

# Descrizione del progetto

Si vuole realizzare un'applicazione web per una pizzeria. Verranno venduti diversi tipi di pizze con ingredienti reperibili a magazzino. Le pizze avranno un tipo, nome e un prezzo.

Il magazzino conterrà le quantità degli ingredienti e quindi la loro disponibilità.

Gli utenti potranno accedere all'applicazione (in via anonima) per consultare il catalogo pizze, visualizzare le pizze e per ricerca.

Gli utenti registrandosi (con nome e cognome, indirizzo, telefono, login e password) e quindi autenticandosi potranno effettuare ordini, visualizzare gli ordini ed eventualmente cancellarli se non ancora consegnati. Gli ordini avranno due stati pending (ancora cancellabili) e delivered, non più modificabili.

Gli ordini dell'utente, conterranno informazioni su tipo e numero di pizze, giorno di consegna, ora di consegna preferita, indirizzo di consegna.

L'applicazione inoltre prevede un accesso amministratore per la gestione degli utenti e degli ordini: modifica dei dati degli utenti e degli ordini.

Si possono individuare le seguenti entità che partecipano ai casi d'uso nella specifica del progetto:

- pizze
- ingredienti
- clienti
- ordini
- amministratori

Le relazioni individuate saranno :

- le pizze sono composte da ingredienti (espressi a DB come unità scalari per semplicità)
- gli ordini contengono lavorazioni

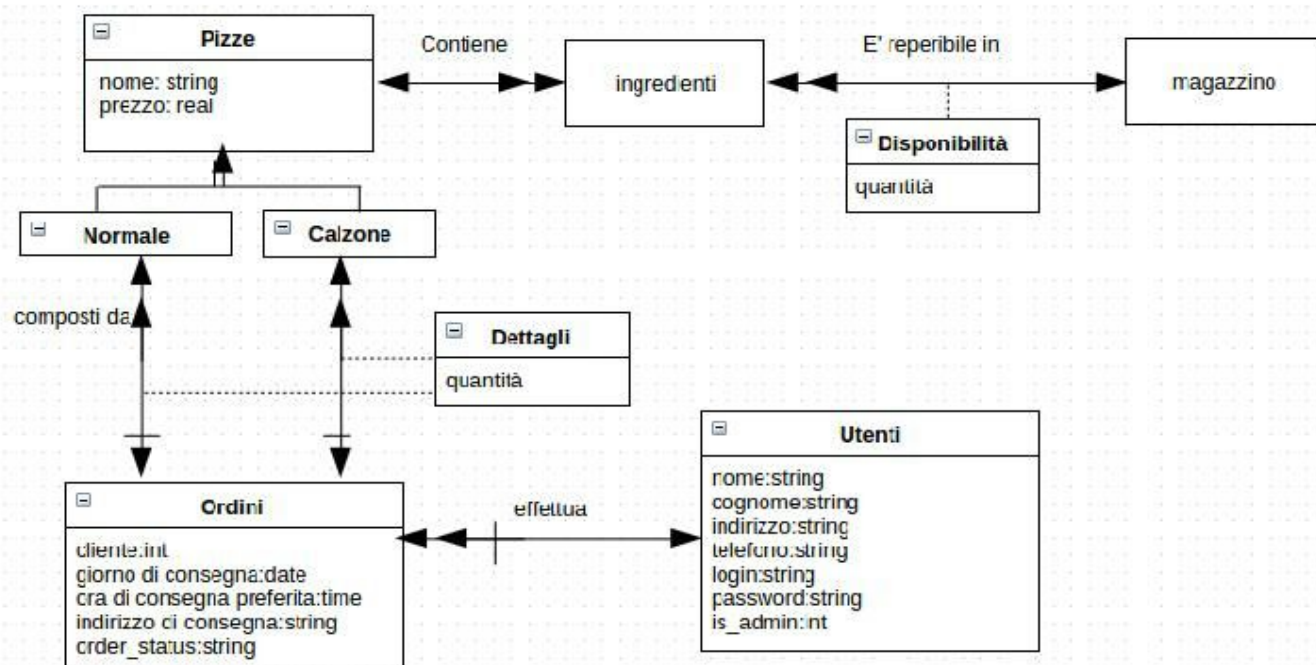
Note:

L'entità magazzino non viene rappresentata concettualmente ma non considerata nell'implementazione E.R. perché unica e assimilabile al concetto di ingredienti e quantità.

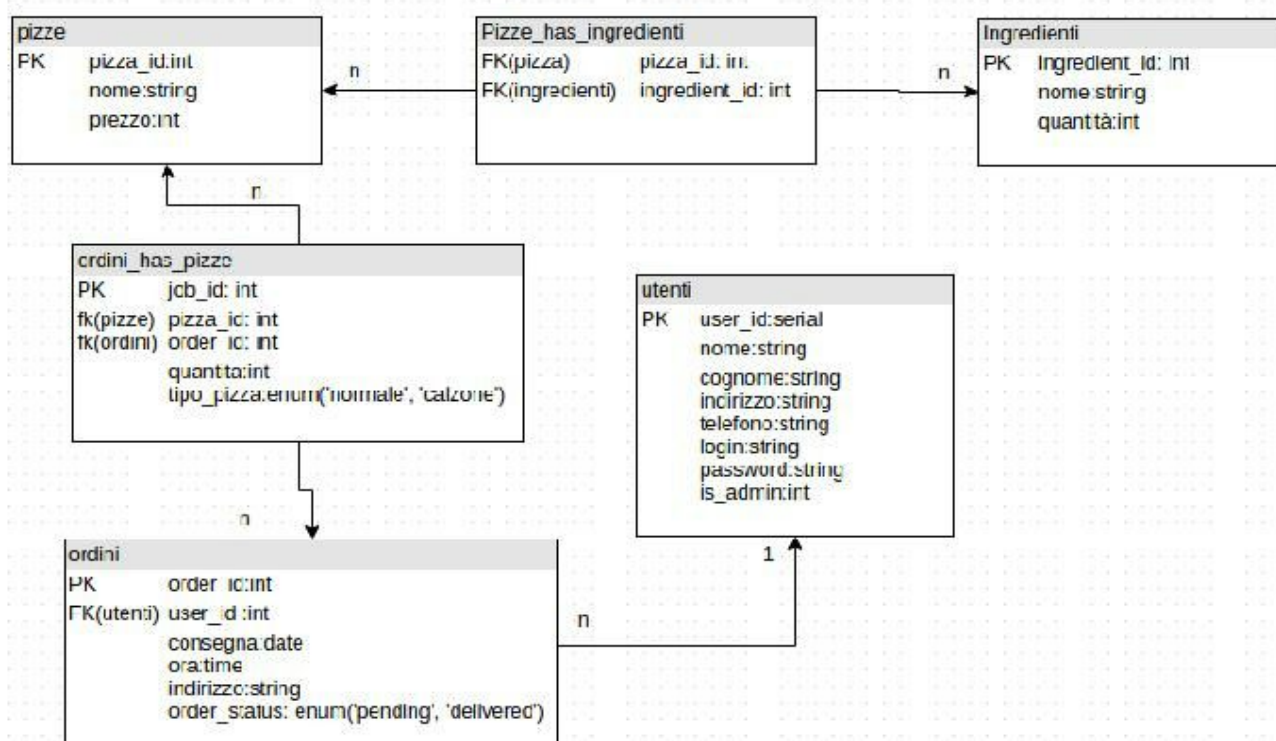
Per tipologie di pizze si assume che tutte le pizze possano essere realizzate con forme differenti: (pizza piana circolare, o pizza arrotolata tipo calzone)

Normale e calzone verranno quindi rappresentate concettualmente come sottoclassi di copertura di Pizze. Nel diagramma relazionale invece verranno rappresentate come "lavorazioni" di prodotti e come relazione n...n tra pizze e ordini.

## Schema Concettuale



## Schema relazionale

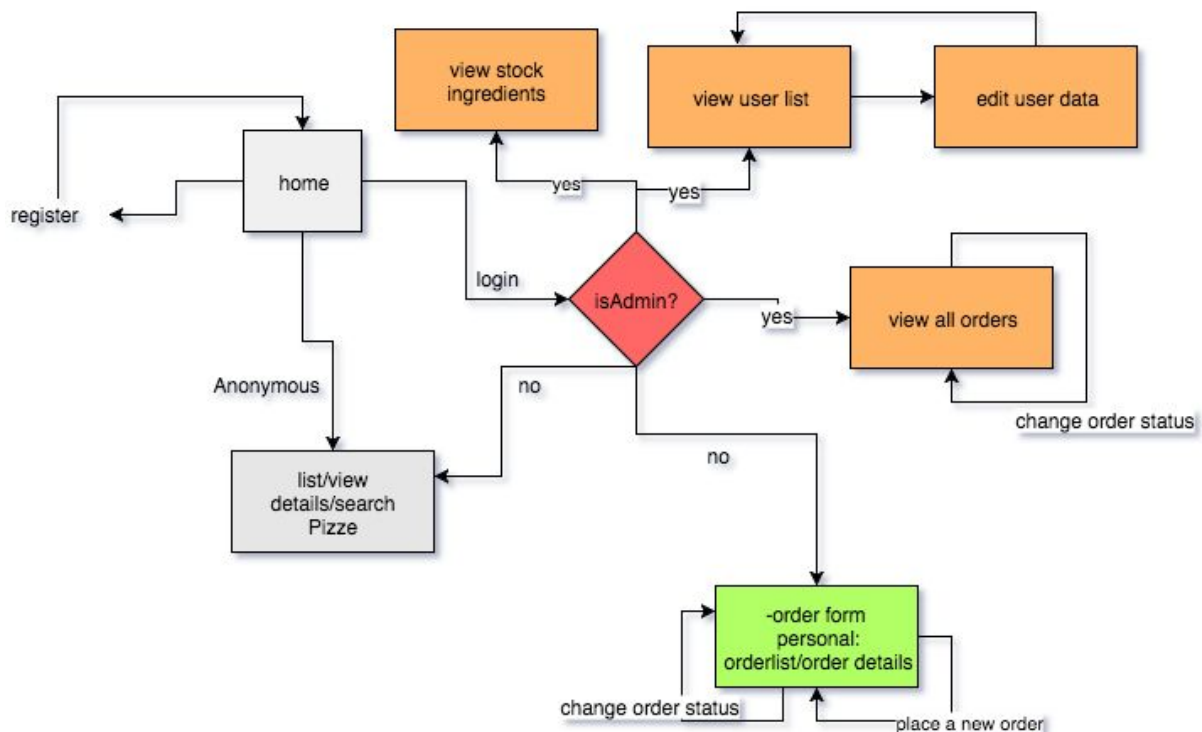


## Codice SQL

Il codice sql completo per la creazione del database si trova **nella cartella pizzeria/database/DDL\_pizzeria.sql**

- Sono state realizzate due function SQL **updateStorage()** , **updateStorageDel()** e impostati i trigger sulle lavorazioni(ordini\_has\_pizze) degli ordini.  
In questo modo vengono aggiornate le quantità degli ingredienti (a magazzino) all'inserimento e cancellazione degli ordini.
- E' stata realizzata anche una function **func\_order\_total()** per coadiuvare la query di rappresentazione degli ordini effettuati esibendo il totale dell'ordine.

## Diagramma e descrizione della struttura dell' applicazione web



L'applicazione web è contenuta nella cartella pizzeria

L'applicazione è stata installata su macchina virtuale vagrant con Apache2 PHP5.6 e Postgres 9.3

E' stato creato un role su postgres per l'autenticazione e i grant sul db dell' app. "pizzeria".

E' stato inizializzato un git repository per il tracciamento delle modifiche.

L'applicativo si compone delle seguenti cartelle principali:

- **classes:** contiene le classi per gestire il Model e quindi gli oggetti/entità del progetto e le connessioni con il DB. In particolare la classe DB usa PDO per stabilire la connessione (implementa pattern singleton per assicurarsi un'unica istanza in memoria alla connessione). Il file `require.inc.php` serve per includere tutte le classi (`require_once`) e renderle disponibili al front end.
- **config:** contiene il file `config.ini` con chiavi e valori di configurazione per il progetto, in questo caso solo i dati di connessione al DB.
- **database:** contiene gli script SQL, il DDL per creare/ricreare il DB "pulito". il DML per effettuare un pre-seeding di dati su cui lavorare ed eseguire una demo di utilizzo dell'app.
- **public:** è la cartella di pubblicazione a cui punta la config. virtualhost di apache (`ServerName pzz.local`)
- **Descrizione del frontend (public)**
  - è stato utilizzato npm/bower per integrare jquery e bootstrap al progetto
  - è stato aggiunto un file `htaccess` per abilitare il `session_autostart` di php, di default disabilitato su apache, per poter tracciare la sessione dopo l'autenticazione dell'utente
  - i files \*.php sotto public/ sono i files del front end che vengono chiamati durante la navigazione.
  - attraverso il login viene identificato l'utente (lo stesso viene usato per identificare anche utenti di amministrazione cui verrà settata una var di sessione apposita per distinguere gli accessi e i permessi alle pagine. La classe User infatti implementa dei metodi statici appositi per verificare che un utente possa navigare una specifica pagina (se autenticato, se amministratore) :adminPage,anonymousPage...
- **Sitemap per utente anonimo:**
  - `index.php` (Home) lista pizze (con dettaglio ingredienti) + ricerca
  - `login.php` pagina di autenticazione
  - `register.php` per registrarsi
- **Sitemap per utente loggato (standard)** (es. user:nicola, pwd:nicola):
  - `index.php` (Home) lista pizze (con dettaglio ingredienti) + ricerca
  - `ordini.php` (form invio nuovo ordine + lista ordini effettuati + possibilità di vedere i dettagli e cancellazione ordini, ma solo se in pending).
    - nel form degli ordini è stato usato jquery per comporre un oggetto json contenente tutte le lavorazioni (pizza, tipo, qta) che poi verrà elaborato lato php per l'inserimento delle relazioni nel model
- **Sitemap per utente loggato (administrator)** (es. user:admin, pwd:admin):
  - `admin_users.php` (home) visualizzazione utenti + edit

- admin\_user\_edit.php edit utente: modifica dati utente
- admin\_orders.php : visualizzazione ordini di tutti i clienti, vista dei dettagli, possibilità di disdire l'ordine (cancellare) o impostare lo stato come consegnato
- magazzino: lista ingredienti disponibili a magazzino con quantità