UNIVERSITÀ DI PISA

Corso di Laurea in Informatica Umanistica

RELAZIONE

Mensana

Un'applicazione web per esplorare i dati della mensa universitaria

Candidato: *Martina Cinquini*

Relatore: *Anna Monreale*

Correlatore: *Beatrice Rapisarda*

Anno Accademico 2015-2016

Indice

Lista Figure

Introduzione

Questa tesi si propone di documentare la realizzazione dell'applicazione web Mensana.

Lo sviluppo di tale applicazione nasce dall'esigenza di fornire un supporto a coloro che collaborano con il progetto Mensana con l'obiettivo di far acquisire informazioni utili che possono suggerire analisi non previste inizialmente, nuovi esperimenti o campionamenti.

Il database su cui si basa l'applicazione è messo a disposizione dall'azienda regionale del Diritto allo Studio Universitario della Regione Toscana e contiene lo storico dei pasti erogati in ogni mensa dell'Università di Pisa dal 2010 al 2015 e i dati anagrafici degli studenti che hanno consumato i vari pasti.

Questi dati sono organizzati in un database relazionale implementato nel DBMS MySQL e popolato con il linguaggio di programmazione Python.

Attraverso una serie di interrogazioni mediante i linguaggi di programmazione PHP e SQL si sono recuperati i dati all'interno del database e con l'utilizzo di alcune librerie Javascript sono state realizzate rappresentazioni grafiche dei dati che, grazie al loro impatto visivo, sono di semplice lettura e di facile memorizzazione. Le interrogazioni consentono inoltre la possibilità di esportare alcune tabelle del database in relazione ai filtri presenti affinché l'utente possa effettuare una selezione basata sulle sue esigenze.

1. Il progetto Mensana

Il progetto Rasupea Mensana si propone di orientare gli studenti universitari verso un processo di consapevolezza del proprio stato di salute indicando i piatti giornalieri del servizio di ristorazione universitario più adeguati ad un corretto ed equilibrato stile alimentare.
Avviato nel 2016 e tutt'ora in corso, il progetto è finanziato[1] dalla Regione Toscana nell'ambito dei "Progetti di ricerca nel settore agro-alimentare" ed è coordinato dal dipartimento di Biologia dell'Università di Pisa e dalla Scuola Superiore Sant'Anna. Inoltre vede coinvolti il Dipartimento di Informatica e il Dipartimento Ricerca Traslazionale e delle Nuove Tecnologie in Medicina e Chirurgia dell'Università di Pisa.[1]
Per gli obiettivi progettuali, il DSU Toscana[2] ha fornito i dati anagrafici degli studenti e lo storico dei pasti consumati dal 2010 in ogni mensa dell'Università di Pisa. Questi dati sono stati organizzati in un database relazionale implementato nel DBMS MySQL e popolato con il linguaggio di programmazione Python. Nei paragrafi successivi verranno introdotti i concetti generali riguardanti il database relazionale e l'implementazione, poi verrà descritto lo schema del database, il popolamento e la creazione delle tabelle.

1.1 Introduzione ai RDBMS

Una base di dati è una collezione di dati strutturati gestita da un particolare sistema software che prende il nome di *Database Management System* (DBMS). Un DBMS ha quattro caratteristiche principali:

1. *efficienza*: garantisce l'accesso a grandi quantità di dati;
2. *affidabilità*: capacità del sistema di conservare sostanzialmente intatto il contenuto della base di dati in caso di malfunzionamento hardware e software;

[1] Decreto dirigenziale 22 aprile 2013, n. 1428. Bollettino Ufficiale Regione Toscana 31.12.2014 (http://www.regione.toscana.it/documents/10180/12190401/)
[2] Azienda regionale per il Diritto allo Studio Universitario della Toscana

3. *efficacia*: capacità di svolgere le operazioni utilizzando un insieme di risorse (tempo e spazio) che sia accettabile per gli utenti;
4. *privatezza*: capacità di limitare l'accesso ai dati agli utenti non autorizzati e di controllare la validità dei dati.

I dati del database rappresentano la struttura di un concetto del mondo reale che, tuttavia, non è direttamente comprensibile ad un calcolatore. Per questo è necessario definire un *modello dei dati*, ossia un insieme di meccanismi di astrazione per la rappresentazione di informazioni, che fornisca i concetti necessari a strutturare i dati di interesse. I modelli dei dati sono detti logici, per sottolineare il fatto che le strutture utilizzate da questi modelli, pur essendo astratte, riflettono una particolare organizzazione (ad alberi, a grafi, a tabelle o a oggetti) e al tempo stesso risultano indipendenti dalla reale struttura fisica dei dati. [2]

Nell'evoluzione delle basi di dati sono stati sviluppati differenti modelli:

- modello gerarchico: è stato il primo modello ad essere definito nella fase iniziale di sviluppo dei DBMS negli anni sessanta. Utilizza una struttura ad albero ed è adatto per rappresentare situazioni nelle quali esiste gerarchia fra i dati;

- modello reticolare: sviluppato in seguito a quello gerarchico. E' basato sull'uso di grafi per cui ogni nodo può essere connesso a più nodi e viceversa;

- modello relazionale: attualmente è il più diffuso. Organizza i dati in insiemi di record e rappresenta le relazioni attraverso tabelle;

- modello ad oggetti: sviluppato come evoluzione del modello relazionale estende alle basi di dati le caratteristiche dei linguaggi di programmazione ad oggetti.

Per sviluppare l'applicazione è stato utilizzato il modello relazionale.

1.1.1 RDBMS

Il modello relazionale, *Relational Database Management System* (RDBMS), nasce nel 1970 e si è largamente affermato nel secolo successivo. I RDBMS sono sistemi che utilizzano un modello logico di rappresentazione dei dati basato sulla teoria degli insiemi e sulla logica del primo ordine, la cui struttura fa leva sul concetto matematico di *relazione*, da cui il nome.

Dati n insiemi, non necessariamente disgiunti, D_1, D_2, ..., D_n, una relazione definita sugli insiemi D_1, D_2, ..., D_n è un qualunque sottoinsieme del prodotto cartesiano $D_1 \times D_2 \times ... D_n$. Gli insiemi D_1, D_2, ..., D_n sono detti domini della relazione. Il numero n dei domini di una relazione è detto grado della relazione. Gli elementi di una relazione sono detti tuple. Ogni tupla è costituita da tanti valori quanti sono i domini della relazione, cioè quanto è il grado della relazione stessa. (Atzeni, Basi di Dati, p.18)

In una tupla $<v_1, v_2, ..., v_k>$ di una relazione di grado k, il generico valore v_i appartiene al dominio D_i. Poiché una relazione è un insieme, l'ordine secondo il quale sono elencate le tuple di una relazione non ha importanza. Il numero delle tuple di una relazione è detto cardinalità della relazione.

Una relazione può essere rappresentata in forma tabellare (vedi Figura 1.1) usando le seguenti convenzioni:

1. ogni riga della tabella corrisponde ad una tupla della relazione;
2. ogni colonna della tabella corrisponde ai valori di uno dei domini di definizione della relazione.

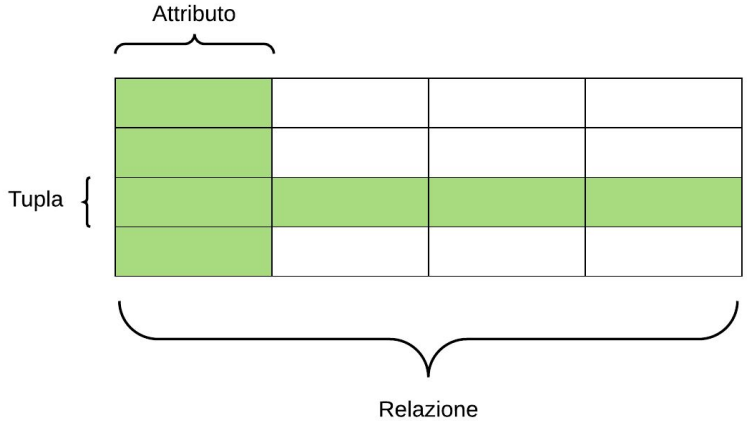


Figura 1.1 Rappresentazione tabellare di una relazione

Lo schema di una relazione è la lista dei nomi degli attributi usati per la rappresentazione tabellare della relazione stessa. Se una relazione ha nome R e gli attributi di R sono A_1, A_2, ..., A_k, lo schema della relazione è di solito rappresentato con R(A_1, A_2, ..., A_k). Indicando con A l'insieme di attributi A_1, A_2, ..., A_k si usa anche rappresentare lo schema con R(A).
L'insieme degli schemi delle relazioni utilizzate per rappresentare i dati e le corrispondenze fra essi è detto schema relazionale del database. L'insieme dei valori attuali delle relazioni è detto database relazionale.

1.2 MySQL

MySQL è un RDBMS multi-piattaforma distribuito dalla compagnia svedese "MySQL AB" come software libero sotto licenza GPL[3]. MySQL si occupa della strutturazione e della gestione dei dati in modo da velocizzare l'accesso, l'inserimento di nuovi elementi e la modifica.
Nell'applicazione web si utilizza phpMyAdmin[4], un software libero scritto in PHP che consente di amministrare il database fornendo un'interfaccia grafica come modalità di interazione con MySQL.

1.3 Lo schema del database Mensana

Il database Mensana è composto da 12 tabelle. Nel modello relazionale, il termine tabella indica l'insieme delle righe e delle colonne di una matrice di dati. Come definito nel paragrafo 1 sezione 1, la struttura di una tabella è specificata da una lista di colonne, ciascuna delle quali ha un nome univoco, un tipo di dato e un dominio, cioè un insieme di valori accettati.
I nomi delle tabelle sono stati codificati in caratteri *ASCII Unicode* e indicano con una breve descrizione il contenuto della tabella corrispondente. Al posto del carattere di spazio, nei nomi si utilizza il carattere underscore ('_') perché il DBMS non ammette nomi con spazi interni. Anche i nomi dei campi seguono le stesse regole utilizzate per nominare le tabelle. Nella Figura 1.2 è rappresentato lo schema del modello logico.

[3] General Public License versione 2
[4] https://www.phpmyadmin.net/

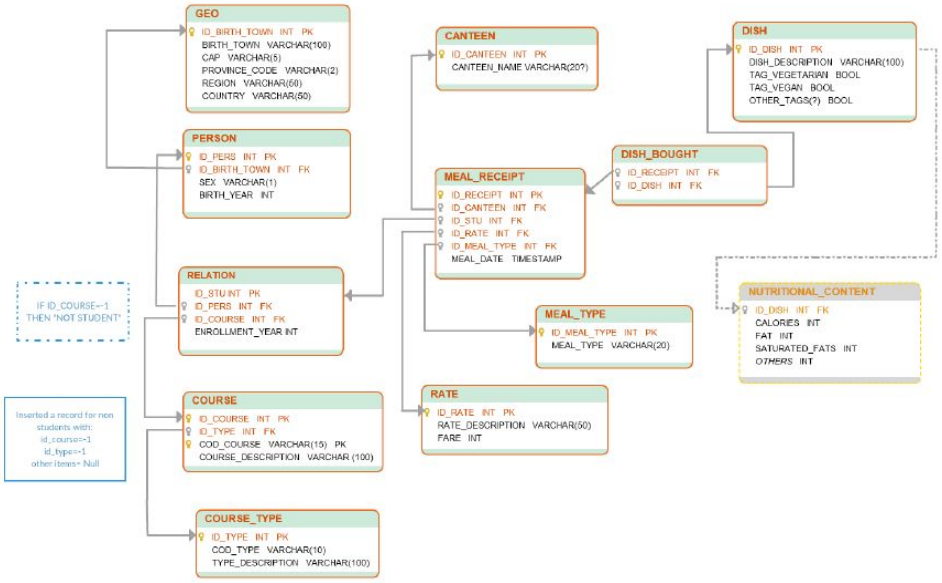


Figura 1.2 Schema del modello logico del database Mensana

La tabella centrale del modello logico è *meal_receipt* che contiene tutte le informazioni rilevanti dello scontrino di un pasto. Ogni tupla, infatti, specifica la mensa in cui il pasto è stato consumato, lo studente che lo ha consumato, il tipo di tariffa applicata e la data di consumazione. Inoltre, ad ogni scontrino è associato un identificatore incrementale (id_receipt) tramite il quale tutte le altre tabelle del database si riferiscono a meal_receipt.

In particolare le altre tabelle contengono le seguenti informazioni:

- tabella *relation:* contiene l'identificatore dello studente(id_stu) correlato all'identificatore della persona(id_pers), il codice del corso a cui lo studente è iscritto e l'anno di iscrizione. La correlazione tra id_stu e id_pers è basata sulla possibilità che lo stesso studente vada a mensa più volte. Pertanto allo stesso id_pers possono essere associati uno o più id_stu;
- tabella *course*: ogni tupla indica l'identificatore, la tipologia, il codice e la descrizione del relativo corso;
- tabella *course_type*: ogni tupla indica la tipologia del relativo corso, il codice e la descrizione della tipologia corrispondente;

- tabella *person:* raccoglie l'identificatore, il genere, l'anno di nascita e l'identificatore della città di nascita della persona;
- tabella *geo:* indica la provenienza di una persona specificando l'identificatore della città di nascita, il codice di avviamento postale, la sigla della provincia e la nazione;
- tabella *canteen:* contiene l'identificatore e il nome della mensa;
- tabella *rate:* contiene l'identificatore e la descrizione della tariffa;
- tabella *dish:* indica l'identificatore della ricevuta e del piatto;
- tabella *dish_bought:* raccoglie le informazioni relative ai piatti acquistati ovvero l'identificatore del piatto, la sua descrizione e la sua composizione;
- tabella *nutritional_content*: raccoglie le informazioni riguardanti la composizione chimica e il valore energetico degli alimenti.

1.4 Popolamento e creazione delle tabelle

Dopo aver caricato il database Mensana con le relative relazioni in MySQL, si è effettuato il popolamento della tabella *nutritional_content*. Inoltre, sono state aggiunte due tabelle: *users* e *macro_aree.* La prima raccoglie le informazioni di accesso all'applicazione di ogni utente (v. cap. 1, par. 4, sez. 2.1), mentre la seconda contiene la suddivisione in macro aree disciplinari dei corsi dell'Università di Pisa (v. cap. 1, par. 4, sez. 2.2).

1.4.1 Popolamento tabella nutritional_content

Il popolamento della tabella *nutritional_content* è stato possibile a seguito dell'associazione (*matching*) tra i dati presenti in due distinti file.
Il primo file, rappresentato in Figura 1.3, contiene gli ingredienti necessari alla preparazione dei piatti[5] distribuiti nelle mense universitarie di Pisa dal servizio di ristorazione del DSU Toscana[6]. Inoltre, sono specificate la quantità

[5] Nella tesi il termine "piatto" viene utilizzato per designare l'insieme degli ingredienti che lo compongono
[6] v. nota 2

dell'ingrediente utilizzato per la composizione del piatto e le relative caratteristiche.

L'obiettivo è assegnare ad ogni ingrediente l'identificatore (ID) del relativo piatto. La modalità di assegnamento e la scelta di Python come linguaggio di programmazione per effettuarlo sarà oggetto del prossimo paragrafo.

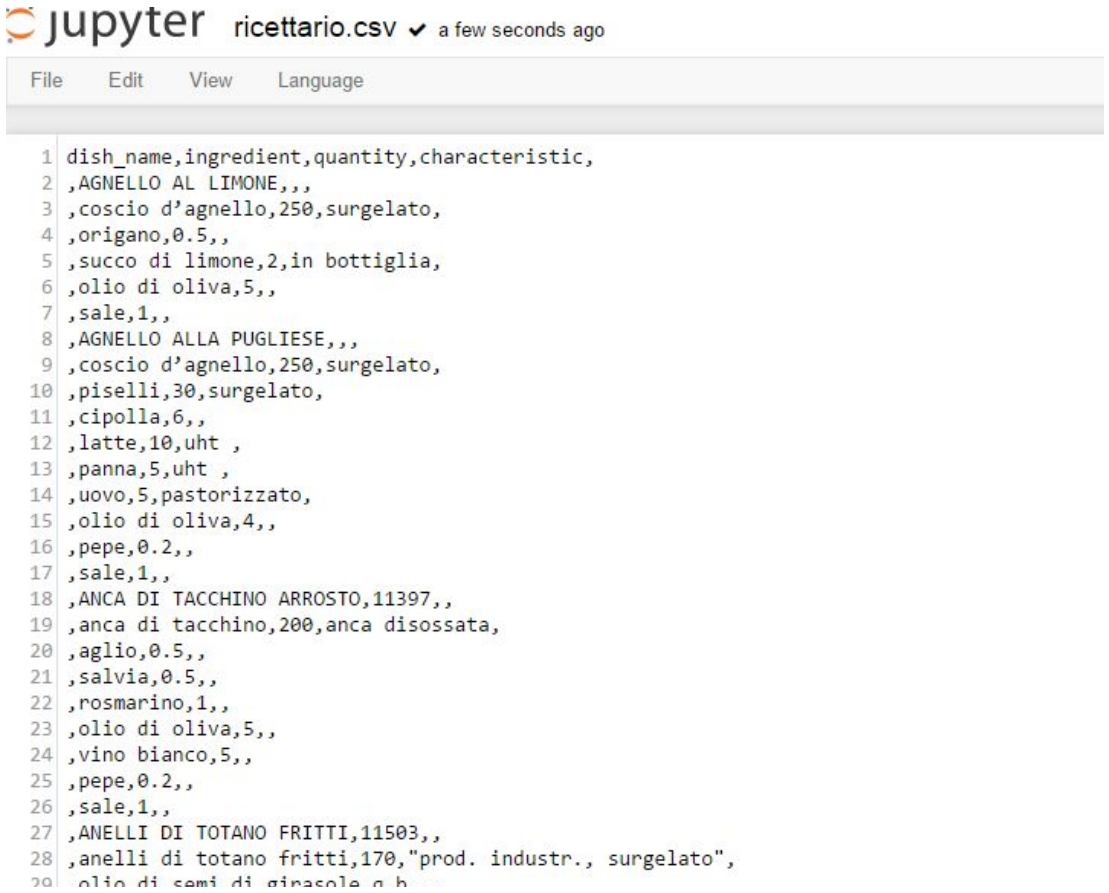
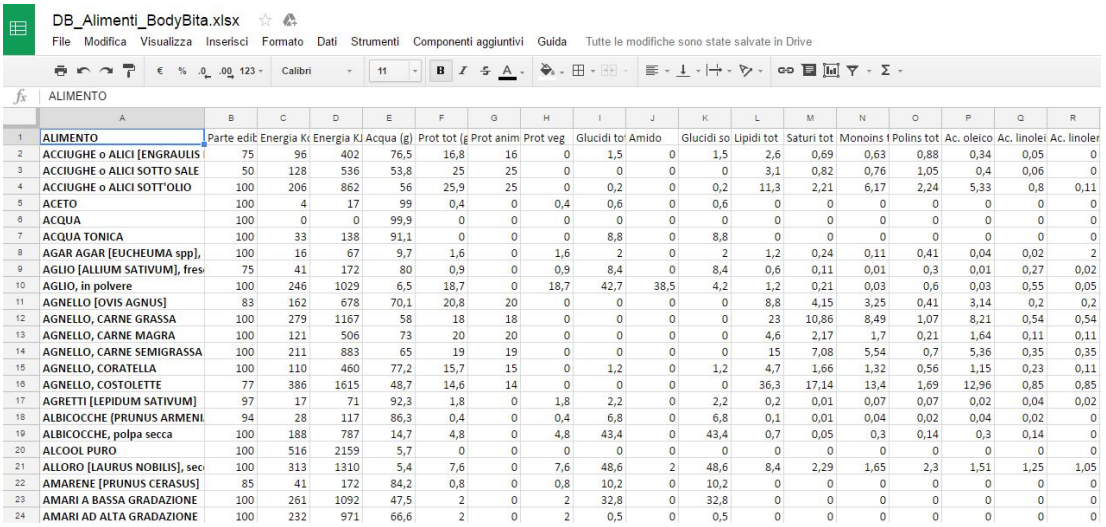


Figura 1.3. File *ricettario.csv*

Invece il secondo file (vedi Figura 1.4) contiene i dati sulla composizione chimica e sul valore energetico degli alimenti ed è fornito da Pharmanutra[7], un'azienda esterna al servizio di ristorazione DSU. Per questo motivo concerne una lista di alimenti generica e a sé stante dalla lista di ingredienti presenti nel primo file. Ciò che ci si propone è trovare la corrispondenza tra gli elementi delle due liste in modo da assegnare la composizione chimica e il valore energetico ad ogni ingrediente delle pietanze erogate nelle mense universitarie pisane. Il procedimento sarà descritto nel paragrafo 4, sezione 2.

[7] Azienda nutraceutica nata a Pisa nel 2003 (http://www.pharmanutra.it/it/azienda/)



DB_Alimenti_BodyBita.xlsx

File Modifica Visualizza Inserisci Formato Dati Strumenti Componenti aggiuntivi Guida Tutte le modifiche sono state salvate in Drive

ALIMENTO

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	ALIMENTO	Parte edit	Energia K	Energia KJ	Acqua (g)	Prot tot (g	Prot anim	Prot veg	Glucidi to	Amido	Glucidi so	Lipidi tot	Saturi tot	Monoins t	Polins tot	Ac. oleico	Ac. linolei	Ac. linoler
2	ACCIUGHE o ALICI [ENGRAULIS	75	96	402	76,5	16,8	16	0	1,5	0	1,5	2,6	0,69	0,63	0,88	0,34	0,05	0
3	ACCIUGHE o ALICI SOTTO SALE	50	128	536	53,8	25	25	0	0	0	0	3,1	0,82	0,76	1,05	0,4	0,06	0
4	ACCIUGHE o ALICI SOTT'OLIO	100	206	862	56	25,9	25	0	0,2	0	0,2	11,3	2,21	6,17	2,24	5,33	0,8	0,11
5	ACETO	100	4	17	99	0,4	0	0,4	0,6	0	0,6	0	0	0	0	0	0	0
6	ACQUA	100	0	0	99,9	0	0	0	0	0	0	0	0	0	0	0	0	0
7	ACQUA TONICA	100	33	138	91,1	0	0	0	8,8	0	8,8	0	0	0	0	0	0	0
8	AGAR AGAR [EUCHEUMA spp],	100	16	67	9,7	1,6	0	1,6	2	0	2	1,2	0,24	0,11	0,41	0,04	0,02	2
9	AGLIO [ALLIUM SATIVUM], fres	75	41	172	80	0,9	0	0,9	8,4	0	8,4	0,6	0,11	0,01	0,3	0,01	0,27	0,02
10	AGLIO, in polvere	100	246	1029	6,5	18,7	0	18,7	42,7	38,5	4,2	1,2	0,21	0,03	0,6	0,03	0,55	0,05
11	AGNELLO [OVIS AGNUS]	83	162	678	70,1	20,8	20	0	0	0	0	8,8	4,15	3,25	0,41	3,14	0,2	0,2
12	AGNELLO, CARNE GRASSA	100	279	1167	58	18	18	0	0	0	0	23	10,86	8,49	1,07	8,21	0,54	0,54
13	AGNELLO, CARNE MAGRA	100	121	506	73	20	20	0	0	0	0	4,6	2,17	1,7	0,21	1,64	0,11	0,11
14	AGNELLO, CARNE SEMIGRASSA	100	211	883	65	19	19	0	0	0	0	15	7,08	5,54	0,7	5,36	0,35	0,35
15	AGNELLO, CORATELLA	100	110	460	77,2	15,7	15	0	1,2	0	1,2	4,7	1,66	1,32	0,56	1,15	0,23	0,11
16	AGNELLO, COSTOLETTE	77	386	1615	48,7	14,6	14	0	0	0	0	36,3	17,14	13,4	1,69	12,96	0,85	0,85
17	AGRETTI [LEPIDUM SATIVUM]	97	17	71	92,3	1,8	0	1,8	2,2	0	2,2	0,2	0,01	0,07	0,07	0,02	0,04	0,02
18	ALBICOCCHE [PRUNUS ARMENI	94	28	117	86,3	0,4	0	0,4	6,8	0	6,8	0,1	0,01	0,04	0,02	0,04	0,02	0
19	ALBICOCCHE, polpa secca	100	188	787	14,7	4,8	0	4,8	43,4	0	43,4	0,7	0,05	0,3	0,14	0,3	0,14	0
20	ALCOOL PURO	100	516	2159	5,7	0	0	0	0	0	0	0	0	0	0	0	0	0
21	ALLORO [LAURUS NOBILIS], sec	100	313	1310	5,4	7,6	0	7,6	48,6	2	48,6	8,4	2,29	1,65	2,3	1,51	1,25	1,05
22	AMARENE [PRUNUS CERASUS]	85	41	172	84,2	0,8	0	0,8	10,2	0	10,2	0	0	0	0	0	0	0
23	AMARI A BASSA GRADAZIONE	100	261	1092	47,5	2	0	2	32,8	0	32,8	0	0	0	0	0	0	0
24	AMARI AD ALTA GRADAZIONE	100	232	971	66,6	2	0	2	0,5	0	0,5	0	0	0	0	0	0	0

Figura 1.4 Estratto del file contenente gli alimenti

1.4.1.1 Associazione dell'identificatore del piatto a ogni ingrediente

L'associazione dell'identificatore del piatto a ogni ingrediente che lo compone è stata effettuata utilizzando come linguaggio di programmazione Python.

Python[8] è un linguaggio di programmazione ad alto livello orientato agli oggetti ideato da Guido van Rossum all'inizio degli anni novanta.
Alcune tra le caratteristiche principali di questo linguaggio sono le variabili non tipizzate e l'uso dell'indentazione per la definizione di blocchi nidificati. Progettato per essere altamente leggibile, esso è un forte supporto all'integrazione con altri linguaggi e programmi ed è fornito di una libreria standard, a sua volta estendibile con altri moduli, che lo rende adatto a molti impieghi.
In questo caso specifico, i moduli utilizzati per l'associazione sono:

- *os*: modulo per eseguire operazioni del sistema operativo sottostante, ad esempio copiare, rinominare o cancellare un file;

[8] http://www.python.it/

- *re*: modulo per implementare la gestione efficiente delle espressioni regolari per la ricerca di corrispondenze di testo (*pattern matching*).

La realizzazione del codice è stata effettuata nell'editor Jupyter Notebook all'interno di Anaconda[9], un ambiente autocontenuto per lo sviluppo di applicazioni per il calcolo scientifico in Python. Anaconda è completamente gratuito per l'utilizzo accademico e contiene al suo interno molte librerie che permettono di programmare in maniera intuitiva e combinare moduli Python.

In Figura 1.5 è riportato un estratto del file contenente il risultato dell'associazione mentre il codice utilizzato è inserito nell'appendice (v. cap. 7, par. 1).

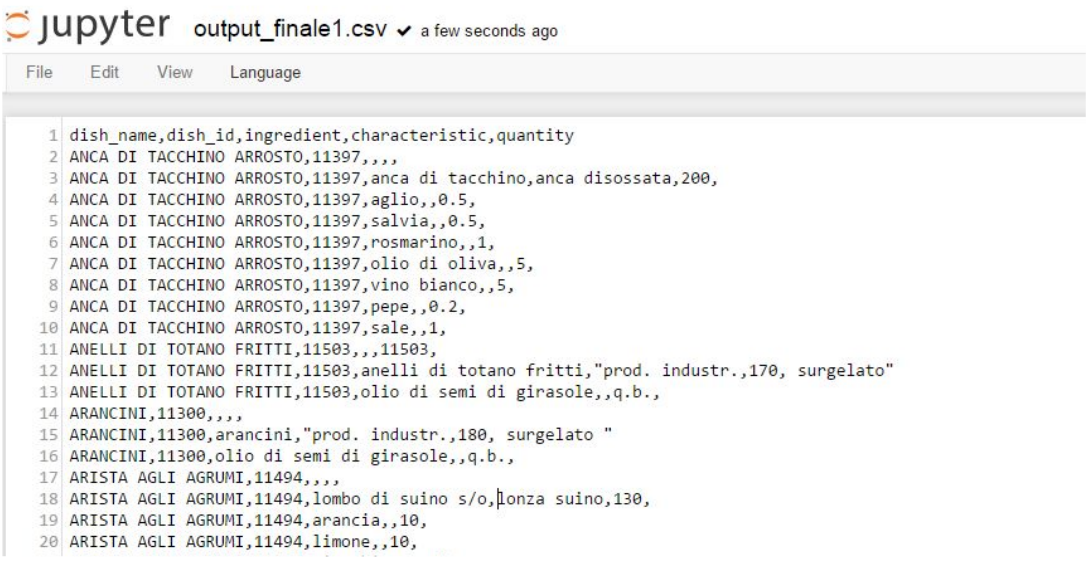


Figura 1.5 Parte del file che contiene il risultato dell'associazione

1.4.1.2 Assegnamento composizione chimica e valore energetico a ogni ingrediente

Il punto di partenza per effettuare l'assegnazione della composizione chimica e del valore energetico agli ingredienti che compongono il piatto è il risultato dell'associazione descritta nel paragrafo precedente e contenuta nel file "output_finale.csv". Anche in questo caso, il linguaggio di programmazione

[9] http://docs.continuum.io/

usato è Python ma le librerie e i moduli importati sono differenti. In particolare, oltre ai precedenti, si utilizza:

- *pandas:* libreria utilizzata per leggere e manipolare dati multidimensionali;
- *codecs:* modulo che definisce le classi di base per i codec standard Python (encoder e decoder) e fornisce l'accesso al registro delle codifiche interno a Python, che gestisce le codifiche stesse e i processi di controllo della gestione degli errori;
- *json*: modulo per la conversione in memoria di oggetti Python verso una rappresentazione serializzata nota come JavaScript Object Notation.

E' necessario considerare come punto di partenza il file "output_finale.csv" poiché, prima di poter assegnare le caratteristiche chimiche ed energetiche agli alimenti, occorre trovare la corrispondenza tra gli ingredienti presenti nel ricettario fornito dal DSU e gli ingredienti presenti nel ricettario fornito dall'azienda esterna (v. cap. 1, par. 4, sez. 2).

Per ogni ingrediente contenuto nel file "output_file.csv" si crea una lista contenente le associazioni significative tra gli alimenti. La lista è ordinata in base al numero di occorrenze di un determinato alimento e sono restituiti i primi quindici alimenti più simili. Quando l'esito dell'associazione è positivo, il *match* è inserito in un dizionario che è necessario ad evitare il ripetersi del procedimento per un ingrediente già analizzato in precedenza. Di seguito, sono riportati una parte dell'output prodotto dal *matching* e un estratto del file risultante mentre il codice utilizzato è inserito in appendice (v. cap. 7, par.2).

```
('Ingrediente: ', 'piselli', ' - Caratteristica: ', 'surgelato')
('AGNELLO ALLA PUGLIESE', ['PISELLI'], 1, ['PISELLI PISUM SATIVUM freschi'])
('AGNELLO ALLA PUGLIESE', ['PISELLI'], 1, ['PISELLI IN SCATOLA'])
('AGNELLO ALLA PUGLIESE', ['PISELLI'], 1, ['PISELLI IN SCATOLA SGOCCIOLATI'])
('AGNELLO ALLA PUGLIESE', ['PISELLI'], 1, ['PISELLI SURGELATI'])
('AGNELLO ALLA PUGLIESE', ['PISELLI'], 1, ['PISELLI secchi'])
Inserisci la scelta (N:B. NOT FOUND se non trovi il match):PISELLI SURGELATI
[('OLIO DI OLIVA', 'OLIO DI OLIVA'), ('COSCIO D AGNELLO', 'AGNELLO CARNE SEMIGRASSA'), ('PISELLI', 'PISELLI SURGELATI'), ('SA
LE', 'SALE da cucina'), ('ORIGANO', 'ORIGANO secco macinato'), ('SUCCO DI LIMONE', 'SUCCO DI LIMONE CONSERVATO')]
```

Figura 1.6. Esempio dell'associazione dell'ingrediente "piselli" tra i due distinti file

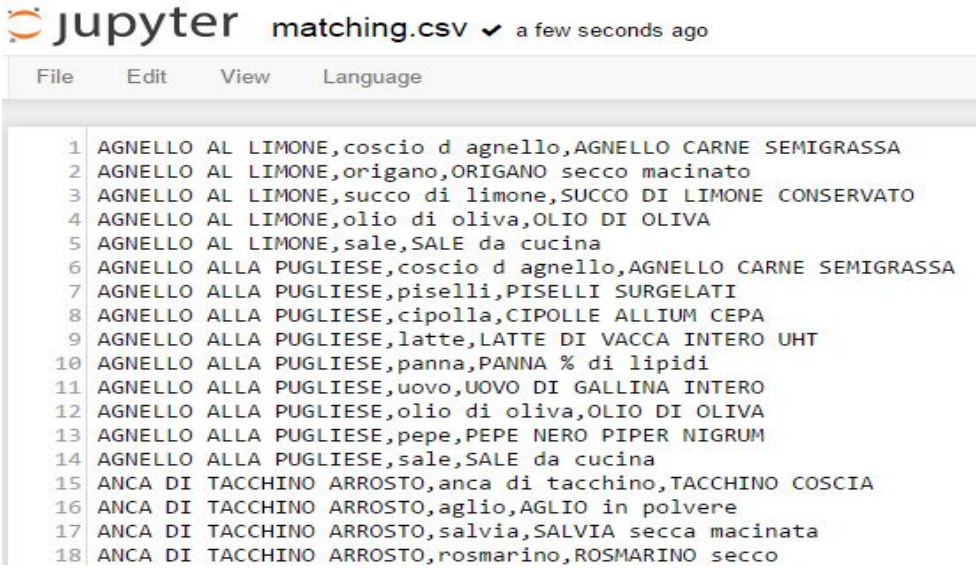


Figura 1.7. Risultato della corrispondenza tra i due file

Lo step successivo riguarda l'effettiva associazione tra l'ingrediente e la sua composizione chimica.

Dato che il file contenente le caratteristiche riporta i valori per ognuna di esse relativi a una parte edibile di 100 grammi, occorre assegnare a ciascun ingrediente i valori corretti calcolati in proporzione alla sua quantità presente nel piatto. Si seleziona il valore della caratteristica e della relativa quantità, i rispettivi valori sono moltiplicati tra loro. Infine il prodotto è diviso per 100. La procedura è ripetuta per ogni caratteristica dell'ingrediente. Di seguito, è riportato la parte di codice Python utilizzato per l'associazione:

```
(..)
finale['quantity']=pd.to_numeric(finale['quantity'], errors='coerce')
finale['Kcal_piatto'] = 0
for x in range(len(finale)):
    if finale.iloc[x,4] is not 'NaN':
        quantita=finale.iloc[x,4]
        calorie=finale.iloc[x,8]
        calorie_totali=round((quantita*calorie)/100,1)
        finale.iloc[x,44]=calorie_totali
(..)
```

1.4.2 Creazione tabelle

DDL, acronimo di *Data Definition Language*, è il linguaggio di "definizione dei dati utilizzato per definire gli schemi logici esterni e fisici e le autorizzazioni per l'accesso." (Atzeni, Basi di Dati, p. 10).

Lo schema di una tabella si definisce mediante l'istruzione `CREATE TABLE` seguita dal nome della tabella e dalla definizione della sua struttura esplicitata all'interno di parentesi tonde che seguono il nome. All'interno delle parentesi sono elencati i diversi campi che compongono la tabella per ognuno dei quali è specificato il tipo di dato e la dimensione. Per identificare la tipologia di valori che una colonna può contenere esistono vari tipi di dati:

- stringhe alfanumeriche di caratteri
- valori numerici che possono essere esatti o approssimati
- valori booleani
- valori temporali

E' necessario identificare correttamente il tipo di valore in modo da ottimizzare le prestazioni di MySQL. Dopo aver specificato il tipo di dato è possibile definire altri attributi facoltativi come `AUTO_INCREMENT` che determina una funzione di auto-incremento della relativa colonna impostando il primo valore ad 1 ed aggiungendo un'unità per ogni nuovo record o `NOT NULL` che vieta la presenza di valori nulli.

Inoltre, è possibile stabilire i vincoli di integrità dei dati ossia proprietà che devono essere soddisfatte dalle istanze delle basi di dati e possono essere *intrarelazionali*, se il suo soddisfacimento è definito rispetto a singole relazioni della base di dati, oppure *interrelazionali*, se coinvolge più relazioni.

Tra i vincoli intrarelazionali vi sono: i vincoli di tupla che esprimono condizioni sui valori di ciascuna tupla indipendentemente dalle altre e i vincoli di chiave che identificano univocamente le tuple della relazione dove esse sono definite. Mentre tra i vincoli interrelazionali vi è il vincolo di integrità referenziale per il quale informazioni in relazioni diverse sono correlate attraverso valori comuni.[2] I vincoli risultano utili al fine di descrivere la realtà di interesse in modo accurato, costituiscono uno

strumento di ausilio alla progettazione e infine sono utilizzati dal sistema nella scelta della strategia di esecuzione delle interrogazioni. [3]

Di seguito, viene illustrata la sintassi per la creazione di una tabella. I valori tra parentesi quadre sono facoltativi ed "(..)" indica la possibilità di inserire un numero arbitrario di campi.

```
CREATE TABLE nome_tabella (
nome_campo (dimensione) tipo di dato [attributo] [vincolo]
(..)
)
```

1.4.2.1 Tabella users

Lo scopo della creazione della tabella *users* è la memorizzazione delle credenziali dell'utente necessarie per accedere all'applicazione web.
Si utilizza il comando `CREATE TABLE` seguito dal nome *users* e all'interno delle parentesi tonde si definiscono quattro campi con le relative specifiche:

- id: indica l'identificatore dell'utente. Il dominio del campo è un valore numerico esatto indicato con `INTEGER`, è vietata la presenza di valori nulli tramite `NOT NULL` e si utilizza `AUTO_INCREMENT` per far sì che il valore del campo venga incrementato automaticamente per ogni nuovo utente;
- nome: contiene il nome e il cognome dell'utente. Il dominio del campo è una stringa di testo con lunghezza massima trenta caratteri. Anche in questo caso è vietata la presenza di valori nulli;
- email: indica l'indirizzo di posta elettronica dell'utente. Il dominio del campo è una stringa alfanumerica con lunghezza massima trenta caratteri. Il campo non può essere nullo;

- password: contiene la password scelta dall'utente. Il dominio del campo indicato con `VARCHAR` ed è una stringa alfanumerica con lunghezza massima 40 caratteri. Il campo non può essere nullo.

Inoltre, si precisano i vincoli intrarelazionali. Il campo "id" è definito come chiave primaria mentre il campo "email" come chiave unica. La figura seguente indica il codice SQL utilizzato per la creazione della tabella:

```
CREATE TABLE `users` (
  `id` int(8) NOT NULL AUTO_INCREMENT,
  `name` varchar(30) NOT NULL,
  `email` varchar(30) NOT NULL,
  `password` varchar(40) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `email` (`email`)
)
```

1.4.2.2 Tabella macro_aree

La finalità della creazione della tabella *macro_aree* è il raggruppamento di corsi di studio, erasmus, specializzazioni e dottorati, appartenenti a diversi settori disciplinari, in un'unica macroarea di riferimento.

Si utilizza il comando `CREATE TABLE` seguito dal nome *macro_aree* e all'interno delle parentesi tonde si definiscono due campi con le relative specifiche:

- id_area: descrive l'identificatore della macro area. Il dominio del campo è un valore numerico esatto con lunghezza massima di 8 cifre. Viene indicato `NOT NULL` affinché il campo non risulti vuoto;
- descrizione_area: contiene la descrizione della macro area. Il dominio del campo è una stringa alfanumerica con lunghezza massima di 100 caratteri. Anche in questo caso, si indica `NOT NULL` affinché il campo non sia vuoto.

Al campo “id_area” è aggiunto il vincolo intrarelazionale di chiave primaria. Di seguito, si mostra il codice SQL utilizzato per la creazione della tabella.

```
CREATE TABLE macro_aree (
id_area integer(11) NOT NULL,
descrizione_area varchar(100) NOT NULL,
PRIMARY KEY (id_area)
)
```

2. La struttura web

Un'applicazione web è qualsiasi software eseguito su un browser web. Il software non necessita di essere installato nel computer poiché si rende disponibile su un server in rete e viene eseguito dal browser in posizione di client. Inizialmente, il client instaura una connessione con il server e invia la richiesta per un servizio. Il server elabora i dati necessari e rende disponibile il servizio richiesto al client.
In questo capitolo saranno descritti gli strumenti di sviluppo lato client con i quali si è realizzata l'applicazione in seguito l'architettura web poi il design di Mensana.

2.1 Strumenti di sviluppo

L'interfaccia utente dell'applicazione web è realizzata con l'utilizzo dei seguenti strumenti: HTML, CSS, Javascript e jQuery.

2.1.1 HTML

HTML[10] (*HyperText Markup Language*) è un linguaggio di marcatura descrittivo ideato nel 1989 da Tim Berners Lee e utilizzato per trasformare documenti testuali in pagine web.
Le caratteristiche fondamentali di HTML, riflesse anche dal nome stesso, sono:

1. *l'ipertestualità*: permette di stabilire collegamenti all'interno di uno o più documenti senza dover seguire un percorso sequenziale per raggiungere l'informazione;
2. *la marcatura*: assicura una corretta separazione tra la struttura e la visualizzazione del testo incorporando, oltre all'informazione che si vuole comunicare attraverso la pagina web, particolari sequenze di caratteri (*marcatori*) che descrivono le modalità di visualizzazione

[10] www.html.it

grafica del documento. I marcatori contengono una sequenza di caratteri racchiusa tra due parentesi angolari ("<" e ">") e hanno un nome che identifica univocamente il valore dell'etichetta.

Quando devono essere applicati a una sezione di testo o di codice, l'ambito di applicazione deve essere delimitato fra un tag di apertura ed uno di chiusura, che coincide col *tag* di apertura preceduto da una barra (/) dopo la parentesi angolare aperta (Es.: `<p>il mio progetto</p>` In questo caso, il testo compreso tra questi due tag verrà visualizzato come un paragrafo dal browser). [4]

Nel corso degli anni si sono sviluppate diverse versioni di questo linguaggio di markup, l'ultima, utilizzata anche per creare la struttura dell'applicazione, è la versione numero 5.

2.1.2 CSS

CSS[11] (*Cascading Style Sheets*) è un linguaggio usato per definire lo stile di una pagina HTML. Questa tecnologia è stata introdotta per separare lo stile dei contenuti di una pagina dalla formattazione.

Un foglio di stile CSS è sintatticamente strutturato come una sequenza di regole, ovvero coppie costituite da un selettore e un blocco di dichiarazioni racchiuso tra parentesi graffe. Un selettore è un predicato che individua certi elementi del documento HTML e che racchiude una o più dichiarazioni, separate con un punto e virgola, e a sua volta costituite dalla coppia proprietà-valore.

Si possono individuare quattro tipi di selettori:

- universale: definito sintatticamente da un asterisco (*). La sua funzione è quella di selezionare tutti gli elementi presenti in un documento

  Esempio: `* {color: black};`

[11] https://www.w3.org/Style/CSS/Overview.en.html

- di tipo: rappresentati dal nome di uno specifico elemento HTML. Servono a selezionare tutti gli elementi di quel tipo specifico presenti in un documento

  Esempio: `p {color: green};`
- di classe: seleziona gli elementi a cui è stata assegnata una classe che presenta la proprietà `class="titolo"`. Il nome della classe viene fatto precedere dal punto

  Esempio: `.titolo {color: red};`
- di identificatore: seleziona univocamente l'elemento a cui è stato assegnato un id che presenta la proprietà `id="titolo"`. Il nome dell'identificatore viene fatto precedere dal carattere cancelletto

  Esempio: `#titolo {color: red};`

L'applicazione utilizza l'ultima versione di CSS, ovvero CSS3. Ad ogni pagina HTML, è collegato lo stesso foglio di stile dichiarato nella sezione `<head>` tramite il tag HTML `<link>`.

```
<head>
[...]
     <link rel="stylesheet" href="style.css" />
[...]
</head>
```

2.1.3 Javascript

JavaScript è un linguaggio di scripting lato client e orientato agli oggetti. Nel suo utilizzo più frequente, quello della programmazione per il web, consiste in un linguaggio formale che fornisce al browser determinate istruzioni da compiere. Una pagina creata in HTML è infatti statica, in quanto una volta che la pagina è stata interpretata dal browser la disposizione degli elementi rimane immutata, così come il loro contenuto. Tramite JavaScript, invece, è possibile conferire dinamicità alle pagine web permettendo, ad esempio, di creare *rollover* sulle immagini o modificare i contenuti in base a input dell'utente.

Il codice JavaScript viene integrato all'interno della pagina HTML grazie al tag `<script>`. E' possibile inserire uno o più tag ed il motore JavaScript elabora i vari contenuti partendo dal primo blocco in alto e proseguendo, in sequenza, verso il basso. In alcuni casi si utilizza JavaScript per manipolare elementi HTML e dato che, anche l'interprete HTML elabora la struttura HTML dall'alto verso il basso, è fondamentale inserire il blocco JavaScript dopo l'elemento HTML di riferimento. [5]

2.1.4 jQuery

jQuery è una libreria Javascript che permette al programmatore di modificare la struttura ad albero del documento HTML, gestire gli eventi, le animazioni e le funzionalità AJAX. È tra le librerie più utilizzate in quanto consente di scrivere il codice in modo compatto e ad alto livello. [5]

I comandi jQuery si basano sull'assunto che il documento HTML contiene oggetti che possono essere richiamati attraverso i propri id o classi, esattamente come in CSS. La sintassi di base è:

```
$(selettore).action()
```

Il simbolo del dollaro è un *alias* per jQuery. Il selettore serve a selezionare e manipolare gli elementi HTML. Questi ultimi, come nel linguaggio CSS, vengono preceduti dal cancelletto se sono identificatori oppure dal punto se sono classi. Infine, l'action è l'azione che si vuole applicare all'elemento selezionato.

Per includere jQuery nell'applicazione web è stato inserito uno script nella sezione `<head>` di ogni pagina HTML contenente il link della CDN[12] di Google.

```
<head>
[...]
     <script
     src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js">
     </script>
[...]
</head>
```

[12] Content Delivery Network

In particolare, nell'applicazione web sono stati utilizzati:

- i metodi per eseguire animazioni mediante le funzioni `show()` e `hide()` per le *checkbox*[13] nelle pagine "Studenti" e "Piatti";
- il metodo `val()`: il primo metodo per restituire il valore dell'elemento corrispondente nella pagina HTML.

2.2 Applicazione web Mensana

Mensana è il nome scelto per l'applicazione web. Il termine deriva da una sentenza della decima satira di Giovenale "*Orandum est ut sit mens sana in corpore sano*" (bisogna credere agli dèi che la mente sia sana in un corpo sano). Questa scelta nasce dall'esigenza di citare il nome del progetto correlato. Inoltre si è esplicitato un nome secondario per specificare la funzionalità per cui l'applicazione è stata creata ovvero "accesso ai dati".

2.2.1 Architettura web

La struttura web di Mensana è suddivisa in quattro pagine HTML:

1. Login
2. Analisi
3. Studenti
4. Piatti

La pagina "Login" (vedi Figura 2.1) è costituita da due form HTML che permettono la registrazione o l'accesso all'applicazione web. Nella sezione `<header>` della pagina web sono inseriti il titolo dell'applicazione, il logo e il menù. Gli elementi del menù possono essere cliccati ma restituiscono un messaggio di errore per invitare l'utente ad effettuare l'accesso.

[13] Elemento HTML che consente all'utente di effettuare scelte multiple

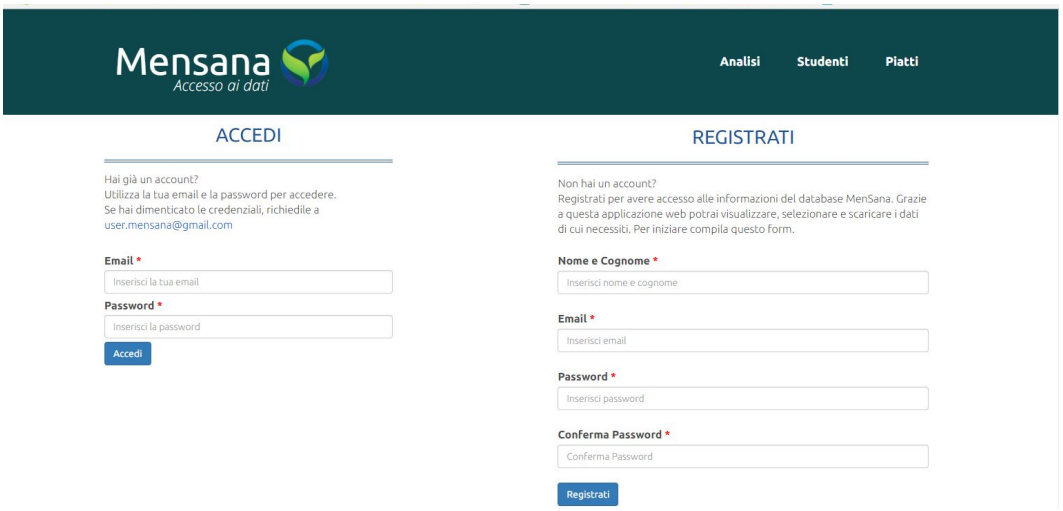


Figura 2.1 Pagina Login

Dopo l'autenticazione effettuata con le proprie credenziali, è possibile accedere alla pagina "Analisi" relativa alle analisi svolte sui dati del database e rappresentate mediante l'uso di grafici. Il titolo e il logo di Mensana sono posti in alto a sinistra mentre, in alto a destra, è collocato il menu che consente all'utente di navigare tra le altre pagine dell'applicazione. In questa pagina è inserito anche un menu di navigazione secondario che permette di scegliere quale analisi rendere visibile.

Le pagine "Studenti" (vedi Figura 2.2) e "Piatti" sono dedicate alla visualizzazione ed esportazione dei dati presenti nel database. Entrambe presentano la stessa struttura web ma coinvolgono differenti tabelle del database. In particolare, "Studenti" interessa *meal_receipt, relation, course, course_type, person, geo* e fornisce una panoramica sulle caratteristiche degli studenti dal punto di vista anagrafico e della carriera universitaria. Invece "Piatti" interessa *meal_receipt, canteen, rate, dish e dish_bought* e fornisce informazioni rispetto ai piatti erogati nelle varie mense presenti nel database.

Le pagine web sono suddivise in tre sezioni:

- filtri disponibili: sezione che consente di selezionare, grazie all'uso dei filtri, i dati specifici di cui l'utente necessita;
- informazioni filtri: sezione che illustra le caratteristiche dei filtri;

- anteprima della tabella: sezione che mostra un'anteprima della tabella che l'utente potrà scaricare nel formato *.csv*[14].

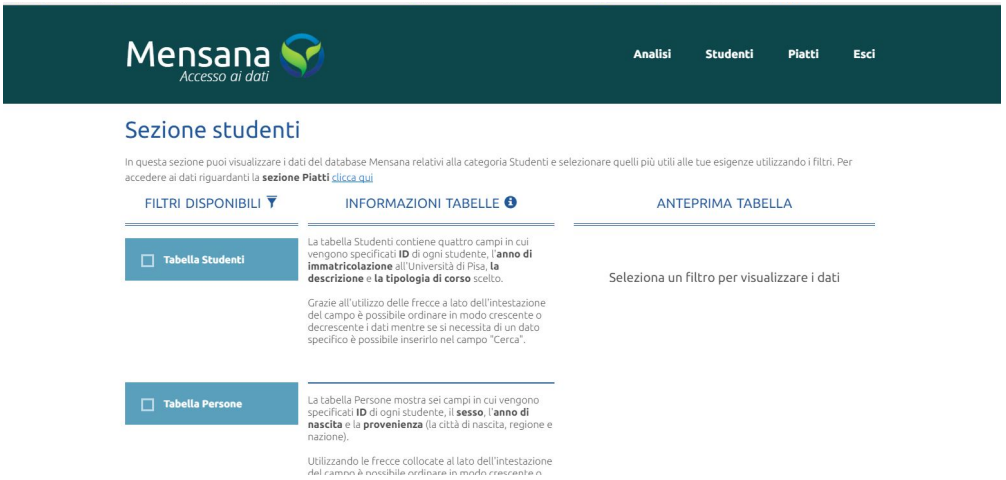


Figura 2.2. Pagina Studenti

2.2.2 Il design

La realizzazione del design per l'applicazione Mensana è stata preceduta da una fase di progettazione.

Inizialmente si è definito il target di riferimento degli utenti che è stato individuato in un gruppo ristretto, composto principalmente da biologi, medici e dottorandi del CNR-Area di Pisa[15] e dotato di credenziali specifiche per accedere all'applicazione. I potenziali utenti sono soggetti che conoscono il progetto Rasupea Mensana e che accedono all'applicazione per esaminare le analisi svolte sui dati presenti nel database, visualizzare le tabelle ed eventualmente scaricarle.

In seguito, si è pensato all'organizzazione dei contenuti all'interno di ogni pagina. L'obiettivo è una struttura chiara e logica in modo tale che le funzioni vengano svolte con il minimo numero di click possibile.[6] Per questo, i contenuti sono raggruppati in aree che contengono informazioni concettualmente omogenee tra loro. Inoltre, sono organizzati in modo da rendere chiaramente distinguibili le informazioni principali da quelle

[14] L'estensione .csv è acronimo di Comma-Separated Values ed è usato per distinguere i file di testo che sono liste di dati convenzionalmente separati dalla virgola o dal punto e virgola.
[15] Consiglio Nazionale delle Ricerche

secondarie o di supporto ponendo quelle rilevanti all'inizio, immediatamente visibili. [7]

Altro aspetto organizzativo di primaria importanza, è la scelta dell'etichetta degli elementi affinché risulti sempre descrittiva rispetto alla funzione che l'elemento svolge utilizzando concetti significativi, comprensibili e facilmente identificabili.

Dopo queste considerazioni, si è iniziato a realizzare l'applicazione.

La prima valutazione è stata la scelta del *font*, ossia una serie completa di caratteri dello stesso tipo, distinti per stile e corpo. Si sono tenuti presenti principalmente due fattori: la leggibilità e la compatibilità multipiattaforma.

Il font scelto per l'applicazione è Ubuntu[16] (vedi Figura 2.3), caratterizzato da una linea chiara. Inoltre per facilitare la leggibilità, i caratteri sono senza grazie e le porzioni testuali sono scritte in nero su sfondo bianco in modo da evidenziare il contrasto.

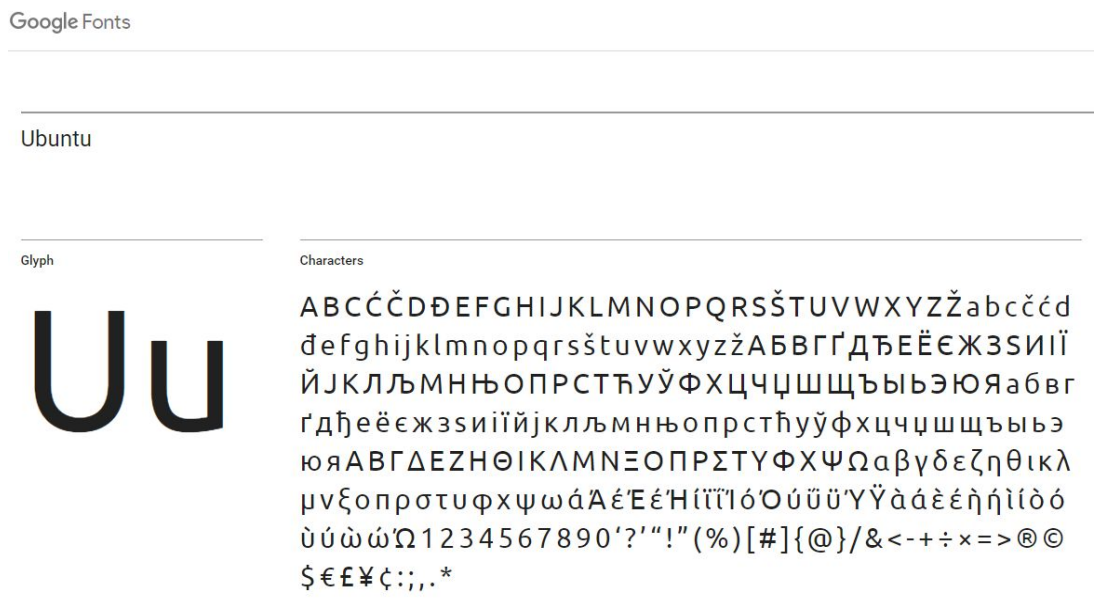


Figura 2.3 Google font Ubuntu

[16] https://fonts.google.com/specimen/Ubuntu

La seconda valutazione è stata la scelta dei colori dell'applicazione.
L'intento era quello di richiamare la colorazione del logo di Mensana (vedi Figura 2.4), basata sulle tonalità del blu e del verde, e quello di garantire una coerenza visuale sia all'interno della pagina navigata sia tra le diverse pagine dell'applicazione. [9]

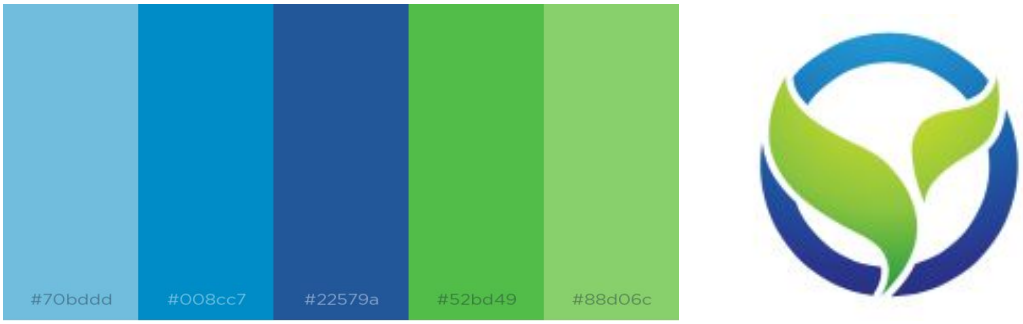


Figura 2.4 Palette di colori e logo Mensana

Inoltre si è sviluppato un layout responsive, cioè capace di adattarsi al dispositivo con il quale l'applicazione è visualizzata, con la finalità di ottimizzare l'esperienza di navigazione dell'utente riducendo al minimo la necessità di ridimensionare e scorrere i contenuti.

Figura 2.5. Sezione Studenti in un dispositivo mobile

3. Analisi di Mensana

In questo capitolo saranno illustrate la progettazione, lo sviluppo e la visualizzazione delle analisi svolte sui dati del database Mensana.

3.1 Progettazione e casi di studio

L'oggetto di indagine sono le abitudini degli studenti dell'Università di Pisa in relazione al servizio di ristorazione gestito dal DSU Toscana[17].
La popolazione di riferimento è la totalità degli studenti presenti nel database e lo scopo dell'analisi è fornire informazioni utili agli utenti che avranno necessità di utilizzare i dati cercando di evidenziare tendenze che possono suggerire analisi non previste inizialmente, nuovi esperimenti o campionamenti.
Considerando la varietà di analisi possibili, si è scelto di suddividere l'indagine secondo due aspetti principali:

1. l'anagrafica degli studenti e il corso di studi da loro frequentato
2. le scelte effettuate dagli studenti in relazione ai pasti

Per quanto concerne il primo punto d'analisi si è ritenuto opportuno esaminare la frequenza della popolazione di riferimento in relazione ad alcune variabili e, in alcuni casi, suddividendo i soggetti in base al genere o alla fascia d'età per considerazioni più specifiche. Le variabili oggetto di studio sono:

- tipologia di corso frequentato e la macroarea disciplinare corrispondente
- mensa frequentata
- periodo di frequenza

Le tabelle del database interessate nell'analisi sono: *meal_receipt, relation, person, course, course_type e geo*. (vedi capitolo 1, paragrafo 1.3).

[17] v. nota 2

Inoltre, ciò che ci si propone di analizzare sono le scelte che gli studenti effettuano in relazione ai pasti basandosi sulla frequenza della scelta considerata. Le variabili osservate sono:

- composizione vassoio[18]
- la tariffa applicata al relativo pasto
- le scelte alimentari
    - il piatto più consumato dalla totalità degli studenti tra queste categorie: carne, pesce, cereali, ortaggi, uova, legumi, frutta
    - il piatto più consumato dagli studenti, suddivisi per genere, tra le categorie sopra elencate

Le tabelle del database interessate sono *meal_receipt, rate, dish, dish_bought, relation e person* descritte nel capitolo 1 al paragrafo 3.

I paragrafi seguenti illustrano le tecnologie utilizzate per sviluppare le analisi mentre i paragrafi 3 e 4 descrivono il risultato delle analisi reso graficamente mediante dei grafici.

3.2 Tecnologie utilizzate

3.2.1 XAMPP

Nello sviluppare applicazioni web è conveniente lavorare in locale[19] e trasferire il progetto sul server remoto solo una volta che si è ottimizzata la configurazione per lo scopo che ci si è prefissati. Ciò permette di avere una maggiore velocità, poiché non è necessario operare sul server trasferendo i file ogni volta che si apporta qualche modifica e perché non si hanno restrizioni imposte dai limiti di banda. Una volta terminato il progetto, si potrà effettuare l'*upload* del lavoro svolto su un server rendendolo disponibile online.

Per poter sviluppare in locale è necessario avere a disposizione un web server, un database e un linguaggio di script lato server.

[18] terminologia utilizzata dal DSU toscana per indicare le possibilità di composizione del piatto tra pasto completo, pasto ridotto con primo, pasto ridotto con secondo (http://www.dsu.toscana.it/servizi/ristorazione/dove-e-cosa-mangiare/)

[19] lo stesso computer svolgerà contemporaneamente sia il ruolo di server e sia di client

Per semplificare il processo di installazione e configurazione è possibile usufruire del pacchetto XAMPP (v. Figura 3.1) che contiene tutti i software precedenti già configurati.

XAMPP è una piattaforma software *cross-platform*, *open source* contenente Apache HTTP Server, il database MySQL e tutti gli strumenti necessari per utilizzare i linguaggi di programmazione PHP e Perl.

Per sviluppare l'applicazione si sono utilizzate le seguenti versioni:

- Versione del server: 10.1.2
- Versione PHP: 5.6.30
- Versione database: 5.5.8

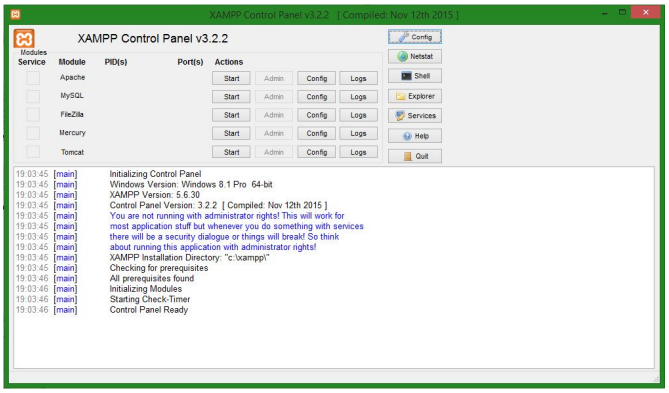


Figura 3.1 Pannello di controllo XAMPP

3.2.1.1 Apache HTTP server

Apache HTTP Server[20] è il web server gratuito più utilizzato in internet. Ha il compito di tradurre le richieste dei browser in pagine web e comprendere come elaborare il codice PHP, poiché quest'ultimo senza un web server non permetterebbe agli utenti web di raggiungere i documenti che contengono il suo codice. Presenta un'architettura modulare, quindi ad ogni richiesta del client vengono svolte funzioni specifiche da ogni modulo di cui è composto, come unità indipendenti. Infatti, ciascun modulo si occupa di una funzionalità, ed il controllo è gestito dal core, il programma principale composto da un ciclo sequenziale di chiamate ai moduli.

[20] https://it.wikipedia.org/wiki/Apache_HTTP_Server

3.2.1.2 PHP

PHP[21], acronimo ricorsivo di *Hypertext Preprocessor*, è un linguaggio di scripting lato server, con licenza open source, originariamente concepito per la realizzazione di pagine Web dinamiche. Attualmente è utilizzato principalmente per sviluppare applicazioni Web.

I programmi scritti in linguaggio PHP vengono eseguiti tramite un apposito software, l'*interprete* PHP. Quest'ultimo si occupa di leggere il codice PHP e, interpretandone le istruzioni, esegue le operazioni corrispondenti. Per questo motivo, il PHP è quello che tecnicamente si definisce un linguaggio *interpretato* ed, in questo, esso si differenzia da altri linguaggi di programmazione, come ad esempio C++ e Java, il cui codice sorgente, per poter essere eseguito, deve prima essere compilato, tradotto cioè in codice macchina. [10]

Altra importante caratteristica è che il codice PHP è immerso (*embedded*) nell'HTML poiché gli script possono essere inseriti all'interno delle pagine web. Le sezioni di codice per essere riconosciute devono essere incluse tra i tag `<?php` e `?>`. Il web server riconosce le pagine PHP, distinguendole da quelle statiche, sulla base dell'estensione *.php*. L'interprete effettua il parsing del file e sostituisce le sezioni di codice PHP con il codice HTML risultante dall'esecuzione.

Infine, dato che PHP opera lato server, prima la pagina sia spedita al browser lo script viene elaborato sul server. Di conseguenza, chi accede ad una pagina PHP non ha la possibilità di leggere le istruzioni in essa contenute. Essendo state processate, ciò che il client vedrà sarà solamente il risultato dell'elaborazione. [11]

[21]http://www.php.net/manual/en/

3.2.2 AJAX

Ajax, acronimo di *Asynchronous Javascript Xml*, è una "tecnica di sviluppo software per la realizzazione di applicazioni web interattive"[22]. Il fondamento che sta alla base della tecnologia Ajax è l'oggetto Javascript `XMLHttpRequest` il quale definisce un API[23] che assegna funzionalità al client per il trasferimento di dati tra client e server. La caratteristica principale di questo oggetto è di essere in grado di effettuare delle richieste *asincrone* ad un server HTTP, permettendo all'utente di svolgere diversi compiti simultaneamente all'interno di una stessa pagina web senza dover attendere la risposta dal server e i tempi di ricaricamento della pagina. [11]
Nell'applicazione web Mensana la chiamata AJAX è stata effettuata tramite jQuery[24] attraverso il metodo `$.ajax()`. Per configurare il metodo è necessario definire i seguenti parametri:

- url: indirizzo a cui inviare la richiesta per estrapolare i dati della risposta;
- type: tipo di richiesta. È possibile assegnare due valori GET o POST;
- data: contenuto della richiesta;
- dataType: tipologia di dati ottenuti come risposta;
- success: funzione richiamata quando la richiesta ha avuto successo;
- error: funzione richiamata quando si è verificato un errore nella richiesta al server.

Di seguito, è riportato un esempio generico di chiamata AJAX con i parametri sopra citati.

[22] https://it.wikipedia.org/wiki/AJAX
[23] Application Programming Interface(API) indica l'insieme di procedure disponibili al programmatore, di solito raggruppate a formare un set di strumenti specifici, per compiere un determinato compito all'interno di un certo programma.
[24] (v. cap. 2, par. 1, sez. 4)

```
$.ajax({
        url: nome_file,
        type: metodo,
        data: dati,
        dataType: tipo_di_dati,
        success: function(result){
                (…)
        },
        error: function(){
                        (..)
        }
});
```

3.2.3 JSON

JSON[25] (*JavaScript Object Notation*) è un semplice formato per lo scambio di dati. Per gli utenti è facile da leggere e scrivere, mentre per le macchine risulta facile da generare e analizzarne la sintassi.

JSON è basato su due strutture di dati universali che vengono supportate da tutti i linguaggi di programmazione e di seguito specificate:

- oggetto, rappresentato in figura 3.2, è una serie non ordinata di nomi/valori. Inizia con { (parentesi graffa sinistra) e finisce con } (parentesi graffa destra). Ogni nome è seguito da : (due punti) e le coppie di nome/valore sono separate da , (virgola).

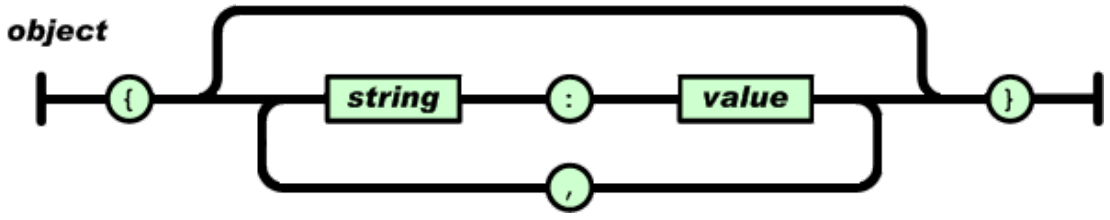


Figura 3.2 Struttura oggetto JSON

- array, rappresentato in figura 3.3, è una raccolta ordinata di valori che comincia con [(parentesi quadra sinistra) e finisce con] (parentesi quadra destra). I valori sono separati da , (virgola). [12]

[25] http://www.json.org/json-it.html

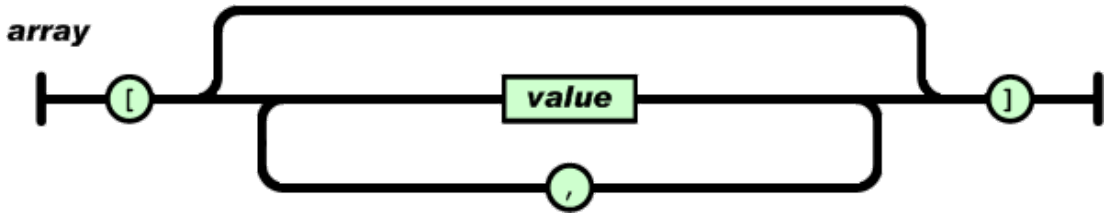


Figura 3.3 Struttura array JSON

3.2.4 Highcharts

Highcharts[26] è una libreria Javascript che permette di creare grafici interattivi per applicazioni web. E' open source, supporta il caricamento dei dati in modo dinamico dal server, permette di esportare i grafici nel formato PDF, PNG, JPG, SVG ed è compatibile su tutti i principali browser e piattaforme mobili.

Highcharts è stato incluso nell'applicazione web inserendo nella sezione <head> del file HTML uno script contenente il link d'accesso alla Content Delivery Network[27] (CDN).

3.3 Sviluppo delle analisi

3.3.1 Accesso ai dati del database

Per utilizzare i dati contenuti all'interno del database è necessario accedervi. Pertanto l'applicazione deve poter comunicare con il RDBMS attraverso una procedura iniziale, la connessione, che avviene tra lo script e il programma che gestisce la base di dati.

L'applicazione web Mensana utilizza *mysqli* procedurale come API[28] di PHP. La lettera *i* del termine *mysqli* sta per *improved* ovvero "migliorata". La nuova estensione è stata introdotta nella *release* 4.1.2 di PHP ed è stata realizzata per

[26] www.highcharts.com

[27] infrastruttura fisica che permette di ottimizzare il processo di fruizione dei contenuti attraverso la memorizzazione del contenuto stesso alla periferia della rete e in prossimità del cliente (https://it.wikipedia.org/wiki/Content_Delivery_Network)

[28] v. nota 22

mettere a disposizione degli sviluppatori delle funzionalità più avanzate rispetto a quelle di MySQL.

La connessione effettuata utilizzando la programmazione procedurale si basa su una funzione specifica denominata `mysqli_connect()` che accetta i seguenti parametri:

- hostname: il nome di rete della macchina ospitante il database server
- username: il nome dell'utente da utilizzare per la connessione
- password: la parola chiave necessaria per l'autenticazione dell'utente
- database: il nome del database che si desidera interfacciare alla propria applicazione.

Inoltre, è possibile controllare l'eventuale verificarsi di errori in fase di connessione tramite l'utilizzo della funzione `mysqli_connect_error()`. Nel caso in cui si verifichino problemi, la funzione restituirà una stringa descrittiva dell'errore generato.

Nel caso dell'applicazione Mensana, `mysqli_connect()` e il metodo `mysqli_connect_error()` sono inseriti all'interno della funzione `openDB()` collocata nel file *libreria.php*. In questo file vengono specificate tutte le funzioni PHP che, in base alla necessità, verranno richiamate nei vari file PHP dell'applicazione web. In questo caso, la funzione `openDB()` è invocata nel file *config.php* nel quale vengono indicati i valori dei parametri della funzione.

3.3.2 Selezione dati utili

Se l'esito della connessione è positivo, è possibile interrogare il database per estrarre i dati.

L'interrogazione MySQL è stata effettuata con l'utilizzo di funzioni PHP apposite quali `mysqli_query()` che permette di eseguire l'interrogazione sul database, `mysqli_fetch_assoc()` che memorizza i dati in un array associativo e `mysqli_free_result()` che libera la memoria associata al risultato. Esse sono inserite nella funzione `select()` del file *libreria.php* per evitare di dover riscrivere il codice ogni volta che vi è la necessità di eseguire

le operazioni. Infatti è sufficiente invocare la funzione, fornendole i parametri necessari per la sua esecuzione. In questo caso, la funzione `select()` ha due parametri formali, il valore assegnato alla variabile della connessione e i valori assegnati alla variabile dell'interrogazione. Quando la funzione è invocata all'interno del file avviene il passaggio dei parametri effettivi relativi ai valori assegnati nel file stesso.

3.3.3 Conversione dei dati in formato JSON

Dopo aver selezionato i dati, i risultati sono restituiti in un array associativo PHP e poi convertiti in formato JSON come segue. Nel codice PHP è inserito il MIME[29] `'Content-Type:application/json'` inoltre è utilizzata la funzione `json_encode()` alla quale sono passati come parametri la variabile PHP contenente l'array (`$response`) e la costante `JSON_NUMERIC_CHECK` che permette la codifica numerica. Di seguito, è riportato come esempio il file "variazione_pasti.php".

```
<?php
      header('Content-Type: application/json');
      require('config.php');

      $response = array();
      $query = "SELECT COUNT(id_receipt) as numero_pasti, YEAR(meal_date) AS anno,
      id_canteen AS id_mensa FROM `meal_receipt` GROUP BY anno, id_mensa"
      $response = select($mysqli, $query);

      echo json_encode($response, JSON_NUMERIC_CHECK);
?>
```

3.4 Realizzazione grafica

Comprendere da una tabella tutte le informazioni in essa contenute non è un intuitivo, pertanto per agevolare l'utente sono state realizzate rappresentazioni grafiche dei dati che, grazie al loro impatto visivo, sono di semplice lettura e di facile memorizzazione. Nelle sezioni di questo paragrafo verranno illustrati i concetti generali della data visualization e l'effettiva implementazione dei grafici con l'uso della libreria Highcharts.

[29] Sistema di comunicazione pensato per la trasmissione di dati binari che permette, grazie all'intestazione, la descrizione del tipo di contenuto del messaggio e della codifica usata.

3.4.1 Data Visualization

La Data Visualization è definita come l'esplorazione visuale/interattiva[30], tramite la relativa rappresentazione grafica, di dati di qualunque dimensione, natura e origine.
Grazie ad un approccio semplice ed intuitivo, permette di identificare fenomeni e trend che risultano invisibili ad una prima analisi dei dati.

La Data Visualization si basa su tre aspetti:

1. visibilità: i dati vengono rappresentati in un formato grafico sfruttando le notevoli capacità della percezione visiva, in grado di recepire ed analizzare, in una immagine, una notevole quantità di dati.
2. comprensione: permette un'immediata comprensione dei dati e fornisce consigli e suggerimenti sulle possibili azioni da intraprendere.
3. condivisione: l'immediata comprensione dei dati e delle analisi permette una rapida condivisione dei report con una conseguente e positiva ricaduta sui processi decisionali futuri.[13]

E' necessario tenere presente che i dati sono registrazioni di descrizioni di una qualsiasi caratteristica della realtà e, di conseguenza, non sono uno strumento su cui basare un processo comunicativo. Tuttavia, se elaborati, organizzati e inseriti in un contesto interpretativo possono determinare variazioni nella conoscenza di un soggetto, ossia possono diventare informazioni. [14]

[30] rappresentazioni che possono essere manipolate in tempo reale dall'utente, cambiando il punto di vista e/o i dati su cui la visualizzazione è costruita.

3.4.2 I grafici di Mensana

Nel presentare le informazioni, risulta di fondamentale importanza la scelta della tipologia di grafico. Essa si basa sul tipo di dato che si intende analizzare e sull'identificazione del target di utenti. Nell'applicazione web Mensana si è optato per diverse tipologie di grafici quali a barre, a torta e a linee. Di seguito è descritta la loro realizzazione.

3.4.2.1 Implementazione

La visualizzazione dei dati è stata effettuata utilizzando la libreria Highcharts.
Come indicato nel capitolo 3 paragrafo 3 sezione 3, i risultati dell'array associativo sono in formato JSON.


```
(..)

{"numero_pasti":783831,"anno":2010,"id_mensa":1},{"numero_pasti":177046,"anno"
:2010,"id_mensa":2},{"numero_pasti":205035,"anno":2010,"id_mensa":7}

(..)
```


Figura 3.4 Parte dei risultati in formato JSON del file *variazione_pasti.php*

Pertanto per poter realizzare la resa grafica grafica, si utilizza la funzione jQuery `$.getJSON`, supportata da Highcharts, alla quale vengono passati due parametri:

- url: è una stringa contenente l'URL[31] a cui è inviata la richiesta;
- function(): è una funzione che viene eseguita solo se la richiesta ha avuto successo.

Di seguito è riportato un esempio di codice contenente la funzione `$.getJSON`:

[31] Uniform Resource Locator: è una sequenza di caratteri che identifica univocamente l'indirizzo di una risorsa in Internet

```
(..)
var url_variazione_pasti = "php/variazione_pasti.php"
    $.getJSON(url_variazione_pasti, function(result) {
        var martiri = Array();
      (..)
        $.each(result, function(index, item) {
            switch (item["id_mensa"]) {
                case 1:
                    martiri.push([item["anno"], item["numero_pasti"]]);
                    break;
                (..)
            }
        });
        (..)
    });
(..)
```

In questo caso, l'obiettivo è mostrare la totalità dei pasti erogati ogni anno nel periodo compreso dal 2010 al 2015 nelle varie mense.

Si dichiara un array per ogni mensa[32]. Il parametro *result* della funzione `$.getJSON` viene passato alla funzione `$.each`, una struttura di controllo iterativa. All'interno di `$.each` si esegue l'istruzione *switch* che controlla il valore dell'identificatore della mensa. Sono definite una serie di casistiche attraverso la parola riservata *case* seguita dall'ID della mensa presa in esame. All'interno di ogni blocco sono inserite le istruzioni che aggiungono i dati all'array della mensa corrispondente utilizzando il metodo `push()`, tuttavia un blocco può essere eseguito solo se vi è una corrispondenza tra l'espressione dello switch e il valore della variabile nel *case*. Al termine di ogni blocco è inserito il comando *break* al fine di interrompere il flusso dello switch ogni volta che si rientra nella casistica di interesse.

Ogni array contenente i distinti risultati in relazione all'id della mensa è assegnato alla variabile *series* di Highcharts con la seguente sintassi:

```
var series: [{
    name: 'nome array'
    data: array_mensa_corrispondente
}]
```

[32] A titolo esemplificativo nell'estratto di codice viene indicata soltanto la mensa Martiri.

La tipologia e gli elementi necessari del grafico sono definiti dichiarando altre variabili che, per poter essere utilizzate, sono combinate in informazioni di configurazione Highcharts. Infine il grafico viene inserito nell'elemento HTML con l'id corrispondente utilizzando questa sintassi `$('#id_elemento').highcharts(json);`. Di seguito è riportato l'estratto di codice con il procedimento sopra descritto e il risultante grafico (vedi Figura 3.5)

```
 (..)
 var series = [{name: 'Martiri',data: martiri}, (..)]
 var chart = {type: 'column'};
 var title = {text: 'Pasti erogati dal 2010 al 2015'};
 var subtitle = {text: 'Tutti gli studenti'};
 var xAxis = {categories: ['2010', '2011', '2012', '2013', '2014',
'2015']};
 var yAxis = {title: {text: 'Numero di scontrini'}};
 var tooltip = {valueSuffix: ' pasti erogati'}

 var json = {};
 json.chart = chart;
 json.title = title;
 json.subtitle = subtitle;
 json.xAxis = xAxis;
 json.yAxis = yAxis;
 json.tooltip = tooltip;
 json.series = series;

 $('#variazione_pasti').highcharts(json);
});
```

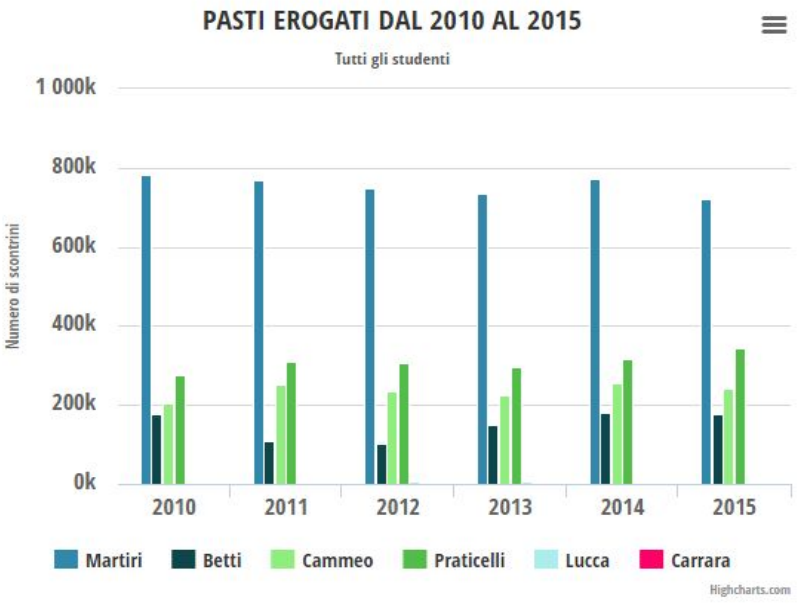


Figura 3.5 Grafico pasti erogati

Lo stesso procedimento è ripetuto per visualizzare tutti i grafici dell'applicazione web. Tuttavia, a seconda del grafico, la funzione `$.getJSON()` viene modificata. Per esempio, nella visualizzazione della frequenza degli studenti suddivisi per genere nel periodo compreso tra il 2010 e il 2015, si inizializza la variabile *i* a zero e si utilizzano due funzioni `$.each()` annidate per eseguire un controllo sulla chiave e assegnare all'array il valore che, in questo caso, è il genere dello studente. Alla fine del ciclo della seconda funzione `$.each()`, la variabile *i* viene incrementata di un'unità. Di seguito è riportato l'estratto di codice.

```
[..]
$.getJSON(url_frequenza, function(result) {

        var dati_uomo = Array();
        var dati_donna = Array();
var i = 0;
        $.each(result, function(index, item) {
            $.each(item, function(key, value) {
                if (key == "sum_M") {
                    dati_uomo[i] = value;
                }
                if (key == "sum_F") {
                    dati_donna[i] = value;
                }
            });
            i++;
         });
[..]
```

3.4.2.2 Interattività

In ambito informatico, il termine interattività "corrisponde a interazioni uomo-macchina in grado di influenzare la performance di un programma[33]". In questo caso, si utilizza per indicare l'interazione tra l'utente e l'applicazione web. Grazie alla tecnologia AJAX[34], l'applicazione può aggiornare dinamicamente i grafici in base alle richieste dell'utente. L'aggiornamento si verifica quando l'utente seleziona un elemento nell'elenco *drop down* (a comparsa) collocato sopra al relativo grafico.

Al tag HTML `<select>` è associato l'evento `onchange()` che punta ad una funzione che il browser chiama ogni volta che l'utente modifica l'opzione

[33] *Atlante della comunicazione.* A cura di Fausto Colombo. Hoepli Editore, 2005. pp.168-169
[34] v. cap. 3, par. 2, sez. 2

nella lista. All'interno della funzione si effettua la chiamata AJAX assegnando GET come tipo di richiesta. Con il metodo GET i dati vengono passati direttamente all'interno dell'indirizzo web della pagina seguito da un punto interrogativo ("?") e dai dati organizzati in coppie *nome/valore*. Nel caso specifico, il nome è assegnato nel file PHP corrispondente mentre il valore è associato attraverso il metodo `val()` di jQuery all'opzione in quel momento scelta dall'utente.

Di seguito, è riportata la parte di codice utilizzata per rendere i grafici interattivi.

```
(..)
function chiamata_anni(){
   var anni= $('#anno_select').val();
   var valore_anni="php/affluenza_orario.php?anno=" +anni;
   $.ajax({
      url: valore_anni,
      type: "GET",
      dataType: "json",
      success: function(result){
            (..)
      },
      error: function(){
            alert("Errore. Si prega di riprovare.");
      }
   });
};
(..)
```

4. Esportazione tabelle del database

L'utente può visualizzare e scaricare i dati del database navigando nelle pagine "Studenti" e "Piatti" dell'applicazione web.

Nella pagina studenti sono coinvolte le tabelle meal_receipt, relation, person, course, course_type, geo mentre nella pagina piatti meal_receipt, meal_type, rate, canteen, dish, dish_bought.

Il recupero dei dati avviene interrogando il database mediante i linguaggi di programmazione PHP e SQL. L'interrogazione è passata come parametro effettivo della funzione PHP `mysqli_query()` che permette di effettuarla. Inoltre, si utilizzano le funzioni PHP `mysqli_num_fields()` per restituire il numero dei campi della tabella, `mysqli_fetch_field()` per recuperare le informazioni relative alle colonne del set di risultati e `mysqli_fetch_row()` per inserire i risultati in un array numerico (vedi Appendice codici capitolo 7 punto 5).

A seguito dell'estrazione dei dati, le relative tabelle HTML vengono generate dinamicamente eseguendo due cicli iterativi prima sugli attributi poi sui record della tabella stessa.

La visualizzazione dei risultati è effettuata utilizzando il plugin[35] Datatables[36] di jQuery che permette di aggiungere alla tabella funzionalità avanzate come la tecnica di *paging* secondo la quale i record che formano la vista principale dell'applicazione sono mostrati in pagine di dimensione uniforme e gli utenti possono navigare tra esse con l'utilizzo di appositi pulsanti. Inoltre, dispone di numerose altre peculiarità tra cui un campo *Cerca* con il quale è possibile effettuare una ricerca all'interno della tabella, l'inserimento di frecce al lato degli attributi per ordinare i record della colonna corrispondente e la possibilità da parte dell'utente di impostare la quantità di record della tabella da visualizzare scegliendo tra i valori 10, 25, 50 oppure 100.

[35] un programma non autonomo che interagisce con un altro programma per ampliare o estendere le funzionalità originarie

[36] https://datatables.net/manual/index

Datatables è incluso nella sezione `<head>` delle pagine HTML tramite il tag `<script>` con il link della CDN[37] di Javascript e un tag `<link>` contenente il link della CDN di CSS. I precedenti link consentono la configurazione di base, oltre a questi, sono inseriti altri tag HTML `<script>` per includere l'estensione *buttons.exportData()* che consente di scaricare la tabella nel formato CSV.

È possibile scegliere se scaricare la tabella completa o parziale in modo che l'utente possa disporre di un'opzione aggiuntiva basata sulle sue esigenze. Ciò si realizza cliccando sui filtri disponibili[38] e accedendo ad un menù a comparsa le cui voci possono essere selezionate o deselezionate. Inizialmente, ogni elemento è selezionato e viene mostrata la tabella completa. Se l'utente non necessita di tutti dati, è sufficiente deselezionare l'elemento a cui non è interessato per aggiornare la tabella senza quel determinato attributo. Ogni volta che la tabella viene modificata, è fornito un feedback all'utente del cambiamento verificatosi attraverso un'anteprima della tabella aggiornata.
L'applicazione aggiorna i dati utilizzando la chiamata AJAX (vedi capitolo 3, paragrafo 2, sezione 3) e assegnando come *type* alla funzione il valore `POST`.

```
[..]
$(document).ready(function(){
    $("input.attributo_tabella").click(function(){
    var check= [];
    $('.attributo_tabella:checked').each(function(i){
          check[i] = $(this).val();
        });
     var check2= check[0];
      $.ajax({
      url: "filtri.php",
      type: "POST",
      data: {check:check2},
      dataType: "html",
      success: function(result){
           $("#risultato").html(result);
       },
      error: function(){
           alert("Errore. Si prega di riprovare.");
      }
    });
[..]
```

[37] v. nota 25
[38] v. cap.2, par.2, sez.1

5. Conclusioni

Lo scopo dello sviluppo dell'applicazione web è rendere accessibili i dati riguardanti le mense dell'Università di Pisa affinché possano essere utilizzabili dagli utenti.

L'applicazione è quindi un supporto necessario per l'interazione tra le informazioni presenti nel database e l'esigenza dell'utente di acquisirle.

Una possibile chiave di lettura di queste informazioni è indicata all'utente nella pagina "Analisi" in modo da fornire una contestualizzazione dei dati di interesse.

Un altro aspetto implementato con attenzione è la realizzazione grafica dell'applicativo che si adatta agli standard Web di usabilità e accessibilità attraverso una struttura chiara e logica che consente ai contenuti di essere significativi, comprensibili e facilmente identificabili.

6. Bibliografia e Sitografia

[1] Articolo web progetto Mensana (https://www.unipi.it/index.php/news/item/8535-una app-aiuta-gli-studenti-a-mangiare-meglio), data di consultazione 20/02/2017

[2] Atzeni Paolo, Stefano Ceri, Stefano Paraboschi, Riccardo Torlone. *Basi Di Dati: Modelli E Linguaggi Di Interrogazione*, Milano, McGraw-Hill, 2013.

[3] Sumathi, S. Esakkirajan. *Fundamentals of Relational Database Management Systems.* Springer, 2011.

[4] Jeremy Keith, Rachel Andrew. *HTML5 for Web Designers*. New York, A Book Apart, 2015.

[5] Jon Duckett, Gilles Ruppert, Jack Moore. *JavaScript & JQuery: Interactive Front-end Web Development*, Indianapolis, Wiley, 2014.

[6] Linee Guida di design per i servizi web della pubblica amministrazione. (http://design.italia.it/), data di consultazione 25/02/2017

[7] Wendy Chisholm, Matt May. *Universal Design for Web Applications That Reach Everyone*, Sebastopol, O'Reilly Media, 2008.

[8] W3School (https://www.w3schools.com/), data di consultazione 10/04/2017

[9] Teoria del colore nell'arte e nel design (http://www.cultorweb.com/CC) data di consultazione 05/05/2017.

[10] David Powers. *PHP Solutions: Dynamic Web Design Made Easy*. Berkeley, Apress, Terza edizione, 2014.

[11] Cristian Darie, Bogdan Brinzarea. *Ajax E PHP: Sviluppare Applicazioni Web Dinamiche*, Milano, Hoepli, 2007.

[12] Strutture dati JSON (http://www.json.org/), data di consultazione 06/05/2017

[13] Data Visualization (https://www.sas.com/it_it/offers/sas-data-visualization-ebook/), data di consultazione 10/05/2017

[14] Andy Kirk. *Data Visualization: A Successful Design Process*. Packt Publishing, 2012.

[15] Sito web Mensana-Rasupea (https://mensana-rasupea.unipi.it/), data di consultazione 10/02/2017

7. Appendice codici

1. Associazione identificatore piatto-ingrediente

```
import os
import re

outfile = open('output_parziale.csv', 'w')
outfile2 = open('output_finale.csv', 'w')

outfile1 = open('output.csv', 'r')
nome_id = dict()
file_ricettario= open('ricettario.csv')
for line in file_ricettario:
    if re.match('[A-Z]', lista1[1]):
        lista1[0]=lista1[1]
        lista1[1]=''
        nome_id[lista1[0]]=lista1[2]

     outstring = ",".join([lista1[0], "", str(lista1[2]),
     lista1[1],lista1[3], lista1[4]])+"\n"
    outfile.write(outstring)

for line in outfile1:
    if lista[0] is not '':
        nome_ricetta=lista[0]
    elif lista[0] is '':
        lista[0]=nome_ricetta
    for nome in nome_id:
          if nome in lista[0]:
                id_x=str(nome_id[nome])
                lista[1]=id_x
    if len(lista)==6:
          outstring2 = ",".join([lista[0], str(lista[1]), lista[3],
          lista[4],str(lista[2]), str(lista[5])])+"\n"
          outfile2.write(outstring2)
```

2. Corrispondenza alimento-ingrediente composizione chimica e valore energetico

```
import pandas as pd

import codecs

import os

import json

ricettario=pd.read_csv('ricettario_tabellare2.csv',   delimiter=',',
encoding='utf-8')

ricettario2=ricettario.dropna(subset=['ingredient'])
```

```
ricettario3=ricettario2[['dish_name','ingredient','characteristic']
]

nutrizioni=pd.read_excel('DB_Alimenti_BodyBita.xlsx',
encoding='utf-8')

nutrizioni=nutrizioni[['ALIMENTO']]

ricettario4=ricettario3[0:4256]

matching_file_lettura= codecs.open('desktop/matching.csv', 'r')

f = codecs.open('desktop/matching.csv', 'w')

with open('desktop/ingredienti_trovati.json', 'r') as
file_da_leggere:

     dizionario_ingredienti_trovati = json.load(file_da_leggere)

for x in range(len(ricettario4)):

    piatto=ricettario4.iloc[x,0]

    ingrediente=list((ricettario4.iloc[x,1]).upper().replace("'",'
    ').replace("(",'').replace(")",'').replace("[",'').replace("]",'
    ').replace(",",' ').split(' '))

    ingrediente_codifica=[str(' '.join(filter(None, ingrediente)))]

    caratteristica=ricettario4.iloc[x,2]

    lista=[]

    for y in range(len(nutrizioni))

         item=list(nutrizioni.iloc[y,0].replace("'",'
         ').replace("(",'').replace(")",'').replace("[",'').replace(
         "]",'').replace(",",' ').split(' '))

         item_codifica=[str(' '.join(filter(None, item)))]

         s=len(set(ingrediente).intersection(item))

         if s>0:

             d=(str(piatto),ingrediente_codifica, s,item_codifica)

             lista.append(d)

    sorted_list = sorted(lista, key=lambda x: x[2], reverse=True)

    top=sorted_list[0:15]

    for ingrediente_finale in ingrediente_codifica:

       if ingrediente_finale not in
       dizionario_ingredienti_trovati.keys():

            print('   ')

            print('Ingrediente: ',str(ricettario4.iloc[x,1]),' -
            Caratteristica: ',caratteristica)
```

```
            for k in top:

                print(k)

            valore = raw_input('Inserisci la scelta (N:B. NOT FOUND
            se non trovi il match):')

            f.write('%s,%s,%s\n' %(piatto,ricettario4.iloc[x,1],
            valore))


            dizionario_ingredienti_trovati[ingrediente_finale] =
            valore

        else:

            valore =
            dizionario_ingredienti_trovati[ingrediente_finale]

            f.write('%s,%s,%s\n' %(piatto,ricettario4.iloc[x,1],
            valore))

        file_json =
        codecs.open('desktop/tirocinio/ingredienti_trovati.json',
        'wb', 'utf-8')

        json.dump(dizionario_ingredienti_trovati, file_json)

file_json.close()

f.close()
```

3. File “libreria.php”

```
<?php
function openDB($servername, $username, $password, $database){
    $conn = mysqli_connect($servername, $username, $password,
    $database);
    mysqli_query($conn, "SET character_set_results = 'utf8',
    character_set_client = 'utf8', character_set_connection =
    'utf8', character_set_database = 'utf8', character_set_server =
    'utf8'");
    if(!$conn) die("Connection failed: ". mysqli_connect_error());
          return $conn;
}

function select($conn,$sql){
    $resultSet = mysqli_query($conn, $sql);
    if(!$resultSet){
          print("Errore nella query $sql: ".mysqli_error($conn));
          return(NULL);
    }
    while ($record = mysqli_fetch_assoc($resultSet))
$records[]=$record;
    mysqli_free_result($resultSet);
```

```
    return $records;
}

function closeDB($conn){
    mysqli_close($conn);
}

?>
```

4. File “config.php”

```
<?php
require ("libreria.php");
$mysqli = openDB("localhost", "root", NULL, "projects");
?>
```

5. File “filtri.php”

```
<?php
require('config.php')
if (isset($_POST["check"]) && $_POST["check"] == "completa") {
     $sql = "SELECT meal_receipt.id_receipt,
     meal_receipt.meal_date, meal_receipt.id_stu,
     course.course_description, course_type.type_description,
     geo.birth_town, geo.region, geo.country FROM meal_receipt JOIN
     relation ON meal_receipt.id_stu=relation.id_stu JOIN course ON
     relation.id_course=course.id_course JOIN course_type ON
     course.id_type=course_type.id_type JOIN person ON
     relation.id_pers=person.id_pers JOIN geo ON
     person.id_birth_town=geo.id_birth_town";
} elseif (isset($_POST["check"]) && $_POST["check"] == "studente")
{
     $sql = "SELECT relation.id_pers, relation.enroll_year,
     course.course_description, course_type.type_description FROM
     relation JOIN course ON relation.id_course=course.id_course
     JOIN course_type ON course.id_type=course_type.id_type";
} elseif (isset($_POST["check"]) && $_POST["check"] == "individuo")
{
     $sql = "SELECT person.id_pers, person.sex, person.birth_year,
     geo.birth_town, geo.region, geo.country FROM person JOIN geo
     ON person.id_birth_town=geo.id_birth_town";
} elseif (isset($_POST["check"]) && $_POST["check"] ==
"completa_piatti") {
    $sql = "SELECT meal_receipt1.id_receipt1, meal_receipt1.id_stu,
    canteen.canteen_name, meal_receipt.meal_date,
    meal_type.meal_type, rate.rate_description, dish_bought.id_dish,
    dish.dish_description, dish.tag_vegetarian,
    dish.dish_composition FROM meal_receipt1 JOIN meal_type ON
    meal_receipt1.id_meal_type=meal_type.id_meal_type JOIN rate ON
    meal_receipt1.id_rate=rate.id_rate JOIN canteen ON
    meal_receipt1.id_canteen=canteen.id_canteen JOIN dish_bought ON
```

```
    meal_receipt1.id_receipt=dish_bought.id_receipt JOIN dish ON
    dish_bought.id_dish=dish.id_dish";
} elseif (isset($_POST["check"]) && $_POST["check"] == "scontrino")
{
    $sql = "SELECT meal_receipt1.id_receipt, meal_receipt1.id_stu,
    canteen.canteen_name, meal_type.meal_type,
    meal_receipt1.meal_date, rate.rate_description FROM
    meal_receipt1 JOIN meal_type ON
    meal_receipt1.id_meal_type=meal_type.id_meal_type JOIN rate ON
    meal_receipt1.id_rate=rate.id_rate JOIN canteen ON
    meal_receipt1.id_canteen=canteen.id_canteen";
} elseif (isset($_POST["check"]) && $_POST["check"] == "pasto") {
    $sql = "SELECT dish_bought.id_dish, dish.dish_description,
    canteen.canteen_name, dish.tag_vegetarian, dish.dish_composition
    FROM meal_receipt1 JOIN dish_bought ON
    meal_receipt1.id_receipt=dish_bought.id_receipt JOIN dish ON
    dish_bought.id_dish=dish.id_dish JOIN canteen ON
    meal_receipt1.id_canteen=canteen.id_canteen";
} else {
    echo "";
    die();
}
$result = mysqli_query($mysqli, $sql) or die(mysqli_error());
echo "<table id='tabella_risultato' class='display table
table-cell-border table-hover' cellspacing='0'><thead>";
for ($i = 0; $i < mysqli_num_fields($result); $i++) {
    $field_info = mysqli_fetch_field($result);
    echo "<th class='colonna_" . $_POST["check"] . "_" . $i .
"'>{$field_info->name}</th>";
}
echo "</thead>";
echo "<tbody>";
while ($row = mysqli_fetch_row($result)) {
    echo "<tr>";
    $y = 0;
    foreach ($row as $_column) {
         echo "<td class='colonna_" . $_POST["check"] . "_" . $y .
        "'>{$_column}</td>";
       $y++;}
    echo "</tr>";
}
echo "</tbody>";
echo "</table>";
?>
```