

Documentazione progetto PizzaDelivery

1 Introduzione.....	4
1.1 Informazioni sul progetto.....	4
1.2 Abstract.....	4
1.3 Scopo.....	4
1.4 Analisi del dominio.....	4
1.5 Analisi e specifica dei requisiti.....	5
1.6 Use case.....	7
1.7 Analisi dei mezzi.....	8
1.7.1 Software.....	9
1.7.2 Hardware.....	9
2 Progettazione.....	10
2.1 Design dell'architettura del sistema.....	10
2.2 Design dei dati e database.....	11
2.3 Design delle interfacce.....	12
2.3.1 Pagina Principale.....	12
2.3.2 Pagina Login.....	13
2.3.3 Pagina Amministratore.....	13
2.3.4 Pagina Ordinazione.....	14
2.3.5 Pagina Fattorini.....	14
2.3.6 Pagina Ordinazioni.....	15
2.3.7 Pagina Consegne.....	15
3 Implementazione.....	16
3.1 Struttura dell'applicativo.....	16
3.1.1 Routing URL.....	16
3.1.2 Gerarchia delle cartelle.....	17
3.1.3 File di configurazione.....	18
3.1.4 Classi Controller.....	19
3.1.5 Cartella libs.....	20
3.1.6 Classi Model.....	21
3.1.7 Scripts.....	22
3.1.8 Views.....	23
3.1.9 Classe Database.....	23
3.2 Database.....	24
3.3 Sicurezza e ruoli dell'applicativo.....	25
3.4 Codice JavaScript lato client.....	26
3.5 Pagina Gestione Pizzeria.....	28
3.6 Implementazione Mappe.....	30
3.7 Approccio Mobile First.....	31
3.8 Installazione debugger.....	32
3.8.1 Download.....	32
3.8.2 Configurazione php.ini.....	32
3.8.3 Configurazione phpStorm.....	33
4 Test.....	34
4.1 Protocollo di test.....	34
4.2 Risultati test.....	38
4.3 Mancanze/limitazioni conosciute.....	38
5 Installazione.....	39
5.1 Requisiti iniziali.....	39
5.2 Copiatura della cartella PizzaDelivery sulla propria macchina.....	39
5.3 Creazione DataBase e inserimento dei dati essenziali.....	39
5.4 Impostazione subfolder della cartella webroot.....	40
5.5 Modifica percorso URL del progetto.....	40
5.6 Configurazione accesso MySQL per interagire con il DataBase.....	41
5.7 Primo accesso tramite interfaccia grafica.....	41
Consuntivo.....	42
6 Conclusioni.....	43

6.1 Sviluppi futuri.....	43
6.2 Considerazioni personali.....	43
7 Bibliografia.....	44
7.1 Sitografia.....	44
8 Allegati.....	44

1 Introduzione

1.1 Informazioni sul progetto

Allievo coinvolto: Jari Näser

Classe: Informatica 4AA Presso la Scuola Arti e Mestieri di Trevano

Docenti responsabili: Ivan Raimondi

Data inizio: 3-09-2019

Data fine: 20-12-2019

1.2 Abstract

With today's world wide food demand being higher than ever and people's laziness as well the idea of this project is to unify these two requests creating a service that brings one of the world's most requested dishes to people's home.

With PizzaDelivery it's now possible for Pizza Stores to offer a really powerful service to it's customers by bringing them their ordered food wherever they like in a quick and cheap way.

This software is capable to manage in a really simple and yet efficient way the serving task and all the challenges that come with it, it's possible to manage all the incoming and outgoing orders, all of your employees such as delivery men, their status and position.

1.3 Scopo

Lo scopo di questo progetto è di realizzare un sistema informatico che semplifichi la gestione delle ordinazioni online accessibile a chiunque, in questo caso per le pizzerie la consegna a domicilio.

Attraverso una piattaforma dinamica e programmata mantenendo un approccio di tipo mobile first è possibile eseguire comande di qualsiasi tipo attraverso l'applicativo Web.

Il programma offre anche svariate pagine per la gestione di tutta l'infrastruttura interna della pizzeria stessa accessibile agli impiegati come le ordinazioni, consegne, i fattorini ed una pagina amministrativa per quanto riguarda la gestione degli utenti ed articoli registrati nel sistema.

1.4 Analisi del dominio

Attualmente non esistono ancora programmi con le funzionalità richieste dal cliente.

Lo scopo è quello di offrire un servizio a basso costo e molto semplice da usare per varie pizzerie che vorrebbero entrare nel mercato delle consegne a domicilio per espandersi in modo molto rapido ed efficiente, cosa che attualmente non è ancora possibile.

1.5 Analisi e specifica dei requisiti

Il committente richiede un sistema che sia capace di gestire il servizio online di consegne a domicilio di una pizzeria attraverso un portale web.

Esso è principalmente composto dalle 5 pagine: principale, ordinazioni, consegne, fattorini e gestione pizzeria che interagiscono continuamente con il database PizzaDelivery.

Infine è possibile utilizzare il programma come cliente normale che esegue degli ordini oppure come impiegato, fattorino oppure amministratore facendo il login con il proprio account potendo accedere a funzionalità e pagine aggiuntive rispetto ai normali clienti.

ID: REQ-01	
Nome	Installazione ambiente di sviluppo
Priorità	1
Versione	1.0
Note	Si necessita di una licenza JetBrains
Sotto requisiti	
001	Si necessita di scaricare ed installare ...AMP (XAMP, MAMP o LAMP a dipendenza del sistema operativo)
002	Si necessita di scaricare ed installare l'IDE PhpStorm della JetBrains
003	Si necessita di scaricare ed installare MySQLWorkbench per la gestione semplificata del database.

ID: REQ-02	
Nome	Progettazione database
Priorità	1
Versione	1.0
Sotto requisiti	
001	Si necessita di essere in chiaro sui vari ruoli dell'applicazione

ID: REQ-03	
Nome	Creazione database
Priorità	1
Versione	1.0
Note	Dipende dal REQ-02 (Progettazione database)
Sotto requisiti	
001	È necessario un diagramma ER da seguire

ID: REQ-04	
Nome	Implementazione modello MVC
Priorità	1
Versione	1.0
Sotto requisiti	
001	Bisogna impostare la root del progetto corretta nelle sue configurazioni per evitare errori nel routing

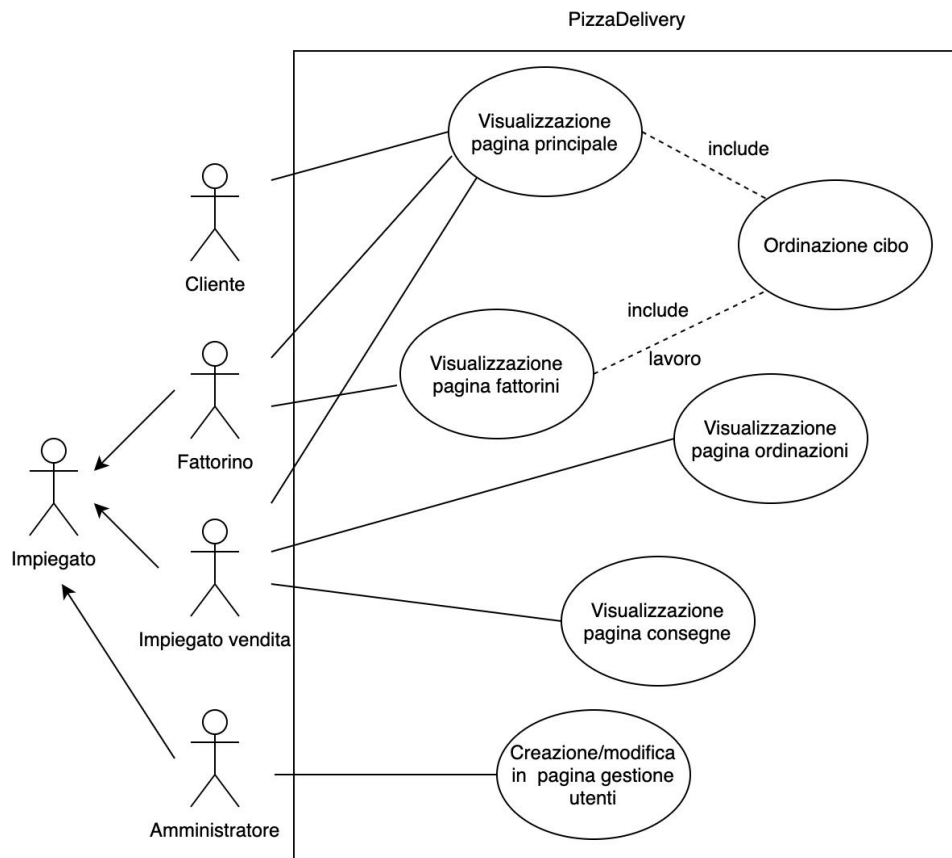
ID: REQ-05	
Nome	Creazione pagina ordinazioni
Priorità	1
Versione	1.0
Note	Dipende dal REQ-04 (implementazione modello MVC)
Sotto requisiti	
001	Si necessita di una tabella nel database che memorizzi tutte le ordinazioni
002	Possibilità di ricavare dati attraverso query dal database
003	Si deve gestire il login dei vari utenti

ID: REQ-06	
Nome	Creazione pagina consegne
Priorità	1
Versione	1.0
Note	Dipende dal REQ-04 (implementazione modello MVC)
Sotto requisiti	
001	Si necessita di una tabella nel database che memorizzi tutte le consegne
002	Possibilità di ricavare le consegne precedenti e attuali attraverso query dal database
003	Si deve gestire il login dei vari utenti

ID: REQ-07	
Nome	Creazione pagina fattorini
Priorità	1
Versione	1.0
Note	Dipende dal REQ-04 (implementazione modello MVC)
Sotto requisiti	
001	Si necessita di una tabella nel database che memorizzi tutti i fattorini e le loro azioni
002	Possibilità di ricavare i dati attraverso query dal database dei fattorini e delle loro azioni giornaliere
003	Si deve gestire il login dei vari utenti

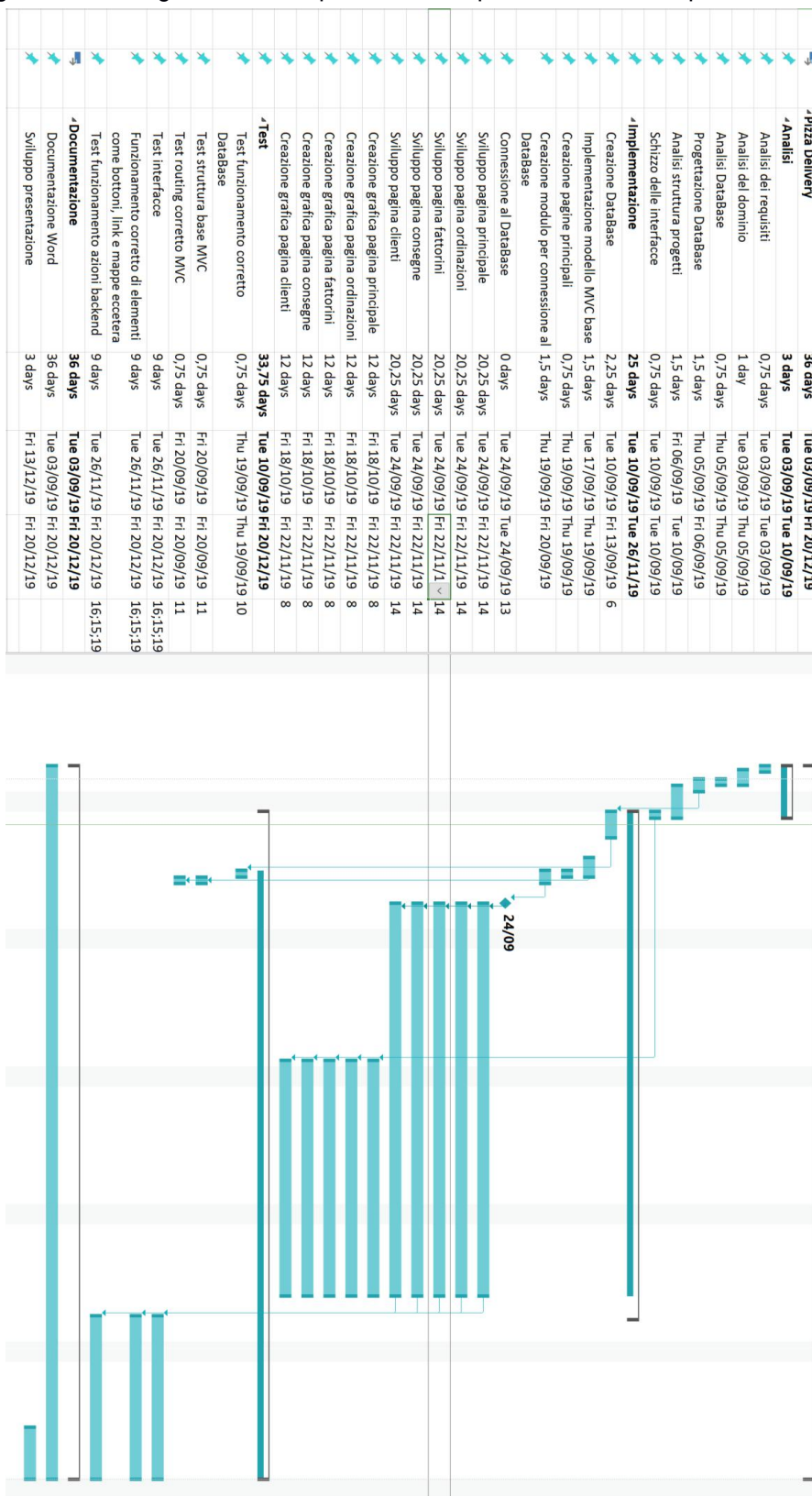
1.6 Use case

Le iterazioni tra i vari tipi di utente ed il programma avvengono nel seguente modo:



1.7 Analisi dei mezzi

Per quanto riguarda la suddivisione dei tempi e quindi la progettazione delle tempistiche di questo progetto ho pensato di gestirle nel seguente modo per sfruttare queste 174 ore disponibili nel miglior dei modi possibili.



1.7.1 Software

Per la realizzazione di questo progetto ho utilizzato i seguenti Software:

- PHPStorm 2017.2.4: IDE principale per la scrittura di tutto il programma.
- MAMP 4.5: Programma che funge da WebServer offrendo i servizi di Apache, MySQL e PHP.
- Google Chrome 79.0.3945.79: Browser utilizzato per testare lo sviluppo dell'applicativo.
- MySQLWorkBench 8.0.17: Software usato per lo sviluppo del codice SQL per il DataBase.
- Sublime Text 3.2.1: Editore di testo per aprire vari file ed eseguire piccole modifiche.
- GitHub Desktop 2.1.3: Software per accedere alla repository di GitHub tramite interfaccia grafica.
- Draw.io 11.2.8: Software usato per quasi tutti i diagrammi e i design delle interfacce.
- Project 2016: Programma utilizzato per effettuare il diagramma di Gantt.
- Pages 8.2: Software per scrivere i diari.
- WPS Office 1.5.0: Software per scrivere i diari e la documentazione.

1.7.2 Hardware

Questo progetto è stato sviluppato unicamente sul mio portatile personale.

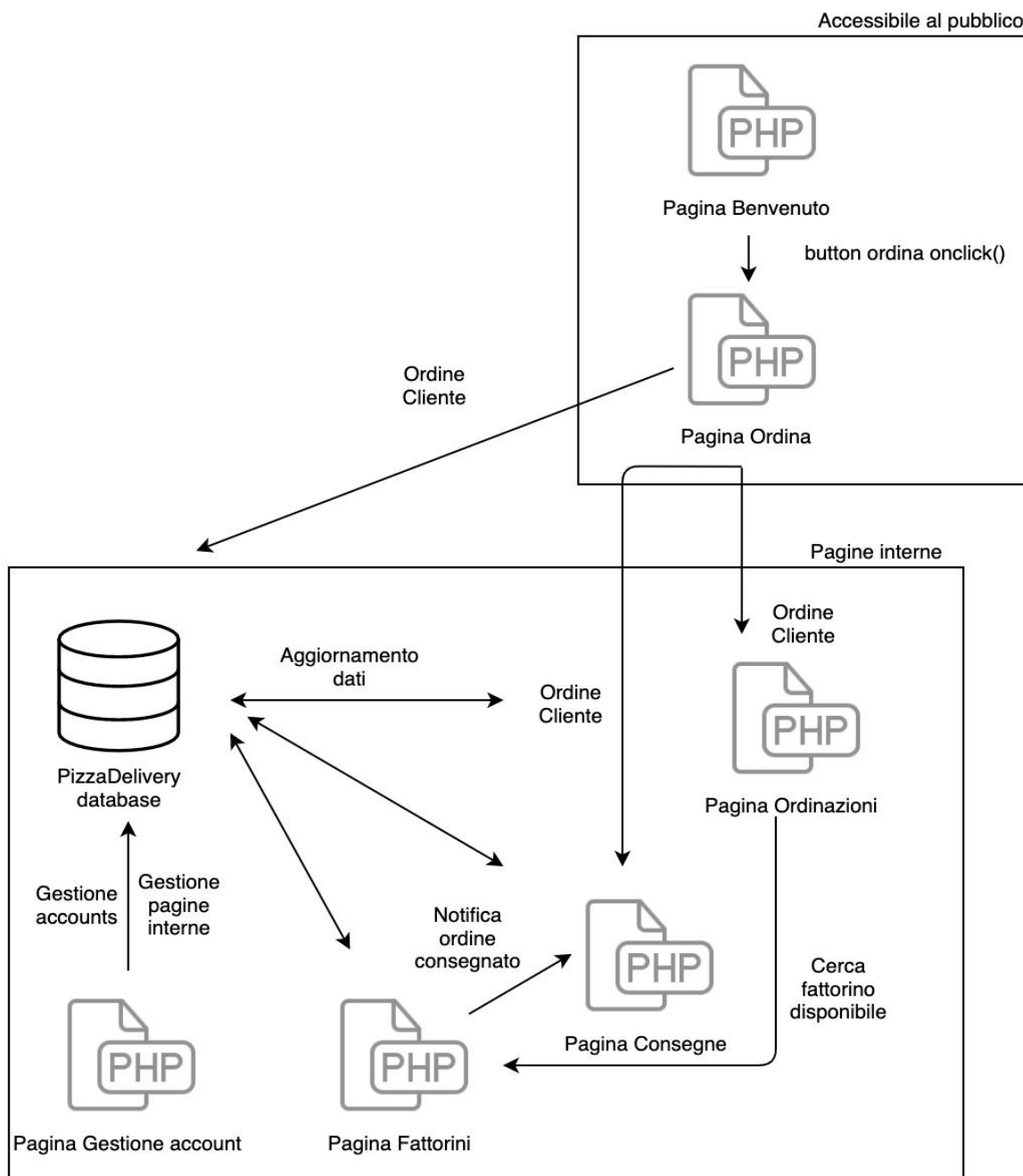
Modello: Apple Mac Book Pro mid 2015

OS: Catalina 10.15.1

2 Progettazione

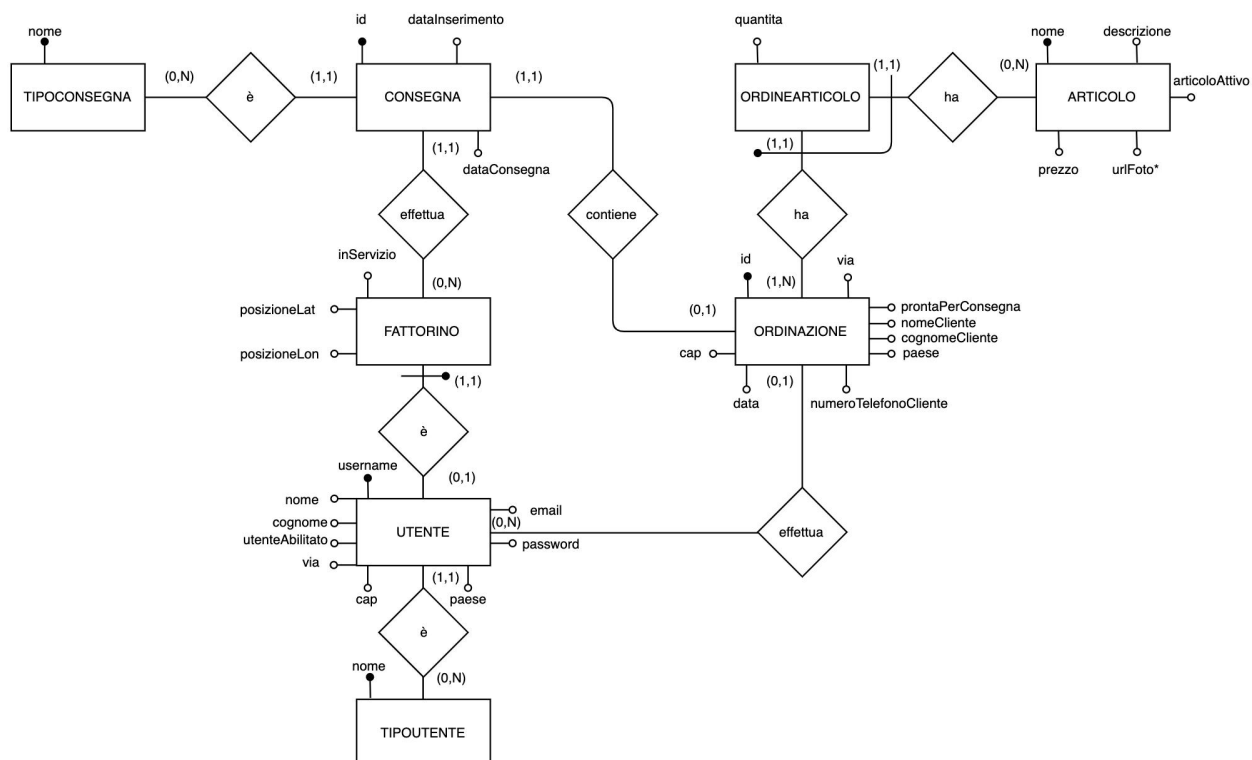
2.1 Design dell'architettura del sistema

L'applicazione funziona principalmente nel seguente modo.



2.2 Design dei dati e database

Il database che ha lo scopo di dare una base sul quale viene costruito tutto il programma è stato progettato nel seguente modo:



2.3 Design delle interfacce

Per le varie interfacce ho cercato di applicare lo stesso stile rendendole belle da vedere ma allo stesso tempo molto intuitive e semplici da usare.

2.3.1 Pagina Principale

È la prima pagina che appare all'apertura del sito sulla quale si può già ordinare oppure effettuare un login per gli impiegati della pizzeria.

INDEX

PizzaDelivery

Login

BENVENUTO
SU
PIZZADELIVERY

desc...

ORDINA ORA

<FOOTER>

2.3.2 Pagina Login

Pagina nel quale si può fare il login se si lavora nella pizzeria e si dispone di un account.

LOGIN

PizzaDelivery

Consegne
Ordinazioni
Fattorini
Login

Login

username

Password

Login

Having trouble logging in? [Click Here](#)

<FOOTER>

2.3.3 Pagina Amministratore


Pagina dalla quale l'amministratore del sistema può gestire, aggiungere, modificare e rimuovere gli account della pizzeria.

ADMIN

PizzaDelivery

Login

Gestione utenti

Username	tipologia	Modifica
paolo.rossi	fattorino	
<list item>		

crea utente

<FOOTER>

ADMIN ON EDIT

PizzaDelivery

Login

franco.rossi

Nome

cognome

email

...

rispristina

elimina

salva

<FOOTER>

2.3.4 Pagina Ordinazione

Pagina sulla quale è possibile selezionare i prodotti da ordinare ed inserire i propri dati per la comanda.

ORDINA

PizzaDelivery
Login

ORDINA

Google maps via picker

Select dei prodotti

Nome	Pz.	Costo
Pizza margherita	2	21.90.-
<list item>		

ORDINA

<FOOTER>

2.3.5 Pagina Fattorini

Pagina dalla quale si possono visualizzare tutti i fattorini dell'azienda e le loro informazioni più importanti.

FATTORINI

PizzaDelivery
Consegne Ordinanze Fattorini <Clienti>

FATTORINI

ID	Nome	Consegne effettuate oggi	Stato
3	Paolo Neri	5	In servizio
<list item>			
<list item>			
<list item>			
...			

<FOOTER>

FATTORINI ONCLICK

PizzaDelivery
Consegne Ordinanze Fattorini <Clienti>

FATTORINO #ID

Nome: Paolo

Cognome: Neri

Stato: Libero

Posizione attuale
Google maps map

Consegne effettuate oggi [11/03/2019]

ID Consegna	Orario	Via	Incasso
345	19:02:29	Via Pedemonte 3	54.20.-
<list item>			
<list item>			
...			

<FOOTER>

2.3.6 Pagina Ordinanze

Pagina dalla quale è possibile visualizzare le ordinazioni, il loro stato e tutte le informazioni importanti una volta cliccato su una di esse.

ORDINAZIONI

ID	Cliente	Posizione	Pz. Ordine
64	Franco Rossi	Via Scuola 11	3
<list item>			
<list item>			
<list item>			
...			

ORDINAZIONE #ID
ONCLICK

Cliente

Nome: Franco

Cognome: Rossi

E-mail: Franco@rossi.ch

Google maps
map
Via scuole 11

Prodotti ordinati

Nome	Prezzo
1x Pizza margherita	12.-
<list item>	
...	

2.3.7 Pagina Consegne

Pagina dalla quale si possono visualizzare tutte le consegne delle ultime 48 ore.

CONSEGNE

PizzaDelivery
Consegne Ordinazioni Fattorini <Clienti>

CONSEGNE

ID Ordine	Fattorino	Data consegna	Stato
64	Paolo Neri	12:34:12 24/10/19	Consegnato
<list item>			
<list item>			
<list item>			
...			

3 Implementazione

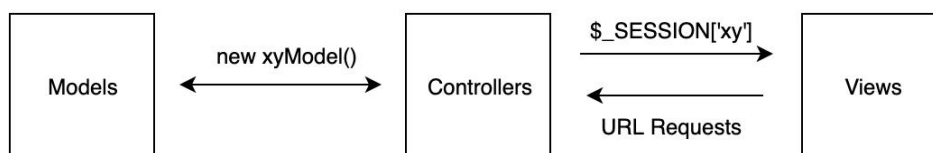
3.1 Struttura dell'applicativo

Tutta la struttura dell'applicativo è stato basato su una base del modello MVC di PHP fornita dal docente Massimo Sartori.

Lo scopo di questo modello Model-View-Controller è quello di semplificare la gestione di un progetto relativamente grande e soprattutto di offrire una visione più globale e pulita su tutti i file che lo compongono e del loro scopo.

La comunicazione tra queste tre principali tipologie di classi e file avviene nel seguente modo:

- I Model comunicano esclusivamente con i controller
- Il controller è la parte principale che collega i model e le views, per richiamare i metodi dai model crea un'istanza di quelli necessari e li richiama attraverso quest'oggetto. Con le views invece mette i dati all'interno della variabile `$_SESSION['xy']` con un nome 'xy' specifico.
- Le views infine ricevono attraverso le sessioni i dati e li stampano, per ricontattare i controllers usano chiamate URL che con il sistema di routing del modello MVC vengono reindirizzate ai metodi della classe model desiderata.



3.1.1 Routing URL

Una delle cose fondamentali dell'MVC è la classe application.php che gestisce il routing di tutte le richieste HTTP sotto forma di URL richiamando i metodi dei controller con eventuali parametri corretti.

Il risultato delle richieste viene effettuato nel seguente modo (esempio):

Richiesta URL:	Controller	Metodo	Parametri
pizzadelivery.com/prodotti/getProdotti	prodotti	getProdotti()	-
pizzadelivery.com/utenti/getUtente/paolo.neri	utenti	getUtente()	"paolo.neri"

3.1.2 Gerarchia delle cartelle

Lo scopo di questa suddivisione è di portare ordine al progetto, avendo più di 60 file e 20 immagini è ora possibile accedere in modo rapido all'elemento desiderato:

<ul style="list-style-type: none"> PizzaDelivery <ul style="list-style-type: none"> application <ul style="list-style-type: none"> config controller database img libs models scripts views <ul style="list-style-type: none"> _templates <ul style="list-style-type: none"> headers footer.php error pages css .htaccess index.php 	<p>MVC: Cartella Padre di tutta la struttura MVC</p> <p>Config: Contiene il file di configurazione della root della struttura</p> <p>Controller: Contiene tutte le classi che fanno da controller per l'applicazione</p> <p>Database: Contiene tutti i file di creazione del database e la classe che collega la struttura MVC con il DB.</p> <p>Models: Contiene tutte le classi che fanno da model per l'applicazione</p> <p>Scripts: Contiene tutti i file javascript che le varie views utilizzano per eseguire svariate operazioni sul lato client</p> <p>Views: Contiene tutte le views: <ul style="list-style-type: none"> ● Header ● Body ● Footer </p>
---	--

3.1.3 File di configurazione

I due file principali che si occupano della configurazione di MVC sono config.php che si trova nella cartella application/config e il .htaccess presente nella root del progetto.

3.1.3.1 .htaccess

Si occupa di gestire ed impostare il funzionamento di tutto il traffico di richieste e non, inoltre va ad interfacciarsi con il servizio Apache del WebServer attivando moduli come il RewriteBase che dà la possibilità di definire una root per il progetto oppure di riscrivere una regola come il RewriteRule.

Importante: questo file è solamente compatibile con WebServer di tipo Apache, se lo si vuole utilizzare ad esempio con Nginx bisognerà cambiare il formato del contenuto.

```
#Esempio: se la cartella configurazione/ e il file configuration.php si trovano nella root del sito e si tenta di
#raggiungere l'indirizzo monsite.com/configuration,
#si verrà automaticamente reindirizzati sul file configuration.php se l'opzione MultiViews è attivata.
Options -MultiViews

# attiva il modulo apache RewriteEngine
RewriteEngine On

# Disallows others to look directly into /public/ folder
Options -Indexes

# Project root
RewriteBase /php/Progetti/PizzaDelivery/src/MVC/

# regole di rewrite generali, se le 3 sotto sono valide allora riscrivi il link (fai partire RewriteCond)
#non eseguirla per una directory
RewriteCond %{REQUEST_FILENAME} !-d
#non eseguirla per regolare file che esiste
RewriteCond %{REQUEST_FILENAME} !-f
#non eseguirla per un link
RewriteCond %{REQUEST_FILENAME} !-l

#riscrivo la regola#In generale: prendi il parametro che trovi e processalo come un classico parametro di
#GET -->?parametro1&parametro2 eccetera
RewriteRule ^(.+)$ index.php?url=$1 [QSA,L]
```

3.1.3.2 config.php

Ha la funzione di specificare l'url del progetto nella costante URL, essa verrà poi usata in tutte le classi ogni volta che bisogna richiamare un metodo di un qualche controller attraverso richiesta HTTP.

```

}/**
 * Configurazione di : URL del progetto
 */
define('URL', 'http://localhost:8888/php/Progetti/PizzaDelivery/src/MVC/');
```

3.1.4 Classi Controller

Lo scopo principale delle classi controller è quello di unire i model con le view fornendo svariati metodi richiamabili attraverso richiesta URL.

Nei loro metodi vengono preparati i dati con un'istanza di una classe Model mettendoli successivamente all'interno della variabile globale \$_SESSION di php.

Infine richiamano le view corrette attraverso il require_once che si occuperà di caricarle e mostrarle a schermo per interagire con l'utente.

Ad esempio la classe *fattorini* esporta il metodo home() che viene caricato come pagina di default al click o ricerca dei fattorini, esso mette in sessione tutti i fattorini ricevuto da *fattoriniModel* e richiama le views necessarie:

```
public function home(){
    $_SESSION['fattorini'] = $this->pdModel->getFattorini();

    // Carico Views
    $this->header->getRightHeader();
    require 'application/views/pages/fattorini/fattorini.php';
    require 'application/views/_templates/footer.php';
}
```

Un esempio della struttura di una di queste classi può essere la classe controller GestionePizzeria.php che mette a disposizione tutti i suoi metodi che sono richiamabili tramite chiamata HTTP, il diagramma della classe è il seguente:



Se ad esempio volessimo disabilitare un utente (franco.neri) basterebbe fare la seguente chiamata con la tipologia di utente corretta:

<http://<serverWebRoot>/PizzaDelivery/gestionePizzeria/disabilita/franco.neri>

3.1.5 Cartella libs

Nella cartella libs sono presenti tutte le librerie scaricate ed usate in questo progetto:

Bootstrap 4.3.1-dist

Si occupa di gestire tutta la struttura html e css delle varie interfacce oltre al loro design e rende anche tutta l'applicazione utilizzabile su tutti le tipologie di dispositivo essendo una libreria creata seguendo l'approccio mobile-first.

jQuery 3.4.1

Libreria basata su JavaScript che semplifica la scrittura del codice lato client con JavaScript offrendo anche metodi suoi per accorciare la lunghezza delle istruzioni.

Popper: 1.3.3

Libreria JavaScript utilizzata da certe operazioni di Bootstrap.

Fontawesome free-5.11.2-web

Libreria di icone che servono a rendere più belle graficamente le pagine html.

3.1.6 Classi Model

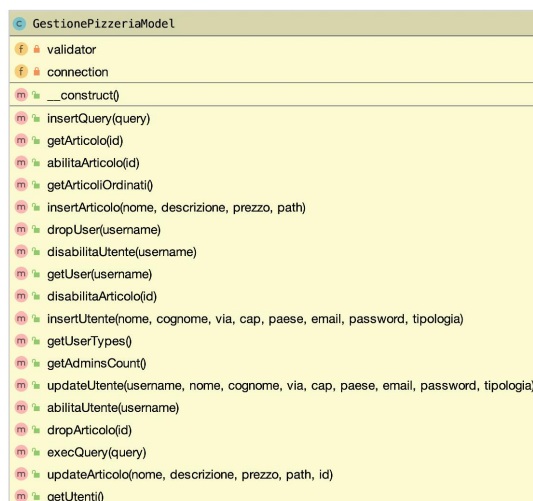
Le classi presenti nella cartella model sono quelle che lavorano direttamente con le fonti di dati come ad esempio un database, file o socket.

Elaborano e preparano i dati in svariati metodi che poi possono essere richiamati da classi esterne (nel nostro caso i controller) attraverso un'istanza del proprio oggetto.

In questo esempio la classe ordinazioniModel.php prepara il metodo getArticoli() che si occupa di ritornare un'array contenente tutti gli articoli presenti nella tabella *articolo* del database.

```
public function getArticoli(){
    return $this->execQuery("SELECT * FROM articolo;");
}
```

Spesso la struttura delle classi Model è molto simile, ad esempio la GestionePizzeriaModel.php è strutturata nel seguente modo:



A differenza del controller GestionePizzeria.php il model lavora direttamente con i dati che si trovano sul database, se ad esempio eseguo il metodo disabilitaArticolo(3) viene impostato a 0 (false) il valore del campo "articoloAttivo" dell'articolo con id = 3.

3.1.7 Scripts

Generalmente tutti gli script utilizzati dalle varie pagine (file .js) sono salvati in questa cartella. Essi si occupano di eseguire determinate azioni sul lato client dell'applicazione (codice accessibile a qualsiasi utente ispezionando la pagina) a differenza di PHP che lavora lato server.

La funzione validate si occupa di controllare il testo immesso in un input che poi viene passato a questo metodo per poter dare un feedback visivo all'utente.

```
function validate(string, maxlen, regex){
    try{
        if(regex === null){
            return (string.length > 0);
        }

        regex = new RegExp(regex);
        string = string.trim();

        if(string.length > 0 && string.length <= maxlen){
            if(regex.test(string)){
                return true;
            }
        }

        return false;
    }catch(error){
        console.log("Error: " + error);
        return false;
    }
}
```

(Metodo semplificato per la dimostrazione, interagisce anche con altri)

A dipendenza del suo esito l'utente vede se il suo input è accettato oppure no.

Informazioni personali

Oppure

Informazioni personali

3.1.8 Views

Nella cartella views sono presenti tutte le pagine del progetto che uniscono l'iterazione dell'utente con quella dell'applicazione.

Mostrano a schermo tutti i dati ricevuti ed elaborati dalle classi controller con un certo senso creando un'interfaccia grafica semplice ed intuitiva da utilizzare accessibile a qualsiasi tipologia di utente.

Un esempio è la pagina di benvenuto che si occupa di accogliere l'utente al suo primo accesso alla pagina.

3.1.9 Classe Database

Lo scopo della classe database.php è quello di stabilire una connessione con il database 'PizzaDelivery' sul quale sono salvati tutti i dati per il funzionamento dell'applicativo e di ritornarla alle classi Model che la utilizzano nel seguente modo:

```
public static function getConnection(){
    if(self::$_connection == null){
        try {
            self::$_connection = new PDO(
                dsn: "mysql:host=" . self::HOST . ";port=" . self::PORT . ";dbname=" . self::DATABASE,
                username: self::USERNAME,
                passwd: self::PASSWORD
            );
            // set the PDO error mode to exception
            self::$_connection->setAttribute( attribute: PDO::ATTR_ERRMODE, value: PDO::ERRMODE_EXCEPTION);
        } catch(PDOException $e){
            header( string: "Location: " . URL . "errorController/requireConnectionError");
        }
    }

    if(isset(self::$_connection)){
        return self::$_connection;
    }else{
        header( string: "Location: " . URL . "errorController/requireConnectionError");
    }
}
```

Oltre a ciò offre anche tutti i metodi che servono ad interagire con il DB, il suo funzionamento è relativamente semplice, basta creare una query ed eseguirla:

```
public function execQuery(string $query){
    try{
        $query = $this->validator->validateString($query);
        $tmp = $this->connection->prepare($query);
        $tmp->execute();
        return $tmp->fetchAll( fetch_style: PDO::FETCH_ASSOC);
    } catch(PDOException $e){
        $_SESSION['queryError'] = $e->getMessage();
        header( string: "Location: " . URL . "errorController/requireQueryError");
    }
}
```

3.2 Database

La fonte principale sulla quale si trovano tutti i dati utilizzati dall'applicazione è il database chiamato PizzaDelivery.

Attraverso la classe database.php spiegata nel capitolo 3.1.9 e l'utente SQL PD_Admin il programma si interfaccia con la banca dati continuando a scambiare, inserire o modificare dati a dipendenza delle operazioni eseguite tramite l'interfaccia grafica costituita dalle varie views.

Per il funzionamento corretto del software il database deve contenere alcuni dati di default/importanti come almeno un'utente amministratore già presente nel database con il quale eseguire il primo login potendo così personalizzare l'applicativo.

Inoltre si necessitano anche dei informazioni come le tre tipologie di utente avanzate (fattorino, impiegato vendita e amministratore) e le tipologie di consegna (da effettuare, in corso e terminata).

Tutto questi dati per il setup iniziale di default si possono trovare nel file **DefaultDataTabelle.sql**.

Inoltre sono anche stati creati dei trigger per semplificare e ridurre i comandi lato PHP, ad esempio quando viene creato un nuovo utente di tipo fattorino viene automaticamente creato un nuovo record nella tabella fattorino con i suoi dati nel seguente modo:

```
DELIMITER //
CREATE TRIGGER addFattorino AFTER INSERT ON Utente FOR EACH ROW
BEGIN
    IF NEW.tipoUtente LIKE 'fattorino' THEN
        INSERT INTO fattorino VALUES (NEW.username, 8.971826, 46.029792, false);
    END IF;
END
//DELIMITER ;
```

Quando invece viene eliminato un utente di tipo fattorino bisogna anche toglierlo nei fattorini per non avere dati incoerenti, per quest'azione non si necessita un trigger essendo sviabile aggiungendo la clausola "ON DELETE CASCADE" sulla chiave esterna che punta al record dell'utente:

```
FOREIGN KEY (username) REFERENCES Utente(username) ON DELETE CASCADE
```


3.3 Sicurezza e ruoli dell'applicativo

Essendo un programma che viene utilizzato commercialmente e che quindi tratta con soldi e dati sensibili di persone deve garantire un livello di sicurezza molto alto.

Questo viene raggiunto implementando ed utilizzando un sistema di login il quale gestisce quattro livelli di sicurezza tra cui si trovano le seguenti tipologie di account.

Utente normale	Persona qualsiasi che effettua un'ordinazione sul sito.
Fattorino	Fattorino che esegue le consegne a domicilio.
Impiegato vendita	Impiegato che si occupa di assegnare le varie ordinazioni e monitorare il loro stato.
Amministratore	Ha il potere completo sull'applicazione potendo anche creare, modificare ed eliminare persone oppure prodotti.

Queste quattro tipologie di utente possono accedere alle rispettive pagine:

	Home	Ordina	Ordinazioni	Consegne	Fattorini	Gestione Pizzeria
Utente normale	X	X				
Fattorino	X	X		X	X	
Impiegato vendita	X	X	X	X	X	
Amministratore	X	X	X	X	X	X

Attraverso un pannello di login è possibile effettuare l'accesso con un account ottenendo più funzionalità di quelle dell'utente normale.

Se ad esempio accedo con un utente di tipo fattorino l'applicazione si adatterà automaticamente cambiando il contenuto della barra di navigazione mostrando solamente le pagine accessibili nel seguente modo:

PizzaDelivery	Home	Ordina	Login
---------------	------	--------	-------

(Prima del login)



PizzaDelivery • fattorino	Home	Ordina	Consegne	Fattorini	Logout
---------------------------	------	--------	----------	-----------	--------

(Dopo il login)

Inoltre l'applicazione blocca anche l'accesso alle pagine nelle quali non si hanno i permessi sufficienti, se come Fattorino provo ad accedere alla pagina Gestione Pizzeria dell'amministratore il programma mi ritorna il seguente errore:

ERRORE: Permessi insufficienti.

3.4 Codice JavaScript lato client

Per rendere l'applicativo in parte più sicuro ma soprattutto più invitante per l'utente ho implementato vari controlli lato client che danno un feedback visivo all'utente come ad esempio nei form della pagina ConfermaOrdine dove bisogna immettere i propri dati personali.

In ogni form sono stati apportati dei criteri di validazione leggermente differenti a dipendenza del loro scopo, ad esempio il campo che richiede il CAP accetta solamente numeri mentre il campo Nr. (numero della via di casa) accetta sia numeri che lettere.

Nr.	Cap
-----	-----

Quando si immette del testo il bordo dei vari campi cambia a dipendenza del contenuto immesso, se è accettato diventa verde altrimenti rosso.

12A?	12
------	----

Inoltre non è nemmeno possibile confermare il proprio ordine se non sono stati completati tutti i campi obbligatori nel modo corretto, la pagina mostra il seguente errore.

Errore: Immetti correttamente tutti i dati.

La validazione dei vari campi funziona nel seguente modo, la struttura è quasi sempre la stessa e cambia solamente l'espressione regolare che definisce i caratteri accettati.

Inizialmente per ogni input c'è un evento collegato che ad ogni carattere nuovo immesso controlla la validità del contenuto per dare un feedback immediato, questo accade nel seguente modo:

```
// nome field
var nameSelector = $('input[name=nome]');
nameSelector.keyup(function(event){
    if(validate(nameSelector.val(), LUNGHEZZA_MASSIMA_NOME, regex: /^[A-Za-zöäüÖÜàèìòùÀÈÌÒÙÉÉ\.\-]+$/)){
        isOk(nameSelector);
        status[0] = true;
    }else{
        isNotOk(nameSelector);
        status[0] = false;
    }
    checkIfOk();
});
```

(Esempio con l'input del campo "nome")

Una volta controllato il testo con il metodo validate a dipendenza del suo valore di ritorno il bordo del campo viene colorato in rosso oppure verde.

La struttura del metodo validate invece è la seguente, riceve tre parametri tra cui il valore dell'input stesso, la lunghezza massima della stringa definita da una costante ed infine l'espressione regolare contenente i caratteri accettabili.

```
function validate(string, maxLen, regex){
  try{
    if(regex === null){
      return (string.length > 0);
    }

    regex = new RegExp(regex);
    string = string.trim();

    if(string.length > 0 && string.length <= maxLen){
      if(regex.test(string)){
        return true;
      }
    }

    return false;
  }catch(error){
    console.log("Error: " + error);
    return false;
  }
}
```

Inizialmente controlla se la regex non è nulla, altrimenti ritorna false. Successivamente crea l'oggetto RegExp con la stringa regex ricevuta come parametro e se la lunghezza della stringa è all'interno dei limiti accettati la testa guardando se rispetta i criteri impostati dall'espressione regolare ritornando true oppure false a dipendenza del suo esito.

Infine i metodi che si occupano di colorare il bordo degli input funzionano molto semplicemente nel seguente modo ricevendo come parametro l'elemento da colorare:

```
function isOk(selector){
  selector.css('border-color', '#abdd92');
}
```

Tutti questi controlli lato client sono generalmente effettuati dai vari file javascript presenti nella cartella scripts.

3.5 Pagina Gestione Pizzeria

La pagina che si occupa di gestire tutta la pizzeria è la GestionePizzeria accessibile solamente agli amministratori del sistema.

Tramite i vari form che contiene è possibile aggiungere, modificare, disabilitare ed eliminare utenti oppure articoli offerti dalla pizzeria.

Per ognuna di queste due sezioni esiste una tabella che mostra i record già presenti nel database, inoltre è anche possibile eseguire una ricerca lato client senza dover effettuare alcune query ma lavorando direttamente sui dati ricevuti e presenti nella tabella grazie alla seguente funzione della libreria jQuery:

```
//Tabella Utente
$("#cercaUtente").on("keyup", function() {
    var value = $(this).val().toLowerCase();
    $("#utentiTable tr").filter(function() {
        $(this).toggle($(this).text().toLowerCase().indexOf(value) > -1)
    });
});
```

Questo trucchetto l'ho trovato su W3Schools (https://www.w3schools.com/jquery/jquery_filters.asp) e funziona nel seguente modo.

Ogni volta che l'utente aggiunge o rimuove un carattere nella barra di ricerca viene richiamata la funzione mostrata nell'immagine soprastante, successivamente prende il contenuto della ricerca e la filtra per tutte le righe della tabella togliendo quelle che non contengono un'uguaglianza con il testo immesso.

Un esempio pratico è il seguente:

Gestisci Utenti

Cerca Utente	+ Crea Utente		
Username	Tipologia	Abilitato	Modifica
franco.gialli	impiegato vendita	✓	
franco.neri	fattorino	✗	
gianni.grandi	fattorino	✓	
jari.naeser	amministratore	✓	

(Nessuna ricerca)

Gestisci Utenti

ja	+ Crea Utente		
Username	Tipologia	Abilitato	Modifica
jari.naeser	amministratore	✓	

(Con ricerca "ja")

Successivamente è possibile aggiungere utenti o articoli attraverso i bottoni "crea Utente"/"crea Articolo". Cliccando sull'icona della voce modifica in una delle righe si possono modificare i dati del record selezionato oppure disattivarlo.

Se ad esempio modifico l'utente franco.gialli arrivo alla seguente interfaccia:

Utente franco.gialli

Nome *	<input type="text" value="Franco"/>
Cognome *	<input type="text" value="Gialli"/>
Via *	<input type="text" value="Via Scuole 12"/>
CAP *	<input type="text" value="6900"/>
Paese *	<input type="text" value="Svizzera"/>
E-Mail *	<input type="text" value="franco.gialli@pizzadelivery.ch"/>
Password *	<input type="text" value="Password non mostrata."/>
Stato *	<input type="button" value="Disabilita"/>
Tipologia *	<input type="text" value="impiegato vendita"/>

Se si modificano i valori dell'utente è possibile salvarli attraverso il bottone aggiorna.

Inoltre esiste anche l'opzione "disabilita" che se attivata disabilita l'account rendendolo inutilizzabile ed inaccessibile, questa feature può tornare utile se un impiegato va in ferie per svariato tempo oppure ha un infortunio assentandolo dal lavoro per un grande periodo di tempo.

Disabilitando il suo account basterà un click da parte dell'amministratore quando torna e si evita anche la perdita di dati collegati al suo account come le consegne effettuare nel caso di un fattorino.

Infine è anche stato implementato un ultimo filtro di sicurezza che riguarda l'eliminazione degli utenti amministratori, infatti il sistema deve sempre averne almeno uno.

Se si prova ad eliminare oppure disabilitare l'ultimo account amministratore il programma ritorna il seguente messaggio di errore:

ERRORE

Stai cercando di eliminare o disattivare l'unico admin del sistema.

Se si vuole proseguire con questa azione eliminarlo attraverso l'SQL.

3.6 Implementazione Mappe

Per quanto riguarda le mappe implementate nelle pagine Ordinazione e Fattorino ho utilizzato il servizio gratuito di MapBox il quale rende molto semplice l'implementazione delle mappe con lo stile di Google Maps nelle proprie pagine attraverso JavaScript.

La documentazione è molto intuitiva e permette di aggiungere molti elementi aggiuntivi come puntatori, popup eccetera.

Attraverso una chiave privata ricevuta alla creazione di un account sul loro sito è possibile fare delle richieste HTTP specificando vari parametri come la posizione della mappa che si vuole ricevere oppure i valori di latitudine e longitudine, successivamente se la richiesta è strutturata correttamente il sito ritorna un file in formato JSON contenente tutti i dati necessari per la costruzione della mappa.

Per mostrare la mappa sulle proprie pagine bisogna seguire la loro documentazione utilizzando i vari metodi e funzioni necessari javascript, uno di questi è la funzione onload:

```
map.on('load', function () {

    map.addImage('pulsing-dot', pulsingDot, { pixelRatio: 2 });

    map.addLayer({
        "id": "places",
        "type": "symbol",
        "source": {
            "type": "geojson",
            "data": {
                "type": "FeatureCollection",
                "features": [{
                    "type": "Feature",
                    "properties": {
                        "description": "<strong>Fattorino: <?php echo $userFattc",
                        "icon": "theatre"
                    },
                    "geometry": {
                        "type": "Point",
                        "coordinates": [<?php echo $fattorino['posizioneLat']; ?

```

Si occupa di impostare i valori più importanti per la mappa come la posizione ed il puntatore, una volta implementato e personalizzato correttamente tutti i metodi e funzioni si ottiene la propria mappa.

Nel mio caso la posizione attuale di ogni fattorino viene mostrata su una mappa basata su MapBox:

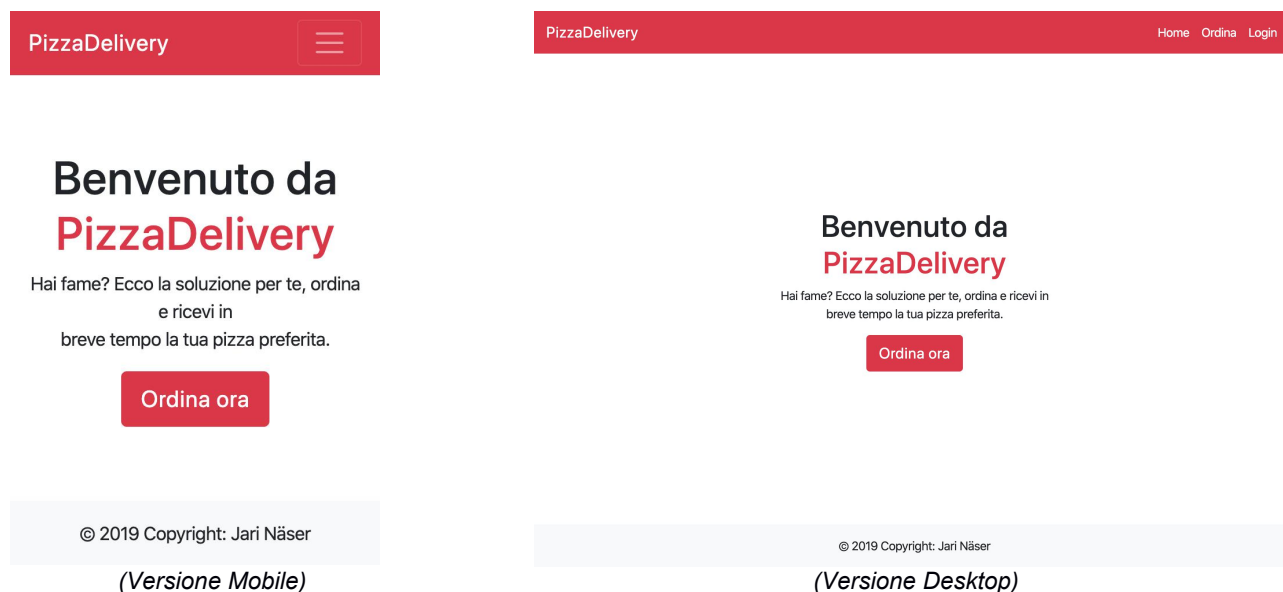


3.7 Approccio Mobile First

Essendo uno dei temi più importanti nella realizzazione di applicativi basati su pagine web ho ritenuto necessario creare un programma che sia utilizzabile ed apprezzabile graficamente su qualsiasi tipologia di dispositivo.

Per poter far ciò ho utilizzato un approccio Mobile first usando la libreria Bootstrap che si occupa di gestire automaticamente gran parte dell'adattamento delle dimensioni a dipendenza della grandezza dello schermo sul quale viene aperta l'applicazione.

Ecco un paragone tra un dispositivo mobile ed un computer desktop:



3.8 Installazione debugger

Il debugger che ho scelto di utilizzare in questo progetto è xDebug.

3.8.1 Download

Usando MAMP non ho dovuto effettuare alcun tipo di download essendo che la sua libreria esiste già nella cartella della versione di php che si utilizza attualmente.

Path sul mio computer: `/Applications/MAMP/bin/php/php7.2.1/lib/php/extensions/no-debug-non-zts-20170718/xdebug.so`

3.8.2 Configurazione php.ini

Per poter implementare il debugger bisogna aprire il file di configurazione php.ini della versione di php utilizzata attualmente dal WebServer.

Per semplificare questo passo è possibile farlo anche dalla IDE di phpStorm sotto "Preferences > Languages & Frameworks > PHP".

Una volta in questo tab bisogna cliccare i tre puntini che stanno di fianco a CLI Interpreter:



E poi successivamente cliccare "Open in Editor"



Una volta aperto il file bisogna aggiungere/modificare i seguenti elementi:

Togliere Zend debugger e Zend tools (rimuovere o commentare le sue seguenti righe)

```
zend_extension=<path_to_zend_debugger>
zend_extension=<path_to_zend_optimizer>
```

Spostarsi in fondo al file alla voce [xdebug] e settare la path di xdebug nel zend_extension

```
zend_extension="/Applications/MAMP/bin/php/php7.2.1/lib/php/extensions/no-debug-non-zts-20170718/xdebug.so"
```

Aggiungere le ulteriori righe di configurazione (non copiare anche le descrizioni delle righe in *italico*)

<code>xdebug.remote_autostart = 1</code>	<i>Inizia una sessione di debug esterna</i>
<code>xdebug.profiler_append = 0</code>	<i>Aggiunge il profiler al nuovo profilo</i>
<code>xdebug.profiler_enable = 0</code>	<i>Disabilita il profiler</i>
<code>xdebug.profiler_enable_trigger = 0</code>	<i>Disabilita il trigger del profiler</i>
<code>xdebug.profiler_output_dir = "/tmp"</code>	<i>Imposta la cartella di output del profiler</i>
<code>xdebug.remote_enable = 1</code>	<i>Definisce se contattare un client di debug (phpstorm)</i>
<code>xdebug.remote_handler = "dbg"</code>	<i>Definisce il protocollo di debug</i>
<code>xdebug.remote_host = "localhost"</code>	<i>Definisce l'host remoto a cui collegarsi</i>
<code>xdebug.remote_log = "/tmp/xdebug.log"</code>	<i>Definisce il file in cui vengono messi i log</i>
<code>xdebug.remote_port = 10000</code>	<i>Definisce la porta di ascolto remota</i>
<code>xdebug.trace_output_dir = "/tmp"</code>	<i>Definisce la cartella in cui mette l'output</i>
<code>xdebug.remote_cookie_expire_time = 36000;</code>	<i>Definisce il tempo di esistenza di un cookie, in questo caso 36000 secondi equivalgono a 10 ore</i>

3.8.3 Configurazione phpStorm

Per poter utilizzare xDebug anche sull'IDE che ho utilizzato per lo sviluppo di questo progetto, phpStorm, basta eseguire pochi passi molto semplici.

Inizialmente bisogna aprire le preferenze e andare nel tab “PHP” sotto “Languages & Frameworks > PHP”.

Successivamente si devono cliccare i tre puntini che stanno alla destra della versione del CLI interpreter

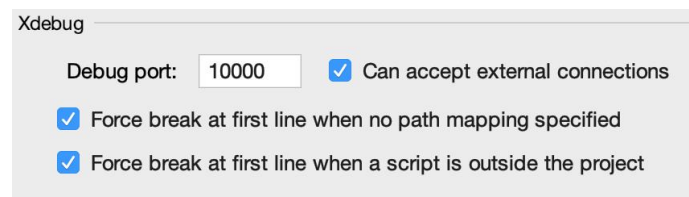


Una volta cliccati dobbiamo inserire la path del file xdebug.so (.dll su windows) inserita precedentemente nel php.ini in “zend_extension” nel textbox “Debugger extension:”

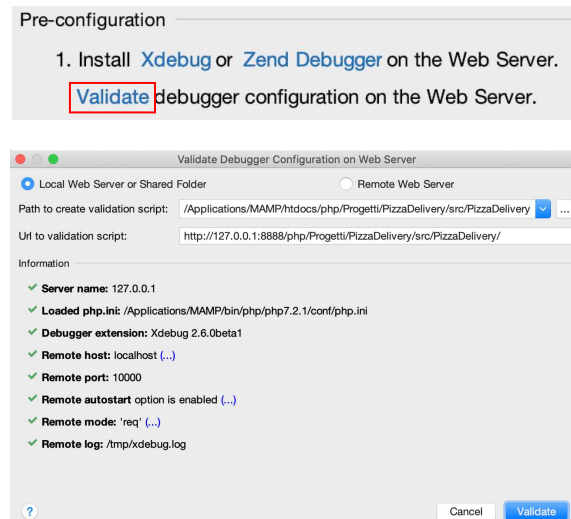


Una volta fatto questo premere “Apply” e “OK” per uscire da questa schermata.

Come passo successivo dobbiamo andare nella voce “debug” del tab PHP nel quale dobbiamo aggiornare la porta di connessione di xdebug inserendo quella specificata precedentemente con “xdebug.remote_port = 10000” nel file php.ini



Infine possiamo cliccare “Validate debugger” che esegue dei test sulle impostazioni appena modificate e se tutto è andato a buon fine appare la seguente finestra



Infine per eseguire un test di debug basta mettere un breakpoint in un file e cliccare l'icona del telefono e quella del “bug”, successivamente appena l'esecuzione del codice arriverà al punto del breakpoint l'IDE entrerà in modalità debug mostrando tutti i dati.



4 Test

4.1 Protocollo di test

Test Case:	TC-01	Nome:	Test del database
Riferimento:	REQ-03		
Descrizione:	Test sul funzionamento corretto della banca dati utilizzata dal progetto		
Prerequisiti:	Disporre di un database funzionante e costruito rispettando il diagramma ER		
Procedura:	<ol style="list-style-type: none"> 1. Collegarsi con il database PizzaDelivery attraverso l'utente PD_Admin 2. Eseguire una insert nella tabella TipoConsegna: <code>INSERT INTO TipoConsegna VALUES ("in corso");</code> 		
Risultati attesi:	La tabella TipoConsegna contiene il nuovo record con il valore "in corso"		

Test Case:	TC-02	Nome:	Test struttura MVC
Riferimento:	REQ-04		
Descrizione:	Test del funzionamento corretto della struttura MVC, viene soprattutto testato il routing corretto all'interno dell'applicazione		
Prerequisiti:	Disporre della struttura MVC configurata correttamente e funzionante		
Procedura:	<ol style="list-style-type: none"> 1. Effettuare una richiesta HTTP tramite browser che richiama il controller home: <code>http://<server>/<htdocs>/PizzaDelivery/home/index</code> 		
Risultati attesi:	Il metodo index del controller home ci ritorna la pagina index di default		

Test Case:	TC-03	Nome:	Test connessione tra modello MVC e database
Riferimento:	-		
Descrizione:	Test del funzionamento e comunicazione corretta tra la struttura MVC e il database		
Prerequisiti:	Disporre di un database funzionante contenente dei dati nella tabella TipoUtente Disporre del file database.php presente di default nella struttura MVC Disporre di una struttura MVC funzionante		
Procedura:	<ol style="list-style-type: none"> 1. Creazione di una classe model di test nella cartella models 2. Creazione di un'oggetto di tipo Database utilizzando la classe Database 3. Eseguire il metodo getConnection dall'oggetto Database 4. Utilizzare il metodo prepare() della connessione inserendo la seguente query: <code>SELECT * FROM TipoUtente;</code> 5. Richiamare il metodo execute() della connessione 6. Eseguire il fetch() della connessione ricavando i dati ritornati dal database 		
Risultati attesi:	Il database ritorna le varie tipologie di utente presenti nella tabella TipoUtente.		

Progetto PizzaDelivery

Test Case:	TC-04	Nome:	Test funzionamento pagina Ordina
Riferimento:	-		
Descrizione:	Test del funzionamento corretto della pagina Ordina e delle sue sotto-pagine.		
Prerequisiti:	Disporre della pagina Ordina funzionante e completa		
Procedura:	<ol style="list-style-type: none"> 1. Accedere alla pagina Ordina 1. Creazione del proprio ordine simulando quello di un'utente qualsiasi 2. Proseguire alla pagina conferma ordine tramite il bottone "Prosegui all'ordinazione" 3. Immettere correttamente i propri dati personali 4. Definire la quantità desiderata per ogni articolo 5. Cliccare il bottone "Ordina" 		
Risultati attesi:	Deve apparire la pagina che comunica "Il tuo ordine verrà elaborato il più presto possibile"		

Test Case:	TC-05	Nome:	Test funzionamento pagina Login
Riferimento:	-		
Descrizione:	Test del funzionamento corretto della pagina di login, deve rispettare tutti i criteri di sicurezza facendo effettuare un login in modo sicuro.		
Prerequisiti:	Disporre di almeno un account con il quale poter effettuare il login e dei livelli di sicurezza come diverse tipologie di account		
Procedura:	<ol style="list-style-type: none"> 2. Accedere alla pagina Login 3. Immettere il nome utente dell'account scelto 4. Immettere la password dell'account scelto 		
Risultati attesi:	Login effettuato correttamente, a dipendenza della tipologia dell'utente testato si possono accedere svariate pagine in più rispetto all'utente qualsiasi.		

Test Case:	TC-06	Nome:	Test funzionamento pagina Ordinanze
Riferimento:	REQ-05		
Descrizione:	Test del funzionamento corretto della pagina Ordinanze		
Prerequisiti:	Aver effettuato il login con un account di tipologia impiegato vendita oppure amministratore		
Procedura:	<ol style="list-style-type: none"> 1. Accedere alla pagina Ordinanze 2. Assegnare un'ordinazione ad un fattorino tramite il bottone "Assegna a fattorino" 		
Risultati attesi:	L'ordinazione assegnata ad un fattorino deve sparire dalle ordinazioni, si è spostata automaticamente nelle consegne.		

Progetto PizzaDelivery

Test Case:	TC-07	Nome:	Test funzionamento pagina Consegne
Riferimento:	REQ-06		
Descrizione:	Test del funzionamento corretto della pagina Consegne		
Prerequisiti:	Aver effettuato il login con un account di tipologia superiore all'utente qualsiasi (quindi fattorino, impiegato vendita oppure amministratore)		
Procedura:	<ol style="list-style-type: none"> 1. Accedere alla pagina Consegne 2. Modificare il filtro che mostra le consegne a dipendenza della loro data d'inserimento "Fino a" 3. Modificare lo stato di una consegna 		
Risultati attesi:	Modificando il filtro "Fino a" si vedono diverse quantità di record, inoltre modificando lo stato di una consegna si vede anche visualmente il cambiamento.		

Test Case:	TC-08	Nome:	Test funzionamento pagina Fattorini
Riferimento:	REQ-07		
Descrizione:	Test del funzionamento corretto della pagina Fattorini		
Prerequisiti:	Aver effettuato il login con un account di tipologia superiore all'utente qualsiasi (quindi fattorino, impiegato vendita oppure amministratore)		
Procedura:	<ol style="list-style-type: none"> 4. Accedere alla pagina Fattorini 1. Cliccare su uno dei fattorini 		
Risultati attesi:	Caricamento della pagina fattorino cliccato con tutti i suoi dati		

Test Case:	TC-09	Nome:	Test funzionamento pagina GestionePizzeria
Riferimento:	-		
Descrizione:	Test del funzionamento corretto della pagina GestionePizzeria		
Prerequisiti:	Aver effettuato il login con un account di tipologia amministratore.		
Procedura:	<ol style="list-style-type: none"> 1. Accedere alla pagina GestionePizzeria <p>I Prossimi test effettuarli sia per gli utenti che per gli articoli:</p> <ol style="list-style-type: none"> 2. Eseguire una ricerca 3. Aggiungere un nuovo elemento 4. Modificare l'elemento creato 5. Disabilitare l'elemento creato 6. Eliminare l'elemento creato 		
Risultati attesi:	<p>Alla ricerca il sistema mostra solamente gli elementi nelle due tabelle che corrispondono al valore cercato, alla creazione di un nuovo elemento esso viene creato, se viene modificato e si salvano le modifiche l'elemento applica i nuovi valori modificati anche nel database.</p> <p>Quando viene cliccato il bottone "disabilita" l'account viene disabilitato ed infine alla sua eliminazione viene cancellato dal database.</p>		

Test Case:	TC-10	Nome:	Test dei criteri di sicurezza
Riferimento:	-		
Descrizione:	Test del corretto funzionamento dei criteri di sicurezza		
Prerequisiti:	Disporre di un programma completamente funzionante		
Procedura:	<ol style="list-style-type: none"> 1. Accedere con un utente qualsiasi alla pagina Gestione Pizzeria 2. Eseguire il login con un account disabilitato 3. Eseguire delle SQL injection immettendo valori pericolosi nei vari form 4. Eliminare oppure disabilitare l'ultimo utente amministratore 5. Richiamare il metodo disabilitaUtente senza aver effettuato il login disabilitando un'utente a scelta: <code>http://<server>/<htdocs>/PizzaDelivery/gestionePizzeria/disabilitaUtente/<nome.utente></code> 		
Risultati attesi:	<p>Il programma ritorna la pagina di errore "permessi insufficienti" se si tenta di accedere una pagina privata come GestionePizzeria senza avere i permessi di amministratore.</p> <p>Il login con un account disabilitato non è possibile e anche le SQL injection non vengono eseguite grazie ai svariati controlli prima dell'esecuzione delle query.</p> <p>Non è possibile disabilitare oppure eliminare l'ultimo amministratore tramite interfaccia grafica e il metodo disabilitaUtente non può venir richiamato senza aver effettuato il login.</p>		

Test Case:	TC-11	Nome:	Funzionamento corretto algoritmo path finding
Riferimento:	-		
Descrizione:	Test del corretto funzionamento dell'algoritmo di path finding che trova la strada più corta da percorrere per un fattorino quando esegue più consegne in una sola uscita		
Prerequisiti:	Disporre di un programma completamente funzionante		
Procedura:	<p>Accedere alla pagina consegne</p> <p>Visualizzare il percorso trovato dall'algoritmo in una consegna</p>		
Risultati attesi:	Il programma ritorna il percorso più breve che il fattorino dovrà effettuare unendo più consegne.		

4.2 Risultati test

Test case	Esito Test	Note
TC-01	PASSATO	
TC-02	PASSATO	
TC-03	PASSATO	
TC-04	PASSATO	
TC-05	PASSATO	
TC-06	PASSATO	
TC-07	PASSATO	
TC-08	PASSATO	
TC-09	PASSATO	
TC-10	PASSATO	
TC-11	NON PASSATO	Non implementato

4.3 Mancanze/limitazioni conosciute

Le uniche mancanze del programma rispetto alle specifiche sono la rilevazione di trovare il percorso più breve possibile quando un fattorino deve eseguire più consegne in una sola uscita, non è stato implementato per la mancanza di tempo e la complessità del problema.

Inoltre quando si vuole modificare lo stato di una consegna non è già selezionato di default il valore attuale, sarebbe da implementare per portare più facilità nell'uso.

5 Installazione

Per poter utilizzare questa applicazione bisogna seguire un paio di passi molto semplici ma fondamentali per il corretto funzionamento dell'applicazione.

5.1 Requisiti iniziali

- Bisogna avere una macchina che faccia da WebServer (Ad esempio con software ...AMP) con:
 - Almeno 200MB di spazio disponibile sul disco
 - PHP installato, versione minima dell'interprete: 7.2.1
 - Servizio MySQL attivo (versione consigliata, da 5.6.38 in su)

5.2 Copiatura della cartella PizzaDelivery sulla propria macchina

Per poter usare l'applicazione bisogna copiare tutta la cartella "PizzaDelivery" ed il suo contenuto presente nella cartella "src" della repository di GitHub nella propria cartella "htdocs" (cartella di default dove apache cerca i file da servire) sul proprio WebServer.

ATTENZIONE: Assicurarsi di copiare tutti i file, anche quelli nascosti come il `.htaccess` altrimenti l'applicazione non funziona.

5.3 Creazione DataBase e inserimento dei dati essenziali

Per la creazione del database basta eseguire il file `PizzaDeliveryDB.sql` attraverso console oppure programmi come MySQL WorkBench avendo il servizio MySQL attivo.

Questo file che si trova in: `PizzaDelivery/application/database/PizzaDeliveryDB.sql` e contiene tutta la struttura del database oltre all'utente che permette di accedere ad MVC al database.

Se il database non è sulla stessa macchina del WebServer bisogna andare a modificare l'indirizzo di accesso dell'utente dal quale esegue le query, quindi l'ip pubblico della macchina WebServer:

Modificare da:

```
DROP USER 'PD_Admin'@'localhost';

CREATE USER 'PD_Admin'@'localhost' IDENTIFIED BY 'P0r74C431CaTo3Enwicncw';
GRANT ALL PRIVILEGES ON PizzaDelivery.* TO 'PD_Admin'@'localhost';
FLUSH PRIVILEGES;
```

A:

```
DROP USER 'PD_Admin'@'10.20.4.135';

CREATE USER 'PD_Admin'@'10.20.4.135' IDENTIFIED BY 'P0r74C431CaTo3Enwicncw';
GRANT ALL PRIVILEGES ON PizzaDelivery.* TO 'PD_Admin'@'10.20.4.135';
FLUSH PRIVILEGES;
```

Altrimenti lasciare "localhost" se tutto si trova sulla stessa macchina.

Per modificare la password invece basta cambiare il valore del campo:

```
IDENTIFIED BY 'NuovaPassword';
```


Infine eseguiamo anche il file DefaultDataTabelle.sql che contiene i dati default del database come le tipologie di utente, utente amministratore eccetera.

Il file è presente nel percorso: *PizzaDelivery/application/database/DefaultDataTabelle.sql*

Se si vogliono modificare dati è consigliato farlo dall'interfaccia "Gestione pizzeria" una volta eseguito il login con l'account admin (admin.pizzadelivery nel nostro caso) e dopo l'esecuzione dell'sql.

5.4 Impostazione subfolder della cartella webroot

Per poter permettere al sistema MVC di funzionare correttamente deve sapere in quale percorso (subfolder) si trova della webroot (cartella htdocs).

Per impostare correttamente questo percorso basta aprire con un qualsiasi editore di testo il file .htaccess presente nella cartella PizzaDelivery ed andare a modificare la seguente riga dove si trova questo percorso di default:

```
# When using the script within a sub-folder, put this path here, like /mysubfolder/
# If your app is in the root of your web folder, then leave it commented out
RewriteBase /php/Progetti/PizzaDelivery/src/PizzaDelivery/
```

Bisogna immettere il percorso del file .htaccess rispettivamente alla cartella htdocs del proprio WebServer:

```
RewriteBase /Il/Tuo/Percorso/dalla/cartella/htdocs
```

Se si mette direttamente la cartella PizzaDelivery nella htdocs il RewriteBase risulterà così:

```
RewriteBase /PizzaDelivery/
```

5.5 Modifica percorso URL del progetto

Per poter effettuare tutte le richieste correttamente all'interno della struttura MVC bisogna impostare la costante "URL" che si trova nel file di configurazione nel seguente percorso:

PizzaDelivery/application/config/config.php

Una volta aperto il file con un qualsiasi editor di testo possiamo andare nella riga dove viene dichiarato l'URL e modificare il suo valore.

```
/**
 * Configurazione di : URL del progetto
 */
define('URL', 'http://localhost:8888/php/Progetti/PizzaDelivery/src/PizzaDelivery/');
```

Esempio di configurazione.

Se si vuole rendere accessibile l'applicazione da ovunque digitare (consigliato per questo progetto):

```
define('URL', 'http://<ip pubblico>:<porta apache>/<percorso dalla cartella htdocs, uguale a quello immesso precedentemente>');
```

Se invece si vuole solamente provarla senza dar la possibilità ad altri utenti di raggiungerlo digitare:

```
define('URL', 'http://localhost:<porta>/<percorso dalla cartella htdocs, uguale a quello immesso precedentemente>');
```

Un risultato finale potrebbe essere il seguente se mettiamo la cartella PizzaDelivery direttamente nella htdocs:

```
define('URL', 'http://10.20.4.145:8080/PizzaDelivery/');
```


5.6 Configurazione accesso MySQL per interagire con il DataBase

Come ultima operazione essenziale per il funzionamento dell'applicazione dobbiamo andare nella classe `database.php` che è quella che conetterà tutta la parte PHP del programma con il Database e modificare i valori delle costanti.

Il file si trova nel seguente percorso: *PizzaDelivery/application/database/database.php*

Le costanti da modificare sono:

- **HOST**: IP pubblico della macchina sulla quale è presente il DB, oppure "localhost" se è la stessa del WebServer.
- **USERNAME ***: Username dell'utente con il quale verranno effettuate le query sul DataBase.
- **PASSWORD ***: Password dell'utente
- **DATABASE ***: Nome del DataBase
- **PORT**: Porta sulla quale ascolta il servizio MySQL

* Campi già impostati di default che rispettano la struttura del programma, sconsigliato modificarli.

5.7 Primo accesso tramite interfaccia grafica

Una volta essendosi assicurati di aver effettuato tutti i passi correttamente del capitolo 5 possiamo accedere al nostro programma tramite browser, per far ciò basta andare nel percorso nel quale si ha piazzato la cartella PizzaDelivery.

Una volta raggiunto l'index con <http://<serverWebRoot>/PizzaDelivery> possiamo andare sulla voce "login" nella barra di navigazione ed entrare con il seguente utente amministratore:

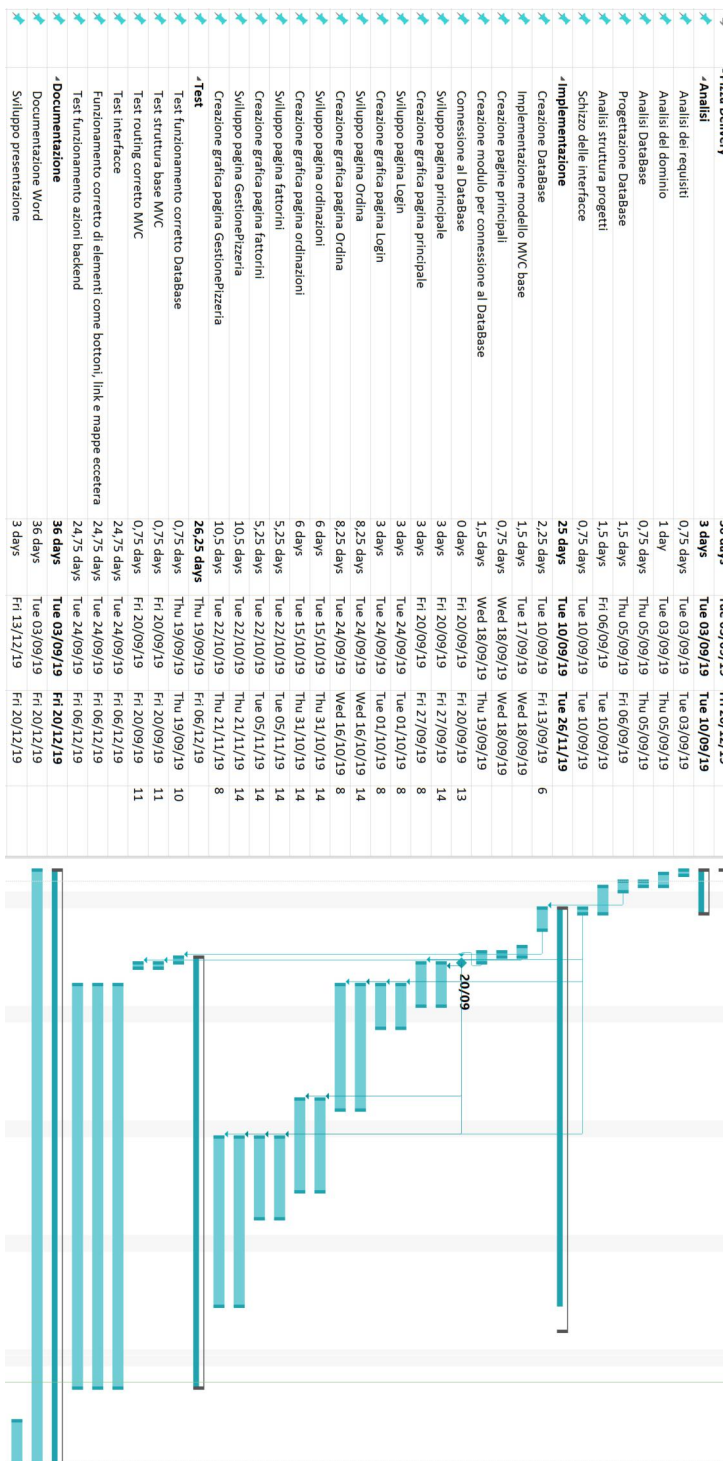
- Username: admin.pizzadelivery
- Password: test

ATTENZIONE: Una volta eseguito il login è vivamente consigliato andare nella pagina Gestione Pizzeria e premere il bottone "Crea Utente", successivamente creare un nuovo utente di tipologia amministratore ed eliminare admin.pizzadelivery oppure cambiarli la password.

Con l'account amministratore è possibile allestire il programma come meglio si preferisce, aggiungendo nuovi utenti ed articoli.

Consuntivo

Come si può vedere dal Gantt consuntivo le tempistiche sono principalmente state rispettate. L'unica parte che ha subito delle modifiche è quella dell'implementazione essendo che lo sviluppo di certe pagine richiedeva più tempo che quello di altre. Inoltre sono anche stati anticipati e prolungati le fasi di test, dopo ogni modifica che effettuavo eseguivo una prova del suo funzionamento oltre ad aver effettuato nuovamente tutti i test alla fine dell'implementazione.



6 Conclusioni

Questa tipologia di applicazione è già presente sul mercato e viene anche adoperata da molte pizzerie. Purtroppo ogni volta che un ristorante vuole offrire un servizio come la consegna di pizze a domicilio devono contattare programmatori oppure aziende che sviluppino questo software esclusivamente per loro dovendo pagare enormi somme che in molti casi non sono possibili finanziare.

Attraverso il prodotto PizzaDelivery diventa possibile gestire in modo semplice la propria pizzeria per quanto riguarda il lato delle consegne e ordinazioni, inoltre rimane anche molto intuitivo il processo di ordinazione per una tipologia qualsiasi di utente.

Infine il vantaggio che offre questo software è quello di poter essere acquistato da svariate pizzerie dando poi l'opzione di personalizzarlo ma soprattutto ad un costo molto più accessibile rendendolo molto competitivo sul mercato.

6.1 Sviluppi futuri

Per quanto riguarda gli sviluppi futuri ci sarebbero varie cose che si potrebbero aggiungere oppure migliorare.

Inizialmente sarebbe da implementare l'algoritmo di pathfinding per i percorsi dei vari fattorini guadagnando tempo e denaro (meno sprechi di benzina). Inoltre restando sempre sui fattorini si potrebbero aggiungere dei sensori di posizione su ogni veicolo oppure sul fattorino stesso per poter aggiornare in tempo reale la mappa del sito fornendo dati attuali e precisi.

Un'ulteriore aggiunta potrebbe essere l'implementazione del pagamento online evitando così lo scambio di denaro tra cliente e fattorino centralizzando il tutto su pagamenti online.

Si potrebbe anche aggiungere l'opzione di far creare un account ai clienti evitando così di dover sempre immettere i propri dati personali e quelli della carta di credito.

Infine per evitare la ricezione di comande eseguite da script l'aggiunta di una qualche tipologia di captcha potrebbe risolvere la problematica.

6.2 Considerazioni personali

Personalmente ho trovato molto interessante accettare questa sfida essendo che era il mio primo progetto di queste dimensioni, di conseguenza era qualcosa di nuovo anche per me.

Durante il percorso di progettazione ed implementazione ho imparato moltissime cose nuove potendo consolidare varie nozioni apprese già precedentemente nei vari moduli seguiti negli scorsi 3 anni oltre ad arricchire le mie conoscenze.

Ho anche capito cosa vuol dire dover progettare e successivamente implementare un progetto abbastanza grande se uno di dimensioni relativamente piccole come questo mi ha già fatto pensare a tantissime cose, casi d'uso e messo vari dubbi che senza una buona riflessione avrei magari svolto in un modo sbagliato.

Sono molto contento di aver scelto questo progetto e anche dei risultati ottenuti.

7 Bibliografia

7.1 Sitografia

- <https://www.php.net/manual/en/>, *Manuale ufficiale di PHP*, 24-09-2019 - 22-11-2019.
- <https://stackoverflow.com/>, StackOverFlow, 24-09-2019 - 22-11-2019.
- <https://www.w3schools.com/php/>, W3Schools PHP, 24-09-2019 - 22-11-2019.
- <https://www.w3schools.com/js/>, W3Schools Java Script, 24-09-2019 - 22-11-2019.
- <https://www.w3schools.com/bootstrap/>, W3Schools Bootstrap, 24-09-2019 - 22-11-2019.
- <https://getbootstrap.com/docs/4.3/getting-started/introduction/>, *Manuale ufficiale di Bootstrap*, 24-09-2019 - 22-11-2019.
- <https://api.jquery.com/>, *Manuale ufficiale di jQuery*, 24-09-2019 - 22-11-2019.
- <https://xdebug.org/docs/remote>, *Manuale ufficiale di xDebug*, 15-11-2019 - 22-11-2019.
- <https://www.draw.io/>, Sito per creare tutti i diagrammi, 24-09-2019 - 22-11-2019.
- <https://docs.mapbox.com/mapbox-gl-js/api/>, Documentazione delle mappe utilizzate nel progetto, 24-09-2019 - 22-11-2019.
- https://www.ilovepdf.com/it/unire_pdf, Sito per unire svariati PDF utilizzato per i diari, 13-12-2019

8 Allegati

- Abstract
- Documentazione del programma (con istruzione di installazione ambiente + installazione xDebug)
- Quaderno Dei Compiti
- Diari di lavoro
- Codice sorgente (sulla piattaforma GitHub: <https://github.com/JariNaeser/PizzaDelivery>)