Università degli Studi di Ferrara

DIPARTIMENTO DI INGEGNERIA

Corso di Laurea in Ingegneria Informatica e dell'Automazione

Progetto di Ingegneria del Software Avanzata

Professor Alberti Marco
Professor Azzolini Damiano

Alessandro Bertelli

Anno Accademico 2021/22

Contents

Introduzione

Lo sviluppo di questo progetto è finalizzato al soddisfacimento dei requisiti richiesti per quanto concerne i corsi di Ingegneria del Software Avanzata e di Tecnologie per le Basi di Dati.
I requisiti specifici richiesti per Tecnologie per le basi di dati sono:

- realizzazione di un'applicazione che utilizza un database,
- utilizzo di linguaggio di programmazione a scelta,
- presentazione dello schema logico del database,
- il database deve contenere almeno 5 tabelle,
- l'applicazione deve fare operazioni di INSERT, UPDATE, DELETE e almeno un JOIN,
- il DBMS deve essere SQL Server, Oracle, DB2 o postgres (non si può utilizzare mySQL o Access).

L'applicazione da me sviluppata è una web app Flask che si interfaccia con un database postgres per gestire le prenotazioni di una pizzeria. Tramite essa infatti, un cliente potrà visualizzare il menu della pizzeria, effettuare le proprie prenotazioni, modificarle o eliminarle. Ho utilizzato psycopg2 per realizzare la connessione tra la web app Flask (sviluppata in python con l'utilizzo di html, css e javascript lato web) e il database postgres. Inoltre ho utilizzato DBeaver come strumento di amministrazione database.

Per quanto riguarda Ingegneria del Software Avanzata, il progetto ha lo scopo di verificare la capacità di condurre un progetto software utilizzando le tecniche e gli strumenti proposti nel corso, con conseguenti sviluppo e discussione del progetto.
Per quanto riguarda l'integrazione di questi requisiti nel mio progetto ho quindi utilizzato:

- **Git, GitHub, GitKraken**: per il controllo di versione,
- **conda**: package manager open source (utilizzato per la gestione delle dipendenze),
- **pytest-flask**: framework di test per la creazione di una suite di test automatizzati.

Descrizione del database

Analisi dei requisiti con descrizione del mini-mondo

La base di dati PIZZERIA tiene traccia di tutte le prenotazioni di una pizzeria per asporto, dei menu delle pizze e delle bevande, dei clienti e dei coupon (buoni sconto) messi a disposizione dalla pizzeria.

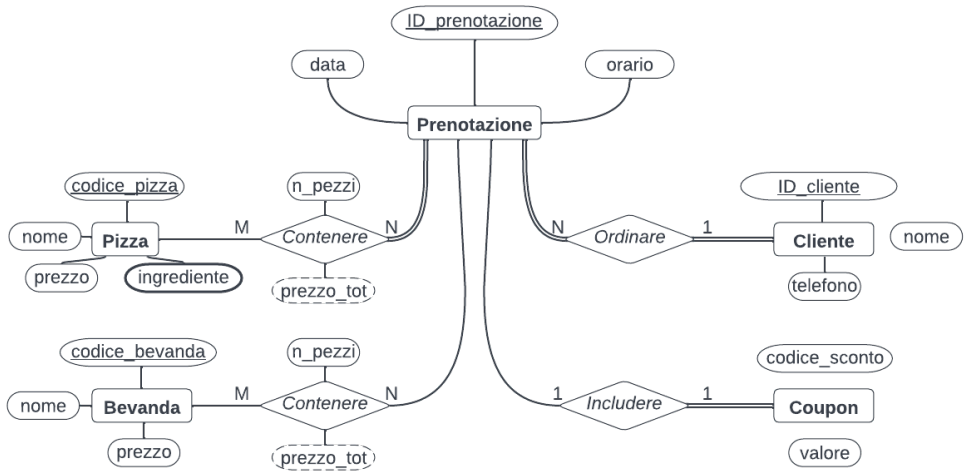
Ogni prenotazione ha un codice univoco, un cliente associato, un orario di riferimento e la lista dei codici dei prodotti (pizze e bevande) desiderati. I prodotti sono di due tipi: pizze o bevande.

Ogni pizza ha un codice univoco, un nome, (la lista di ingredienti) e un prezzo. Ogni bevanda ha un codice univoco, un nome e un prezzo. Ogni cliente è identificato da un ID cliente univoco che sarà associato al nome e, in modo facoltativo, al numero di telefono del cliente. Inoltre la pizzeria in particolari occasioni rilascia dei coupon a qualche cliente permettendogli di avere diversi tipi di sconto.

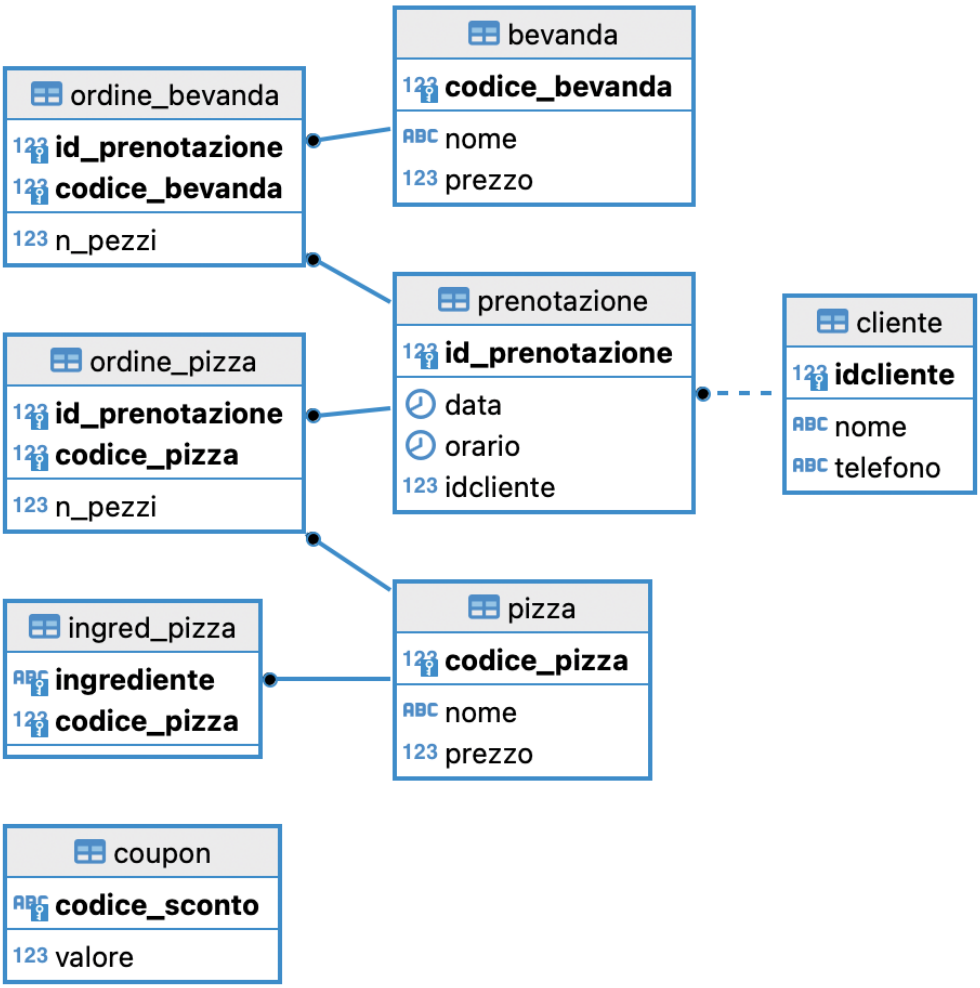
Ogni coupon ha un codice univoco identificativo e un valore in euro pari al valore dello sconto applicabile (chiaramente dopo essere stato utilizzato viene eliminato dal database, i coupon non sono cumulabili).

Progettazione dello schema ER/EER

Schema ER non normalizzato



Schema ER normalizzato - *autogenerato da DBeaver*



Schema Relazionale

Normalizzazione in Terza Forma Normale (3NF)

PRENOTAZIONE

ID_PRENOTAZIONE	DATA	ORARIO	IDCLIENTE

CLIENTE

IDCLIENTE	NOME	TELEFONO

COUPON

CODICE_SCONTO	VALORE

PIZZA

CODICE_PIZZA	NOME	PREZZO

ORDINE_PIZZA

ID_PRENOTAZIONE	**CODICE_PIZZA**	N_PEZZI

INGRED_PIZZA

INGREDIENTE	**CODICE_PIZZA**

BEVANDA

CODICE_BEVANDA	NOME	PREZZO

ORDINE_BEVANDA

ID_PRENOTAZIONE	**CODICE_BEVANDA**	N_PEZZI

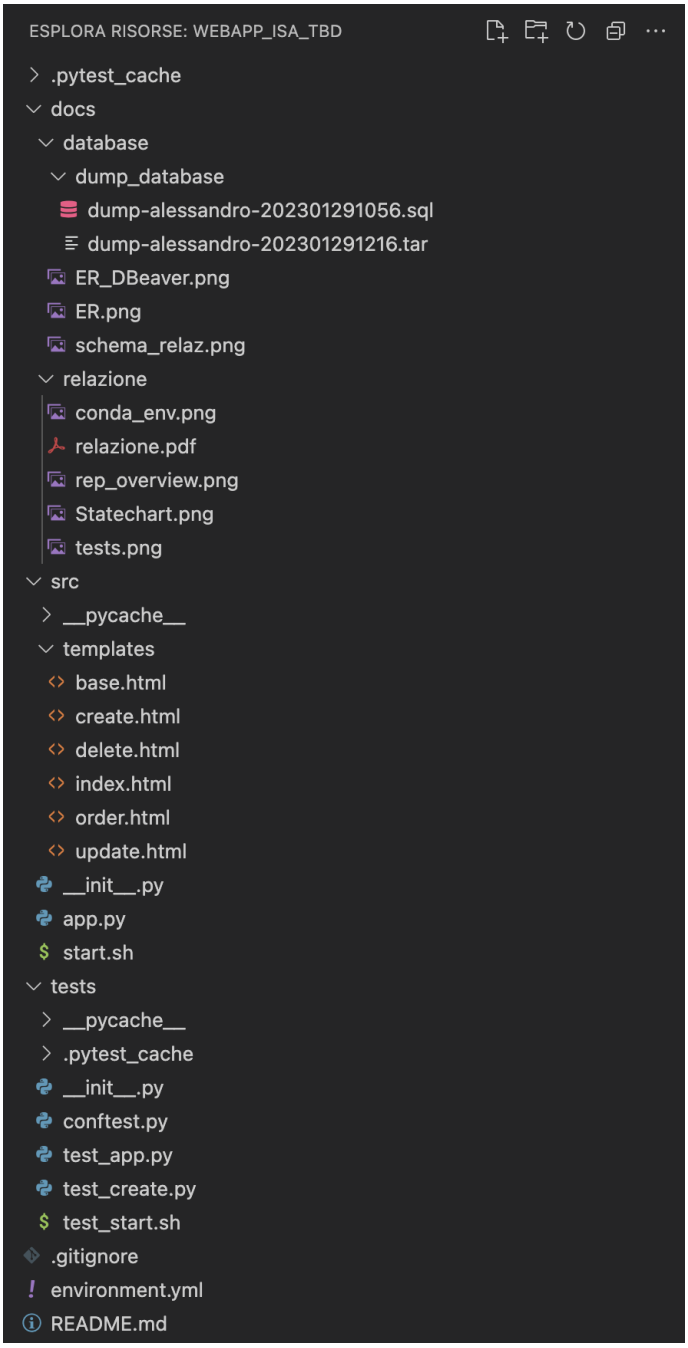
LEGENDA:

- **chiave primaria**
- chiave esterna

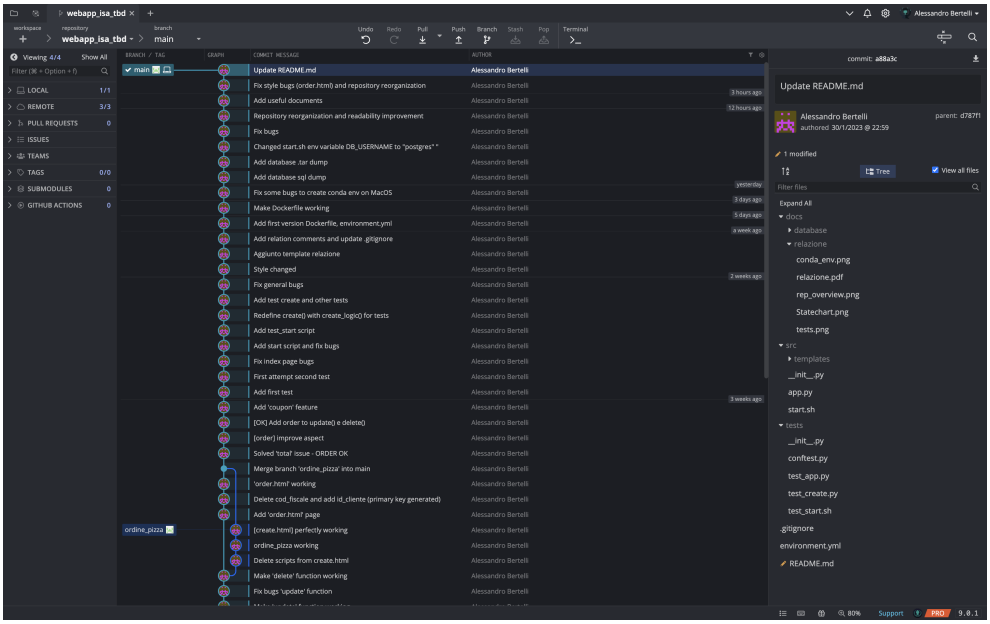
Requisiti di sistema

Controllo di versione: *Git/GitKraken*

Organizzazione repository



GUI GitKraken



Gestione delle dipendenze: *conda*

Per quanto riguarda la gestione delle dipendenze ho utilizzato *conda*, un package manager open source che consente di installare, eseguire e aggiornare rapidamente pacchetti e relative dipendenze tramite l'utilizzo di ambienti virtuali. Per quest'applicazione tutte le dipendenze sono racchiuse all'interno dell'ambiente virtuale conda *proj_env*, di cui riporto l'estrazione tramite il file *environment.yml*.

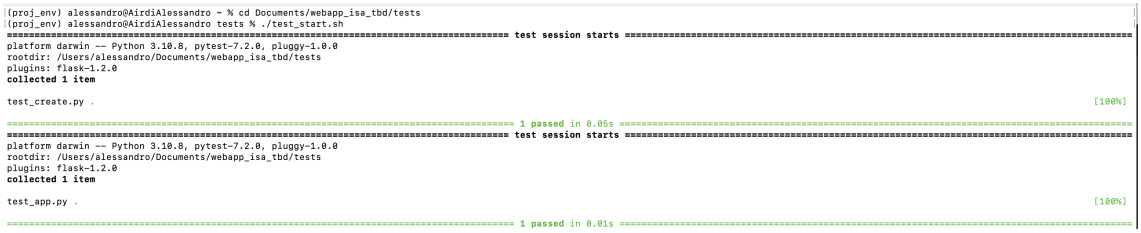
environment.yml

```yaml
name: proj_env
channels:
  - defaults
dependencies:
  - bzip2=1.0.8
  - ca-certificates=2022.10.11
  - certifi=2022.9.24
  - libffi=3.4.2
  - libuuid=1.41.5
  - ncurses=6.3
  - openssl=1.1.1s
  - pip=22.2.2
  - python=3.10.8
  - readline=8.2
  - setuptools=65.5.0
  - sqlite=3.40.0
  - tk=8.6.12
  - tzdata=2022g
  - wheel=0.37.1
  - xz=5.2.8
  - zlib=1.2.13
  - pip:
    - attrs==22.2.0
    - charset-normalizer==3.0.1
    - click==8.1.3
    - exceptiongroup==1.1.0
    - flask==2.2.2
    - flask-testing==0.8.1
    - idna==3.4
    - iniconfig==2.0.0
    - itsdangerous==2.1.2
    - jinja2==3.1.2
    - markupsafe==2.1.1
    - packaging==23.0
    - pluggy==1.0.0
    - psycopg2-binary==2.9.5
    - pytest==7.2.0
    - pytest-flask==1.2.0
    - requests==2.28.2
    - tomli==2.0.1
    - urllib3==1.26.14
    - werkzeug==2.2.2
prefix: /home/alessandro/anaconda3/envs/proj_env
```

Suite di test automatizzati: *pytest-flask*

Per sviluppare una suite di test automatizzati ho utilizzato la libreria *pytest-flask*. Come si può notare dalla struttura della repository, tutti i file di test sono, come indicato dalle best practice, organizzati nella cartella *tests* in questo modo:

- **conftest.py**: configurazione iniziale che permette, tramite database URI, la connessione al database per l'esecuzione dei test,
- **test_app.py**: test che verifica la corretta connessione al database,
- **test_create.py**: test che verifica il corretto funzionamento della funzione create(). Ho impostato dei dati di test e il test client per testare la richiesta HTTP POST al fine di controllare che le prenotazioni vengano inserite correttamente all'interno del database,
- **test_start.sh**: script bash per avviare automaticamente i test inserendo le variabili globali d'ambiente per connettersi al database.



Statechart

Di seguito è riportato lo statechart relativo al funzionamento della web app **PIZZApp**.