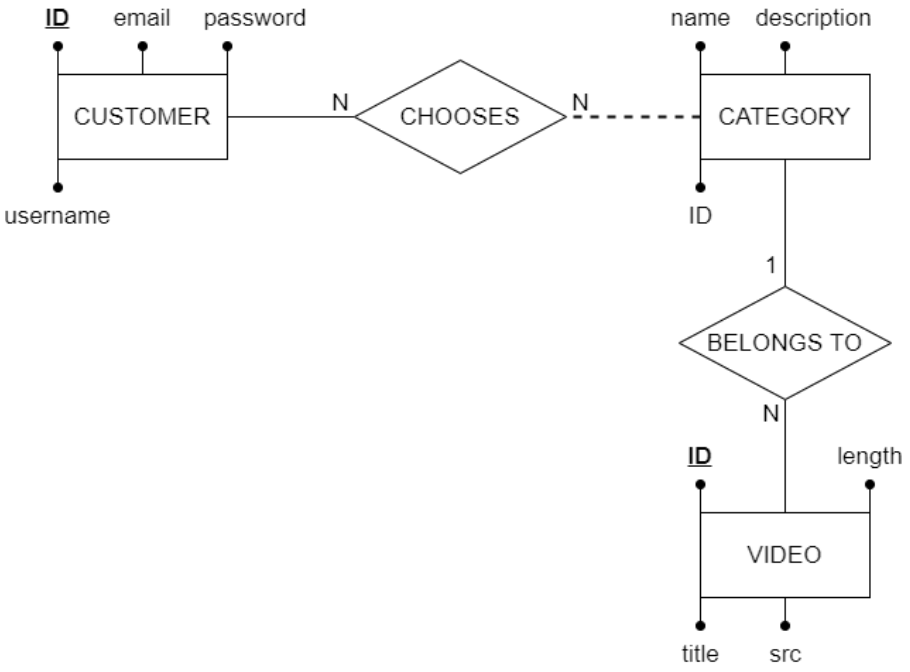

Figura 3: Modello concettuale minimale

L'immagine riportata di sopra, mostra le entità e relazioni indispensabili per progettare il database del software applicativo.

Piccola descrizione delle entità:

- **customer**: contiene le credenziali di accesso del cliente;
- **category**: contiene una serie di categorie dei video, salvate con il nome *(documentario, commedia, storico, azione, ecc.)* e un'eventuale descrizione *(es. "una serie di video che tratta vicende reali del passato", ecc.)*. Esiste una categoria particolare, *"Tutti gli altri"*, per raggruppare i video che non appartengono ad una categoria specifica;
- **video**: contiene una serie di video, ciascuno salvato con il titolo *(es. "chi era Malcom X?", "i segreti degli oceani", ecc.)*, la lunghezza (in minuti) e la categoria di appartenenza.

Ritengo importante fare una precisazione per quanto riguarda l'attributo **password** dell'entità **customer**: per questioni di **privacy**, quello che effettivamente viene salvato nel database non è la password in chiaro, ma una stringa ottenuta dopo che la password viene data come parametro in ingresso di una **funzione hash**. Verrà data un spiegazione più dettagliata nella sezione "Implementazione".

Essendo lo schema precedente troppo minimale, ho deciso di creare uno più completo aggiungendo alcune entità e relazioni che rendono il database più corposo.

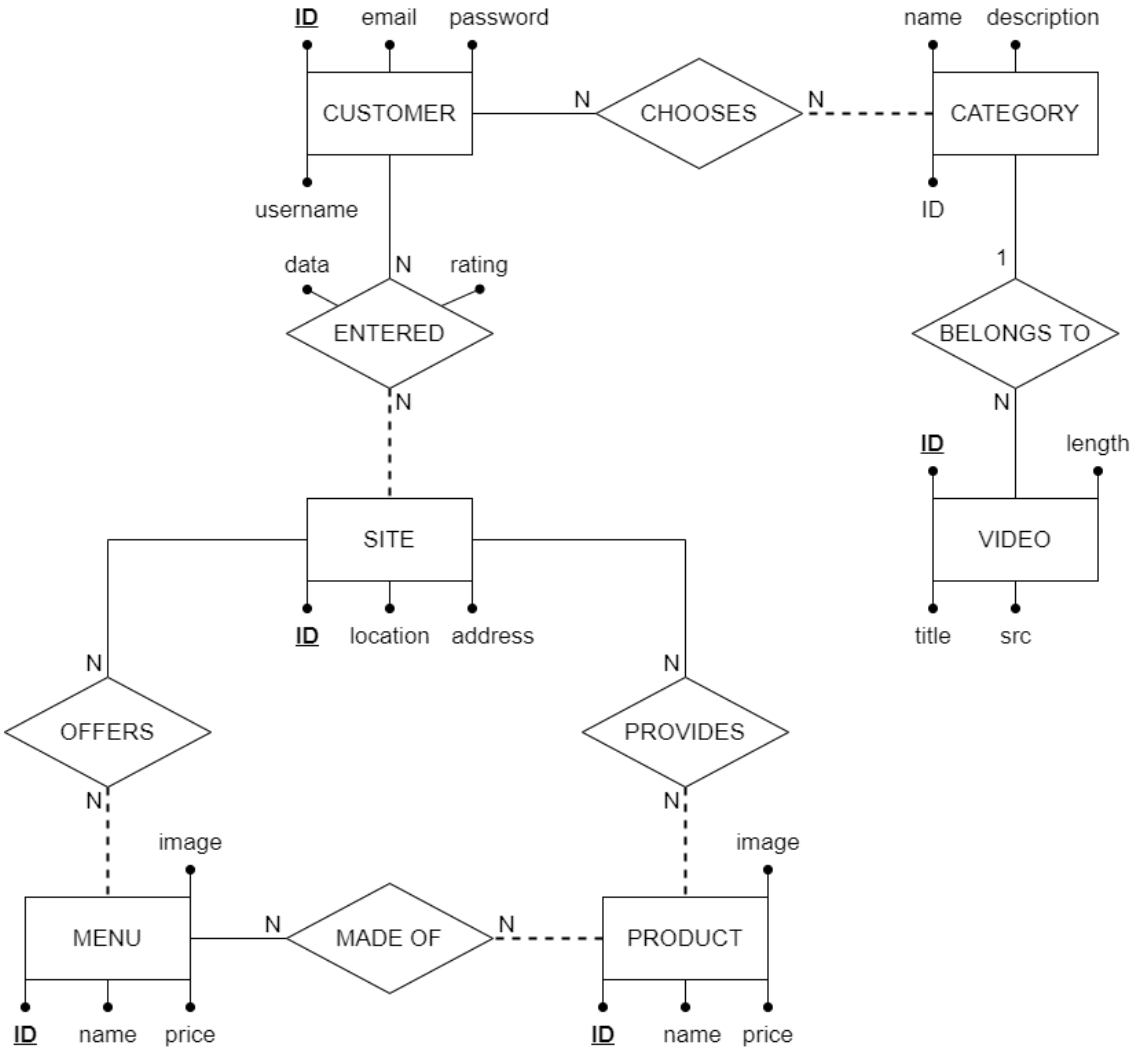

Figura 4: Modello concettuale completo

Le parti aggiuntive sono: **site, menu, product** come entità; **entered, offers, provides, made of** come relazioni.

Anche in questo caso è stata riportata un breve descrizione delle varie entità:

- **site**: contiene le informazioni inerenti alla sede fisica di un determinato fast-food, ovvero il luogo e l'indirizzo;

- **menu**: contiene una serie di menù, offerti da un determinato fastfood, ciascuno salvato con il nome *(es. Kansas, Vegetarian, Classico, ecc.)*, il prezzo e un'immagine[1]. I menù a loro volta sono costituti dai singoli prodotti;
- **product**: contiene una serie di prodotti, provvisti da un determinato fastfood, ciascuno salvato con il nome *(es. Hamburger, Patatine fritte, Bibita, ecc.)*, il prezzo, un'immagine[1] e la lista degli ingredienti che compongono quel prodotto.

2.3.2 Modello logico

Il **modello logico** discende dal modello concettuale e disegna un'architettura che tiene conto delle strutture proprie di quel particolare tipo di base di dati. Ciò significa che è possibile realizzare diversi varietà di database a partire da uno stesso modello concettuale.

L'immagine riportata di sotto mostra lo schema logico ottenuto dallo schema concettuale presentato in precedenza.

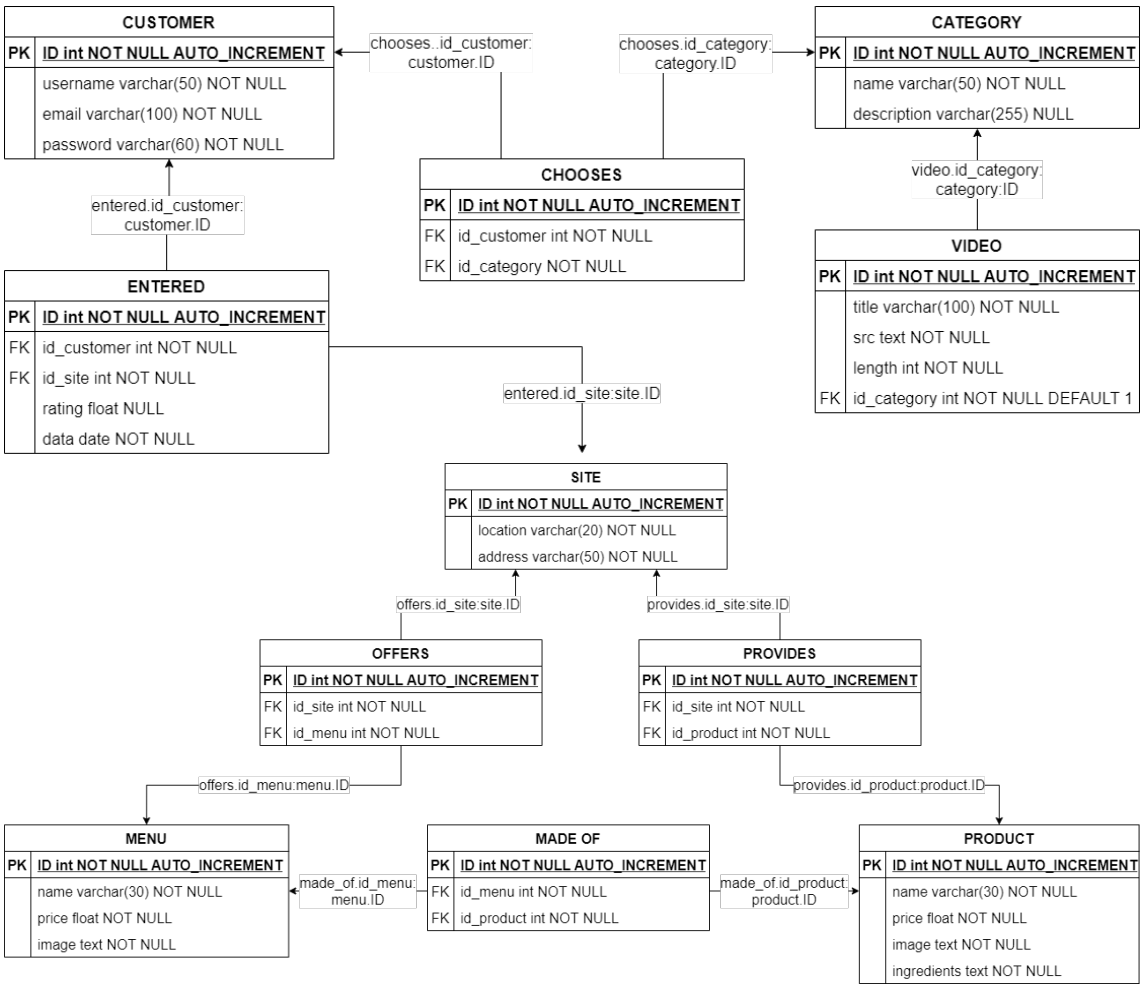

Figura 5: Modello logico

[1] in realtà è il **percorso** del file salvato nello storage

Il modello logico ci fornisce molte più informazioni rispetto a quello concettuale: per ogni **attributo** di un'entità vengono definiti il **tipo** ***(varchar, int, float, text, date, ecc.)*** e i **vincoli** ***(not null, null, primary key, auto increment, ecc.)***.

Ma particolare interesse assumono le **chiavi** e le **relazioni**:

- si possono notare le posizioni delle chiavi esterne, in base al tipo di relazione, osservando gli attributi delle tabelle oppure le frecce che indicano l'entità a cui è associata una determinata chiave esterna;
- si possono notare come le **relazioni N:N** diventino delle tabelle contenenti le chiavi esterne delle entità in relazione ed eventuali attributi extra.

2.3.3 Relazioni

È doveroso spiegare i motivi per i quali sono state create le relazioni mostrate attraverso i due modelli:

- **chooses**: la relazione ha una cardinalità **N:N** perché un cliente può scegliere molteplici categorie tra quelle presenti nel database, così come diverse categorie possono essere scelte da svariati clienti. C'è da precisare che possono esserci categorie che non sono state mai scelte da qualche cliente (si tratta di **parzialità** dal lato dell'entità categoria);
- **belongs to**: la relazione ha una cardinalità **N:1** perché un video può essere associato ad **una sola** categoria, mentre una categoria è composta da diversi video. Nel caso in cui un video non appartenga ad una categoria specifica, viene assegnato di **default** alla categoria *"Tutti gli altri"*;
- **entered**: la relazione ha una cardinalità **N:N** perché diversi clienti possono essere entrati (ed effettuato il login) in diverse sedi. Inoltre il cliente può dare una valutazione personale al fastfood visitato. Ho pensato fosse interessante il calcolo del numero dei clienti acceduti ad ogni fastfood a fine giornata come dato statistico. Fatto il conteggio, il contenuto della relazione viene cancellato;
- **offers**: la relazione ha una cardinalità **N:N** perché un determinato fastfood può offrire più di un menù, ma non tutti i menù sono offerti da almeno una sede (si prendano in considerazione menù nuovi o offerti solo da sedi particolari);
- **provides**: la relazione ha una cardinalità **N:N** perché un determinato fastfood può offrire diversi prodotti (come singoli), ma non tutti i prodotti sono offerti da almeno una sede (si prendano in considerazione prodotti nuovi o offerti solo da sedi particolari);
- **made of**: la relazione ha una cardinalità **N:N** perché i menù sono costituiti da almeno un prodotto, ma possono essere composti da prodotti differenti. Al contrario, non tutti i prodotti fanno parte di un menù, ma alcuni di essi sono offerti solo come singoli.

3 Implementazione

Per l'implementazione vera e propria sono state utilizzate determinate tecnologie e sono state indicate le ragioni per le scelte delle stesse.

3.1 Front-end

Questa parte rappresenta quella che è l'**interfaccia utente**, ovvero quel lato che si frappone tra il tablet e il cliente, che consente all'utente di interagire in modo reciproco.
L'obiettivo è quello di ottenere dai clienti i dati in input per il backend.

La **libreria open-source** utilizzata per creare l'applicativo è **React**.

Per quanto riguardo il design, ho preso in considerazione l'utilizzo di **Ant Design**, un'ottima e professionale libreria per la creazione di interfacce utente.

3.1.1 React

Si tratta di un **web framework** sviluppato da **Facebook** e serve per la creazione di applicazioni a **pagina singola**. Permette di suddividere le parti della pagina in **componenti React** e vengono renderizzati dopo esser stati convertiti dal **transcompiler Javascript Babel** nella versione compatibile con i browser odierni.

Ho deciso di adottare questa libreria perché ho preso un certa dimestichezza nel programmare in React, oltre ad essere molto utilizzata per i suoi vantaggi.
Per esempio: nel caso in cui la pagina debba essere aggiornata, vengono "ri-renderizzati" solo i componenti modificati della pagina. Questo è dovuto grazie all'utilizzo del **Virtual DOM *(Document Object Model)***.

3.2 Back-end

Per la parte di **back-end**, abbiamo **MariaDB** (fork derivato da **MySQL**) come database e degli script in linguaggio **PHP** per manipolare i dati contenuti nella base di dati e quelli all'interno delle richieste del client (tablet).
Il motivo di queste due scelte è per il semplice fatto che sono le uniche tecnologie analizzate durante l'anno scolastico: in alternativa, si poteva adottare un database diverso che avrebbe svolto lo stesso compito (**Microsoft SQL Server, MongoDB**) o utilizzare un linguaggio differente (**C#, Node.js, Java, Python, ecc.**).

3.2.1 Interazione con il database

Tramite PHP si effettuano le **interrogazioni al database** e i dati ottenuti dalle **query** vengono ritornati al lato client nel **formato JSON**, scelto perché più diffuso e, soprattutto, semplice da gestire con Javascript.

Una parte che considero di fondamentale importanza è la gestione degli errori e dei casi particolari: controllo dell'esito della registrazione e login, ecc.

3.2.2 Connessione a MariaDB tramite PHP

Ho deciso di riportare una piccola parte di codice per mostrare come avviene la connessione al server database tramite **PHP**.

```php
<?php
// Dice a livello dello script che gli errori verranno mostrati
// e che non verranno loggati
ini_set('display_errors', 1);
ini_set('log_errors', 0);

// Parametri di configurazione
$host = '<your_host>';
$db = 'fastfood';
$user = '<your_username>';
$pass = '<your_password>';
$charset = 'utf8';

// Data Source Name
$dsn = "mysql:host=$host;dbname=$db;charset=$charset";

// Attributi opzionali
$opt = [
    // Trasforma tutti gli errori SQL in eccezioni PHP
    PDO::ATTR_ERRMODE            => PDO::ERRMODE_EXCEPTION,

    // La query restituisce un array indicizzato
    // in base al nome della colonna
    // come restituito nel set di risultati
    PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC
];

// Oggetto che rappresenta la connessione al database
$pdo = new PDO($dsn, $user, $pass, $opt);

// Serve per creare una sessione
session_start();

// Questo e' il modo per leggere i dati della richiesta POST
$_POST = json_decode(file_get_contents("php://input"), true);
?>
```

La parte fondamentale è rappresentata dalla variabile **pdo** (istanza della classe **PDO**), la quale rappresenta l'effettiva connessione al database e fornisce varie funzioni che permettono di effettuare determinate azioni sul database.

Una particolare attenzione si deve porre su due punti:

1. **i parametri di configurazione**: si può notare come per le variabili **host, user e pass** non siano stati assegnati valori veri, ma solo un'etichetta che mostra il campo da inserire. Il motivo di questa scelta è data dal fatto che, escluso l'**host** (indirizzo IP del server database), si trattano di credenziali private dell'utente che vuole connettersi al database.

2. **gli attributi opzionali**: servono per settare degli attributi aggiuntivi alla connessione, in modo tale da gestire determinati eventi e situazioni. In questo caso, gli attributi che ho deciso di aggiungere sono: quello per gestire gli errori, i quali vengono trasformati in eccezioni, e quello per gestire il formato dei dati di ritorno del risultato di una query.

3.2.3 Gestione delle password

Sono state riportate le due funzioni che permettono di **criptare** la password (in fase di registrazione) e di verificare (in fase di login) se la password inserita dall'utente coincide con quella salvata nel database.

```
1 // Hash della password
2 $hash = password_hash($password, PASSWORD_DEFAULT);
3
4 // Verifica della password
5 password_verify($password, $hash);
```

Per descrivere il meccanismo, è stato riportato un esempio:

- **fase di registrazione**: se la password fosse "pippo", quello che otteniamo come risultato della funzione *password_hash()* è:
  "$2y10fMQpP9.fOM15XOeBcJk8CecPsdaWWzFXovVXKXqRTy6P7VEqTnGCq".
  Questa stringa rappresenta l'hash della password, da salvare nel database;
- **fase di login**: se la password fosse ancora "pippo" e l'hash (ottenuto in qualche modo attraverso le query) è quello precedente, allora la funzione *password_verify()* ritorna come risultato **true**. Ovviamente se la password non è corretta, la funzione ritorna **false**.

Questa sottosezione serve solo per spiegare come vengono gestite le password.
Tuttavia, rispetto a quello che è stato mostrato, sono presenti vari controlli prima di procedere nella registrazione o nel login del cliente.

Ad esempio: in fase di registrazione, bisogna controllare se l'email fornita dal nuovo cliente è già presente nel database, ecc.

Detto ciò, considero il momento adatto per passare alle conclusioni di questo elaborato, riportando alcuni pareri e riflessioni personali.

4 Conclusione

Ricapitolando, l'obiettivo di questa relazione è quello di descrivere come si potrebbe organizzare un servizio, sia dal punto di vista hardware che software, in base a quello che è stato richiesto: i fastfood devono permettere ai clienti di usufruire dei tablet inseriti ad ogni postazione, in modo da consentire la visualizzazione dei video di intrattenimento.

È stata proposta una possibile soluzione che include: una **server farm** di un **data center** per la gestione dei dati e dell'applicativo, una **sede centrale** proprietaria dell'applicativo, i singoli fastfood con i tablet e i dispositivi dei dipendenti.
Si è cercato di mettere assieme tutte queste componenti in modo da raggiungere l'obiettivo e quindi realizzare il servizio.

Naturalmente, il sottoscritto è stato sottoposto ad alcune scelte abbastanza fondamentali che hanno influenzato per la maggior parte determinati aspetti. Ne sono elencate alcune:

- una decisione importante da prendere è apparsa quando dovevo scegliere se fosse migliore adottare un sistema basato su cloud oppure installare dei server nello stesso edificio della sede centrale.
- una seconda decisione presa è stata quella riguardo alla suddivisione della sede centrale. Dovevo scegliere in che modo separare i vari comparti, se adottare delle VLAN oppure effettuare il subnetting di un'unico indirizzo di rete.

4.1 Problemi riscontrati

Non ci sono stati particolari problemi nella realizzazione di questo servizio.
Le uniche incertezze si sono presentate durante la simulazione del sistema o nella progettazione del database: dimenticanza dei comandi per gestire i privilegi delle VLAN o per abilitare il PAT sui router, dubbi nella scelta delle tipologie delle relazioni in base alle entità collegate.

4.2 Altre soluzioni ed ipotesi aggiuntive

Si potevano ipotizzare delle soluzioni differenti, una tra le quali già descritte precedentemente, ovvero quello di installare i server nell'edificio della sede centrale.
Si poteva ampliare il database per andare a salvare ulteriori informazioni importanti: gestione del pagamento direttamente sul tablet e quindi tenere traccia di tutte le transazioni, ecc.
Era conveniente aggiungere dei **web services**, gestiti totalmente dalla sede centrale, in modo adottare anche un livello intermedio fra client e il server web che ospita l'applicazione. Ciò avrebbe permesso di creare un servizio flessibile, scalabile, come un **sistema distribuito**.

Tuttavia, considero concretizzato il raggiungimento dell'obiettivo di realizzare il servizio richiesto, trattando diversi argomenti affrontanti nell'arco dell'anno scolastico, con le dovute spiegazioni personali.