

Università degli Studi di Catania

Dipartimento di ingegneria informatica

Corso di Laurea in ingegneria informatica, secondo livello

*Irene Baldacchino*

*Marco D’Alessandro*

Just It

progetto di Advanced programming languages

Anno Accademico 2020 – 2021

Indice

[Capitolo 1 – Introduzione 3](#_Toc89166601)

[Capitolo 2 – Client 4](#_Toc89166602)

[2.1 – Ristoratore 4](#_Toc89166603)

[2.2 – Cliente 4](#_Toc89166604)

[2.3 – Alcune scelte implementative 5](#_Toc89166605)

[Capitolo 3 – Server 5](#_Toc89166606)

[3.1 – Gestione Database 6](#_Toc89166607)

[Capitolo 4 – Script R 6](#_Toc89166608)

[Capitolo 5 – Conclusioni 6](#_Toc89166609)

# Introduzione

**JUST IT** è un software implementato con lo scopo di gestire un meccanismo di prenotazione per consegne a domicilio, simile ad altri software già esistenti come Just It, Glovo o Deliveroo.

Dopo l’avvenuta registrazione o login, in base alla tipologia di utente (ristoratore o cliente), saranno visibili due diverse interfacce con funzionalità totalmente differenti al fine di simulare un vero e proprio meccanismo di scambio di ordini.

L’applicativo è stato suddiviso in quattro parti:

1. **Client**, implementato mediante l’utilizzo di C# poiché permette una programmazione ad eventi, essenziale per l’interazione dell’utente con l’interfaccia;
2. **Server**, implementato mediante l’utilizzo di Python insieme a Flask, framework web;
3. **Statistiche,** effettuate mediante l’utilizzo di R, nato per lo sviluppo di analisi statistica dei dati;
4. **Database** MySQL per rendere permanenti i dati inseriti.

Ogni capitolo sarà dedicato ad ogni singola parte implementativa del software, evidenziando alcune scelte implementative effettuate durante la sua realizzazione.

# Client

Il client, quindi la parte dedicata all’interazione con l’utente, è stata implementata utilizzando come linguaggio di programmazione C#.

Questo ha permesso la creazione di un’interfaccia minimale per la generazione di eventi che eseguono chiamate REST al server.

Inizialmente, l’utente visualizzerà un’interfaccia comune sia al ristoratore che al cliente, dove sarà possibile effettuare il login o la registrazione.

Quindi la schermata principale che visualizzerà è la seguente:

//IMMAGINE LOGIN E REGISTRAZIONE.

In questo caso inserendo le proprie credenziali sarà possibile accedere direttamente alla home associata al ristoratore o a quella del cliente.

## – Ristoratore

Nel caso in cui l’utente risulti essere un ristoratore questo visualizzerà la seguente homepage.

//IMMAGINE HOME RISTORANTE

Da questa avrà la possibilità di:

1. **aprire/chiudere** il ristorante (in modo tale che il cliente possa visualizzarlo ed abbia la possibilità di poter effettuare un ordine);
2. **inserire, visualizzare o cancellare (gestire)** i menù con i relativi piatti;
3. **visualizzare** gli ordini effettuati dai clienti;
4. **cambiare lo stato dell’ordine in confermato o spedito;**
5. **visualizzare il feedback** rilasciato dagli utenti.

## – Cliente

Nel caso in cui l’utente risulti essere un cliente questo visualizzerà la seguente homepage.

//IMMAGINE HOME cliente

Da questa avrà la possibilità di:

* 1. **Visualizzare i ristoranti aperti;**
  2. **Visualizzare il menù** del ristorante scelto**;**
  3. **Effettuare un ordine;**
  4. **Visualizzare lo stato degli ordini attuali e precedenti;**
  5. **Rilasciare un feedback al ristorante** dopo aver ricevuto un ordine.

## – Alcune scelte implementative

# Server

La parte server permette di gestire la registrazione, il login e tutte le funzioni relative ai clienti e ai ristoratori. E’ stato implementato in Python tramite l’ausilio del framework web “Flask”, il quale permette al programmatore di gestire ciascuna funzione tramite l’utilizzo di opportune route, ciascuna delle quali richiamabile tramite uno specifico URL.



*Panoramica route*

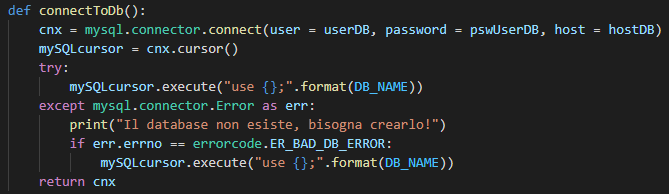
Ciascuna route può implementare una o più funzioni, ciascuna delle quali caratterizzata da un certo tipo di richiesta e/o dai parametri passati dal client.

Possono essere divise in tre gruppi, in relazione alle funzionalità e alle entità coinvolte:

* Route relative a fase di login e registrazione*:* ***/login, /logout, /insertCliente, /insertRistorante, /verificaUsername***
* Route relative a gestione funzionalità Cliente e Ristoratore: ***/OpenCloseRistorante, /menu, /pietanza, /ordine, /ordine\_pietanza***
* Route relative alle statistiche: ***/questionario***

## – Gestione Database

Oltre all’interazione client-server, viene anche gestita la connessione al Database MySQL per l’aggiunta, la rimozione, la modifica e la lettura dei dati relativi agli utenti e agli ordini. Per tale scopo è stata usata la libreria di Python “mysql-connector”. Il funzionamento prevede una prima fase di connessione al database tramite la funzione *connectToDb( ),* la quale restituisce un MySQLConnection object che verrà usato per l’esecuzione delle varie query.

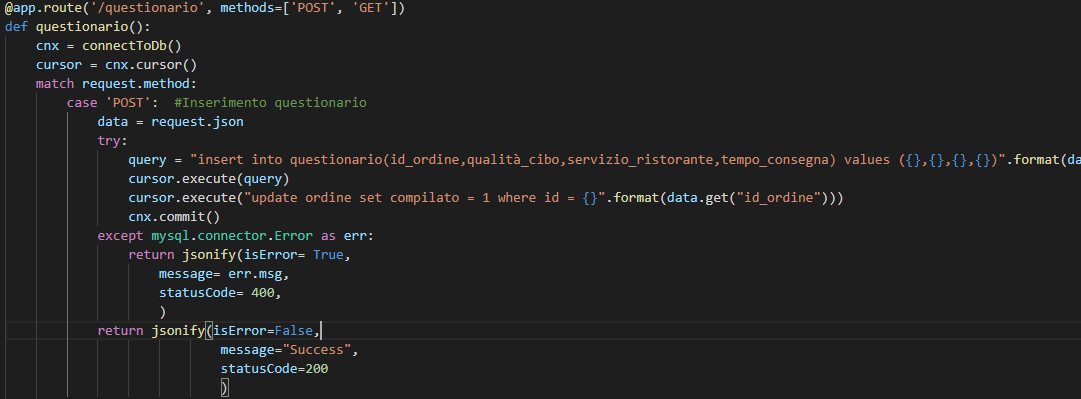


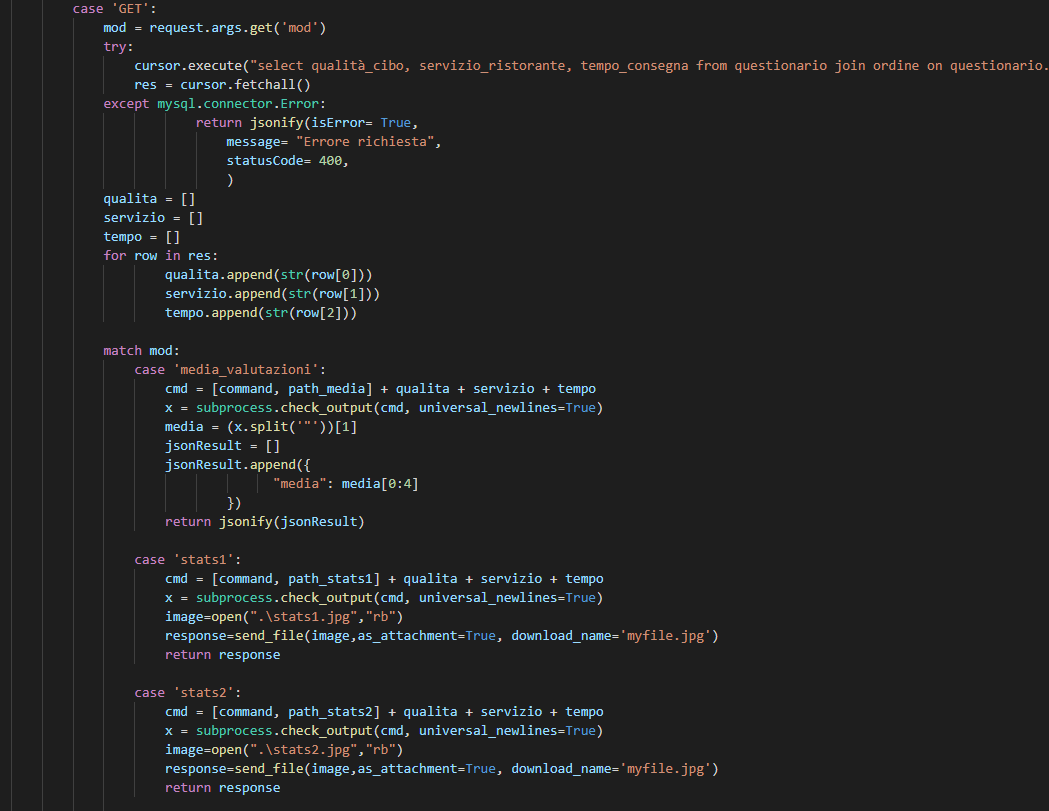
*Connessione al database*

## – Analisi di alcune scelte implementative

Al fine di garantire maggiore pulizia e leggibilità al codice è stato scelto di fornire più funzionalità a ciascuna route tramite la modifica dei parametri in ingresso piuttosto che associare a ciascuna funzionalità una route diversa.

La differenziazione può avvenire in un primo momento tramite il tipo di richiesta (GET, POST, PUT, DELETE), e nel caso in cui più funzionalità condividano lo stesso tipo, tramite uno switch-case. Un esempio si ha nella route relativa alle statistiche ordini. Si ha in particolare la possibilità di inserire dati nel database tramite una richiesta di tipo POST, ma anche di prelevare ciascuna statistica tramite richieste di tipo GET che differiscono per il valore del parametro “mod” passato in ingresso.





# Script R

# Conclusioni