
Candidato: Peter Catania
Azienda: SAMT
Periodo: 03/09/2019
Presentazione: 20/12/2019

Situazione iniziale

Lo scopo di questo progetto è di creare un'applicazione web che permetta di gestire le fatturazioni di una piccola azienda. Questo progetto permette di gestire tutte le informazioni delle fatture e le fatture stesse. All'applicazione possono accedere gli utenti normali che possono solo visualizzare le fatture. Può accedere anche l'amministratore che invece può gestire tutte le fatture e gli utenti presenti sul sito web.

Attuazione

Per questo progetto utilizzo la struttura web MVC (Model View Controller), questa mi ha permesso di separare chiaramente il codice e creare una struttura chiara. Per le interfacce ho utilizzato una libreria che implementa bootstrap, per avere un sito con un front-end semplice ed essere responsive. Ho iniziato con l'implementazione della gestione utenti, e infine la gestione delle informazioni delle fatture.

Risultati

Gli obiettivi prefissati sono stati in parte raggiunti, si possono gestire le informazioni delle fatture. Non si possono però creare le fatture o modificarle. Si possono visualizzare le fatture sia come utente che come amministratore. Le pagine web hanno accesso limitato e solo l'amministratore può accedere a tutte.

Gestione fatturazioni

Titolo del progetto: Gestione fatturazioni
Alunno/a: Peter Catania
Classe: I4AC
Anno scolastico: 2019/2020
Docente responsabile: Massimo Sartori

1	Introduzione	3
1.1	Informazioni sul progetto	3
1.2	Abstract.....	3
1.3	Scopo	3
2	Analisi	4
2.1	Analisi del dominio.....	4
2.2	Analisi e specifica dei requisiti.....	4
2.3	Use case	10
2.4	Pianificazione	12
2.5	Analisi dei mezzi	13
2.5.1	Software	13
2.5.2	Hardware.....	13
3	Progettazione.....	14
3.1	Design dei dati e database.....	14
3.1.1	Descrizione tabelle del database	14
3.1.2	Schema struttura dati del database.....	18
3.1.3	Diagramma ER del database	19
3.2	Design delle interfacce	20
3.2.1	Design struttura fattura/richiamo	20
3.2.2	Design interfaccia accesso	22
3.2.3	Design interfaccia registrazione.....	24
3.2.4	Design interfaccia creazione fattura/richiamo	25
3.2.5	Design lista fatture/richiami e filtri.....	27
3.2.6	Design interfaccia gestione clienti	29
3.2.7	Design interfaccia gestione prodotti	31
3.2.8	Design interfaccia gestione utenti	32
3.3	Design delle classi	33
3.3.1	UML classi model.....	33
3.3.2	UML classi controller	34
3.4	Design procedurale	35
4	Implementazione	36
4.1	Classi di validazione dati	36
4.1.1	Classe Model Validator.....	36
5	Test.....	42
5.1	Protocollo di test	42
5.2	Risultati test	48
5.3	Mancanze/limitazioni conosciute	49
6	Consuntivo	49
7	Conclusioni	50
7.1	Sviluppi futuri	50
7.2	Considerazioni personali	50
8	Sitografia	51
9	Allegati	51

1 Introduzione

1.1 Informazioni sul progetto

Progetto: Gestione fatturazioni

Docente responsabile: Massimo Sartori

Componenti del gruppo: Peter Catania

Luogo di lavoro: Aula 427 Scuola arti e mestieri Trevano

Classe: I4AC

Materia: Progetti individuali

Data di Inizio: 3/09/2019

Data di fine: 20/12/2020

1.2 Abstract

As the size of a company increases their invoices augment as well. The small companies save their invoice on their personal's computers and almost of the time they will end up all disordered in some random directory. This companies with their increase their clients increase as well, like the products sold and the consequent increase of the invoices. This invoice with no organization will end up unrecoverable or really difficult to searching the one you're looking for. This site is meant for fix this problem and simplify the reparability of invoices and their data; they will be available through anywhere you won't. The invoices data is stored and managed on a server accessible from website.

1.3 Scopo

Lo scopo di questo progetto è di creare un sito web che permette di gestire le fatturazioni di una piccola azienda. Il sito permette agli utenti normali di visualizzare le fatture, invece l'amministratore è quello che nel sito web si occupa di gestire tutte le fatture e tutto quello che le riguarda, prevalentemente le informazioni delle fatture e gli utenti registrati sul sito.

Il sito permette all'Amministratore di gestire facilmente le fatture, grazie anche a delle funzioni utili a velocizzare o semplificare il lavoro. La interfaccia utente deve essere ideata per semplificare la navigazione e l'utilizzo del sito.

2 Analisi

2.1 Analisi del dominio

Una azienda vuole un sito web per gestire le fatture da mandare ai propri clienti.

A questo sito devono avere l'accesso più utenti e deve essere accessibile sono agli impiegati autorizzati.

L'azienda ogni anno manda diverse fatture ai suoi clienti, quindi il sito deve permettere di memorizzarle tutte in maniera efficace, ma anche permettere di ricercarle quando ce ne bisogno, visto che alcune volte i clienti si possono dimenticare di pagare le fatture, e quindi il sito deve permettere anche di mandare i richiami.

Il problema principale è che le fatture vengono salvate e ricercate manualmente, e per risolvere questo problema si è preferito implementare una soluzione personalizzata, anche se esistono altre soluzioni simili come applicativi specifici per le fatture.

Il sito deve essere utilizzabile da chiunque, quindi deve avere un'interfaccia intuitiva.

2.2 Analisi e specifica dei requisiti

ID: REQ-01	
Nome	Creazione Pagina Login
Priorità	1
Versione	1.0
Note	<p>Quando un utente entra nel sito deve come prima cosa potersi registrare, poi a dipendenza di che tipo di utente si tratta potrà accedere a diverse pagine.</p> <p>L'Amministratore può vedere tutto, invece un Utente normale deve poter solo visualizzare le fatture.</p>
Sotto requisiti	
001	Deve esserci una pagina per il login a sestante dalle altre.
002	Deve esserci la possibilità di loggarsi sia come utente che come amministratore.

ID: REQ-02	
Nome	Creazione Pagina Registrazione
Priorità	1
Versione	1.0
Note	<p>La pagina di registrazione deve essere accessibile alla visualizzazione del sito</p> <p>La registrazione iniziale viene solo effettuata dagli utenti, l'amministratore è già presente nel sistema e non può</p>

essere eliminato.

Sotto requisiti

001

Deve esserci una pagina per la registrazione a sestante dalle altre.

ID: REQ-03	
Nome	Creazione pagina gestione utenti
Priorità	1
Versione	1.0
Note	L'Amministratore può accedere a questa pagina ma gli utenti no.
Sotto requisiti	
001	Deve esserci una pagina per la gestione utenti a sestante dalle altre.
002	Devono essere visualizzati tutti gli utenti salvati.
003	Devono essere presenti le funzionalità base sugli utenti: creazione, modifica ed eliminazione.

ID: REQ-04	
Nome	Creazione pagina gestione prodotti
Priorità	1
Versione	1.0
Note	L'Amministratore può accedere a questa pagina ma gli utenti no.
Sotto requisiti	
001	Deve esserci una pagina per la gestione prodotti a sestante dalle altre.
002	Devono essere visualizzati tutti i prodotti salvati.
003	Devono essere presenti le funzionalità base sui prodotti: creazione, modifica ed eliminazione.

ID: REQ-05	
Nome	Creazione pagina gestione clienti
Priorità	1
Versione	1.0
Note	L'Amministratore può accedere a questa pagina ma gli utenti no.
Sotto requisiti	
001	Deve esserci una pagina per la gestione clienti a sestante dalle altre.
002	Devono essere visualizzati tutti i clienti salvati.
003	Devono essere presenti le funzionalità base sui clienti: creazione, modifica ed eliminazione.

ID: REQ-06	
Nome	Creazione pagina gestione fatture
Priorità	1
Versione	1.0
Note	L'Amministratore può vedere tutto, invece un Utente normale deve poter solo visualizzare le fatture e non poter eseguire nessuna azione su di esse.
Sotto requisiti	
001	Deve esserci una pagina per la gestione fatture a sestante dalle altre.
002	Devono essere visualizzati tutte le fatture salvate.
003	Devono essere presenti le funzionalità base sulle fatture: creazione, modifica ed eliminazione.

ID: REQ-07	
Nome	Implementazione funzionalità richiamo di una fattura
Priorità	1
Versione	1.0
Note	Quando una fattura è in ritardo, verrà notificata che è in ritardo e ci sarà la possibilità di mandare un richiamo, se scade anche il richiamo ci sarà la possibilità di mandarne un altro, e così via.
Sotto requisiti	
001	Deve essere presente una notifica che indichi una fattura/richiamo scaduto.
002	Deve essere presente la possibilità di mandare un richiamo.

ID: REQ-08	
Nome	Implementazione funzionalità stampa fatture/richiami
Priorità	1
Versione	1.0
Note	Deve essere sono accessibile all' Amministratore . Quando una fattura/richiamo è stato creato dovrà esserci la possibilità di stamparlo.
Sotto requisiti	
001	Deve essere presente un bottone di stampa, che se schiacciato esporta la fattura/richiamo in formato PDF.
002	Deve essere visibile il numero di stampe effettuate di ogni singola fattura/richiamo.

ID: REQ-09	
Nome	Implementazione funzionalità ricerca fatture/richiami
Priorità	1
Versione	1.0
Note	<p>Deve essere sono accessibile all'Amministratore.</p> <p>Dove è presente la lista delle fatturazioni/richiami, devono essere presenti dei filtri per poter ricercare una certa fattura.</p>
Sotto requisiti	
001	Deve essere presente un filtro, che filtra le fatture per la data specificata.
002	Deve essere presente un filtro, che filtra le fatture per cliente.
003	Deve essere presente un filtro, che filtra le fatture per importo minimo e massimo.

ID: REQ-10	
Nome	Implementazione segnalazione fattura già pagata
Priorità	1
Versione	1.0
Note	
Sotto requisiti	
001	Deve essere possibile segnalare se una fattura è già stata pagata.
002	Deve essere possibile salvare la data di pagamento di una fattura.

ID: REQ-11	
Nome	Implementazione funzionalità copia fatture
Priorità	2
Versione	1.0
Note	Deve essere sono accessibile all' Amministratore . Una fattura può essere copiata per farne una simile.
Sotto requisiti	
001	Deve essere possibile copiare una fattura, che è già stata salvata

2.3 Use case

Al sito possono accedere gli **User** e l'**Administrator**, ma gli **User** sono limitati a visualizzare solo le fatture/richiami.

Gli **User** prima di entrare nel sito si devono registrare.

L'**Administrator** può per prima cosa aggiungere/gestire le fatture, e solo dopo potrà stampare le fatture oppure mandare un richiamo.

Una volta che una fattura è stata stampata, è l'**Administrator** che si occuperà di spedirla al cliente.

L'**Administrator** si deve anche occupare di registrare i clienti dentro il sistema.

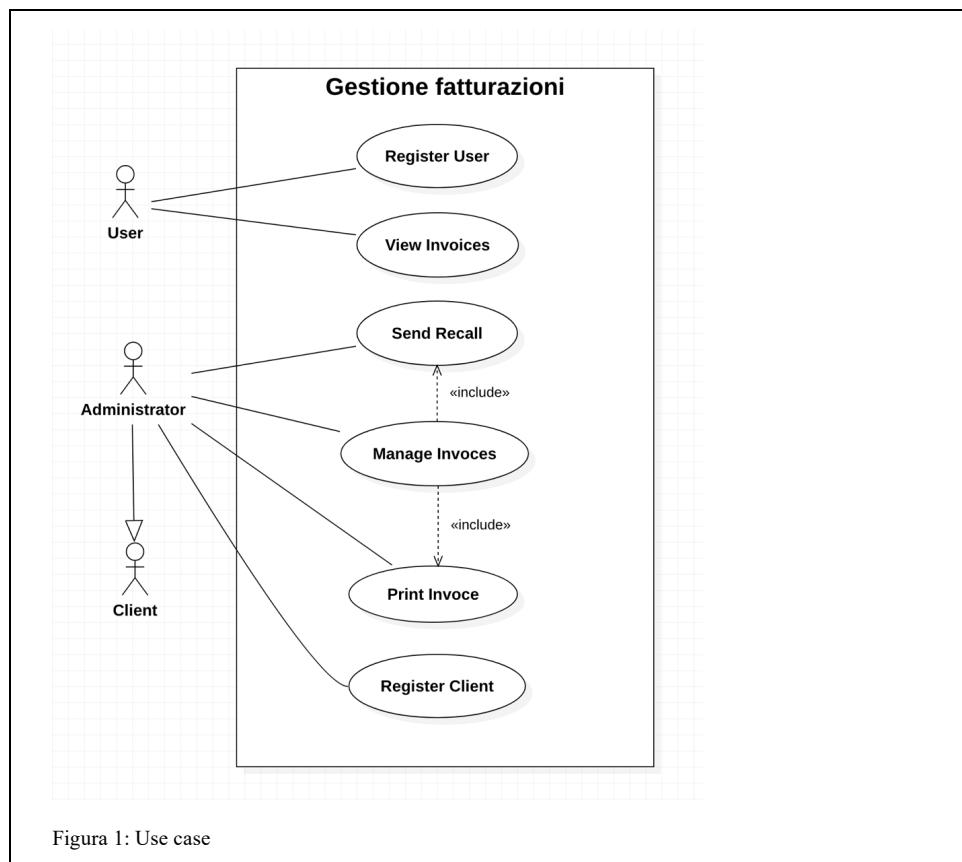


Figura 1: Use case

2.4 Pianificazione

La prima fase del progetto è quella di analisi, dove per prima cosa si esegue un'analisi del QDC per vedere se ci sono domande, se ce ne sono allora si richiede le risposte.

Dopo si esegue l'analisi di Dominio e per finire l'analisi dei requisiti.

La seconda fase è quella di progettazione, dove si comincia con quella di sistema, poi quella del Database, la progettazione delle classi e infine la progettazione dell'interfaccia utente.

La terza fase inizia con l'implementazione del DB, poi con l'implementazione delle classi per finire con l'implementazione dell'interfaccia grafica.

Finita l'implementazione si deve testare il tutto, cominciando con il test del DB e poi procedendo con il test delle pagine web, dopo la sistemazione dei problemi riscontrati si può dire che il progetto è pronto per la consegna.

Alla fine del progetto si crea la presentazione del prodotto e si prepara il tutto per la consegna, compresi i diari e la documentazione che sono stati redatti durante tutta la durata del progetto.

Il Gantt preventivo è presente anche come allegato in un formato più leggibile.

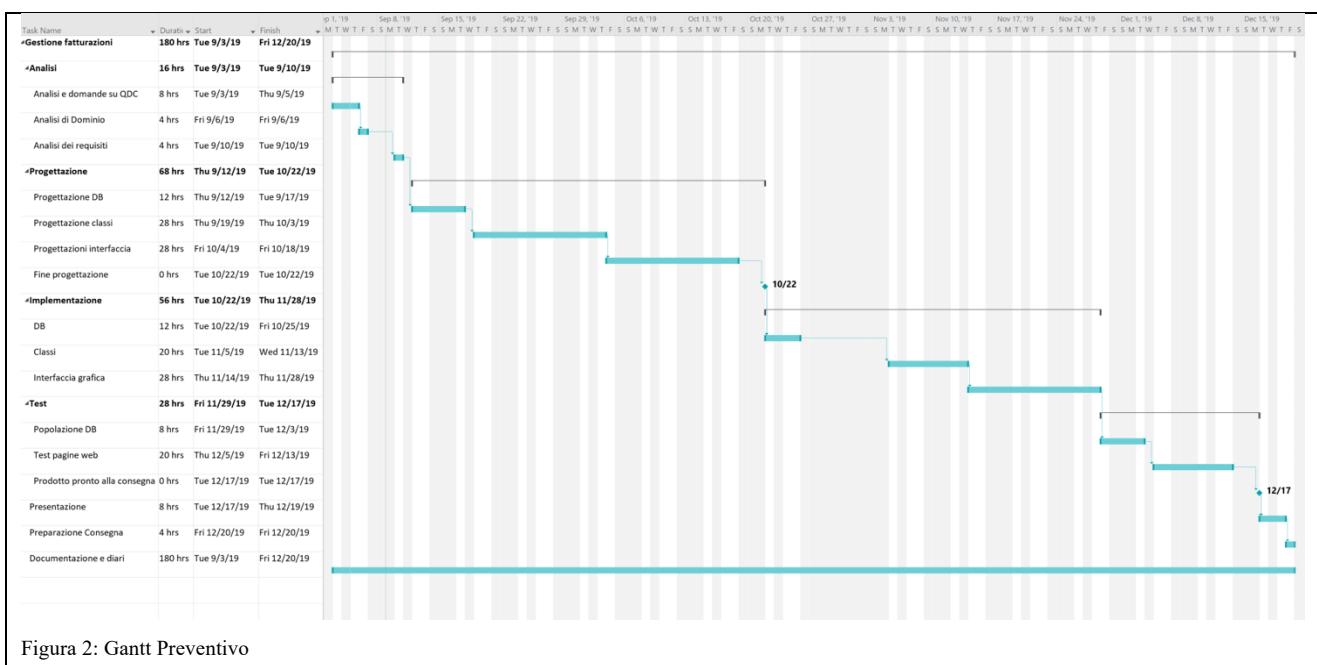


Figura 2: Gantt Preventivo

2.5 Analisi dei mezzi

2.5.1 Software

Sistema Operativo

- OSX 10.15 Catalina

Programmi

- PHPStorm 2019.2.3 (IDE)
- Microsoft Project 2016 (Creazione della pianificazione)
- Microsoft Word 2016 (Documentazione)
- Sublime Text 3 (Editor di testo)
- InfinityDesigner 1.6.1 (Progettazione Grafica)
- OmniGraffle 7.11.2 (Diagrammi di flusso, Schemi Database, Design Interfacce)

Software

- Apache 2.4.42 (Unix)
- PHP 7.4.1
- MySql 8.0.18

2.5.2 Hardware

Per svolgere questo progetto ho utilizzato il mio portatile, con le seguenti specifiche:

Apple MacBook Pro 15" 2017, Processore Intel® Core I7, RAM 16GB, 512GB SSD.

3 Progettazione

3.1 Design dei dati e database

3.1.1 Descrizione tabelle del database

La tabella **client** contiene tutte le informazioni di un cliente.

Client		
Nome del Dato	Tipo	Descrizione
id	(PK) int	Identificativo di questa tabella.
name	varchar(50)	Nome del cliente.
surname	varchar(50)	Cognome del cliente.
street	varchar(100)	Via appartenente dell'indirizzo di fatturazione del cliente.
house_no	varchar(6)	Numero civico appartenente all'indirizzo di fatturazione del cliente.
telephone	varchar(15)	Telefono del cliente.
email	varchar(100)	Email del cliente.
city_id	(FK) varchar(50)	Identificativo della città associata, appartenente all'indirizzo di fatturazione dell'azienda.

La tabella **client_company** contiene il nome dell'azienda del cliente, se si tratta di un **cliente giuridico**.

Client_company		
Nome del Dato	Tipo	Descrizione
id	(PK) int	Identificativo di questa tabella.
name	varchar(50)	Nome dell'azienda del cliente.
client_id	(FK) int	Identificativo del cliente associato.

La tabella **typology** contiene le varie tipologie di fatture/richiami.

Typology		
Nome del Dato	Tipo	Descrizione
id	(PK) int	Identificativo di questa tabella.
name	varchar(100)	Il nome della tipologia.

La tabella **product** contiene tutte le informazioni dei prodotti contenuti in una fattura/richiamo.

Product		
Nome del Dato	Tipo	Descrizione
id	(PK) int	Identificativo di questa tabella.
description	varchar(100)	Breve descrizione del prodotto.
price	decimal(19,4)	Prezzo del prodotto.

La tabella **Invoice** contiene tutte le informazioni di una fattura/richiamo.

Invoice		
Nome del Dato	Tipo	Descrizione
id	(PK) int	Identificativo di questa tabella.
print_no	int	Numero di stampe di una fattura/richiamo effettuate.
payment_date	date	Data del giorno in cui la fattura/richiamo è stata pagata.
creation_date	date	Data del giorno in cui è stata creata la fattura/richiamo.
status	varchar(7)	Specifica lo stato della fattura, può avere diversi valori: saved; paid; expired.
callback	tinyint	Specifica se si tratta di una fattura o di un richiamo: 0 = Fattura; 1 = Richiamo.
client_id	(FK) int	Identificativo del cliente associato.
typology_id	(FK) varchar(100)	Identificativo della tipologia associata.

La tabella **inserted** contiene le informazioni, che ci sono quando è presente un'associazione di un prodotto con una fattura.

Inserted		
Nome del Dato	Tipo	Descrizione
product_id	(PK) int	Identificativo del prodotto associato.
invoice_id	(PK) int	Identificativo della fattura/richiamo associato.
sell_date	date	Data di quando è stata effettuata la vendita di un prodotto/servizio
quantity	int	La quantità venduta del prodotto associato con la fattura.

La tabella **company** contiene tutte le informazioni dell'azienda che emette fatture.

Company		
Nome del Dato	Tipo	Descrizione
id	(PK) int	Identificativo di questa tabella.
name	varchar(50)	Nome dell'azienda.
logo_path	varchar(100)	Il percorso sul server per poter ottenere il logo dell'azienda.
iban	varchar(30)	Il numero del conto dell'azienda.
street	varchar(100)	Via appartenete all'indirizzo di fatturazione dell'azienda.
house_no	varchar(6)	Numero civico appartenete all'indirizzo di fatturazione dell'azienda.
email	varchar(100)	Email dell'azienda.
site	varchar(100)	Sito web dell'azienda.
telefono	varchar(15)	Telefono dell'azienda.
city_id	(FK) varchar(50)	Identificativo della città associata, appartenete all'indirizzo di fatturazione dell'azienda.

La tabella **user** contiene tutte le informazioni di un utente del sito web, e se l’utente è stato abilitato dall’amministratore.

User		
Nome del Dato	Tipo	Descrizione
id	(PK) int	Identificativo di questa tabella.
username	varchar(50)	Nome dell’utente.
password	char(64)	Password dell’utente, codificata in sha256.
email	varchar(100)	Email dell’utente.
enabled	tinyint	Specifica se l’utente è stato abilitato dall’amministratore: 0 = utente non abilitato; 1 = utente abilitato.

La tabella **administrator** contiene tutte le informazioni dell’amministratore del sito web.

Administrator		
Nome del Dato	Tipo	Descrizione
id	(PK) int	Identificativo di questa tabella.
username	varchar(50)	Nome dell’amministratore.
password	char(64)	Password dell’amministratore, codificata in sha256.
email	varchar(100)	Email dell’amministratore.

La tabella **city** contiene, le informazioni di una città.

City		
Nome del Dato	Tipo	Descrizione
id	(PK) int	Identificativo di questa tabella.
name	varchar(50)	Il nome della città.
nap	int	Numero di avviamento postale.

3.1.2 Schema struttura dati del database

Questo diagramma ha la stessa funzione di uno schema logico e specifica quali tabelle sono presenti all'interno del database “**Invoices**”, e il nome e il tipo di ogni dato memorizzato al loro interno. Si può anche vedere le dipendenze fra le tabelle, e così si può capire quali tabelle non hanno nessuna dipendenza e quali meno, e questo aiuta durante la fase di implementazione.

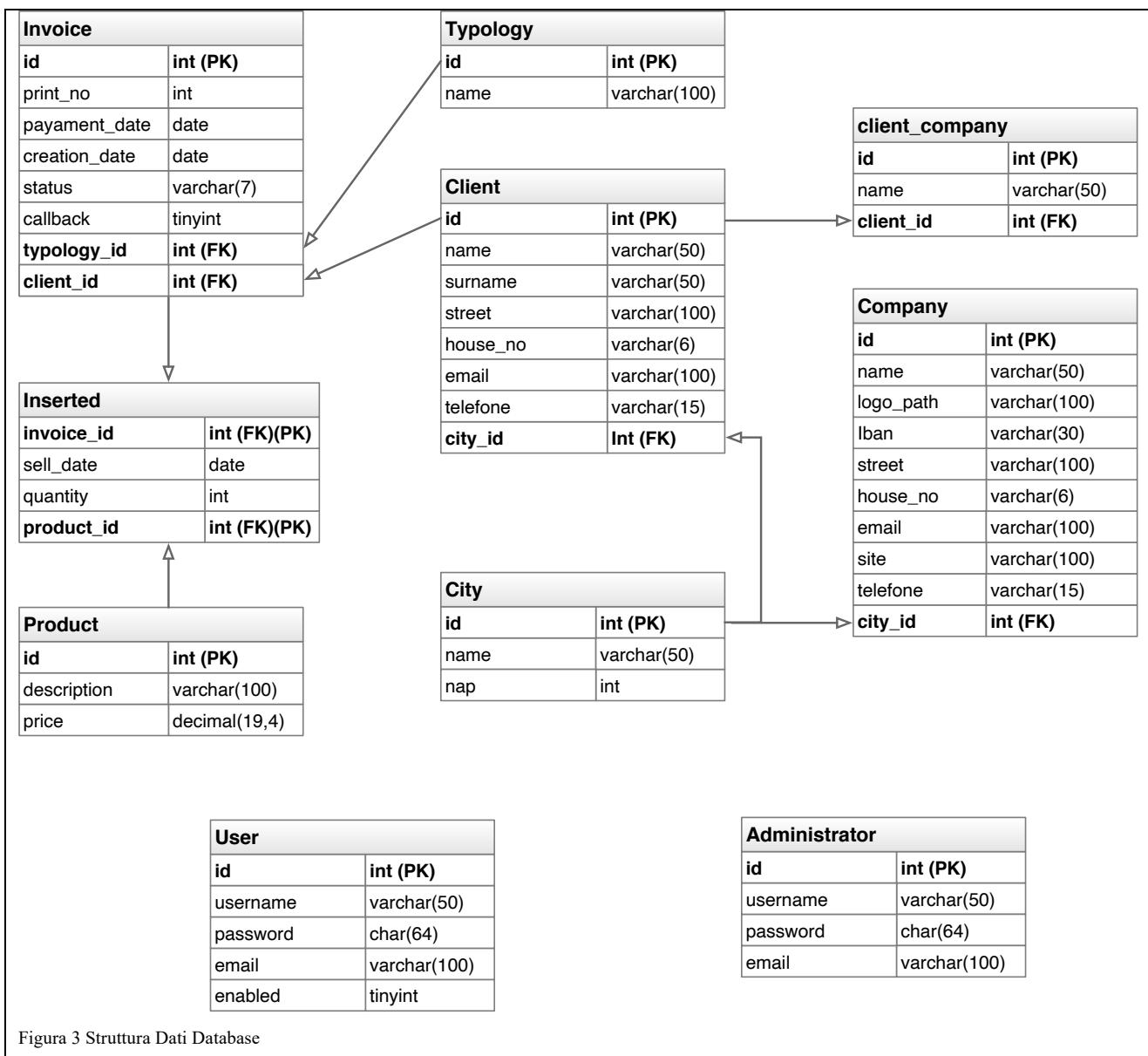


Figura 3 Struttura Dati Database

3.1.3 Diagramma ER del database

Questo è il diagramma ER che descrive il database, e fa vedere le varie tabelle come sono relazionate fra loro, ma anche le varie colonne che possiedono al loro interno.

Da questo schema si può comprendere come verranno inseriti i dati e a cosa serviranno.

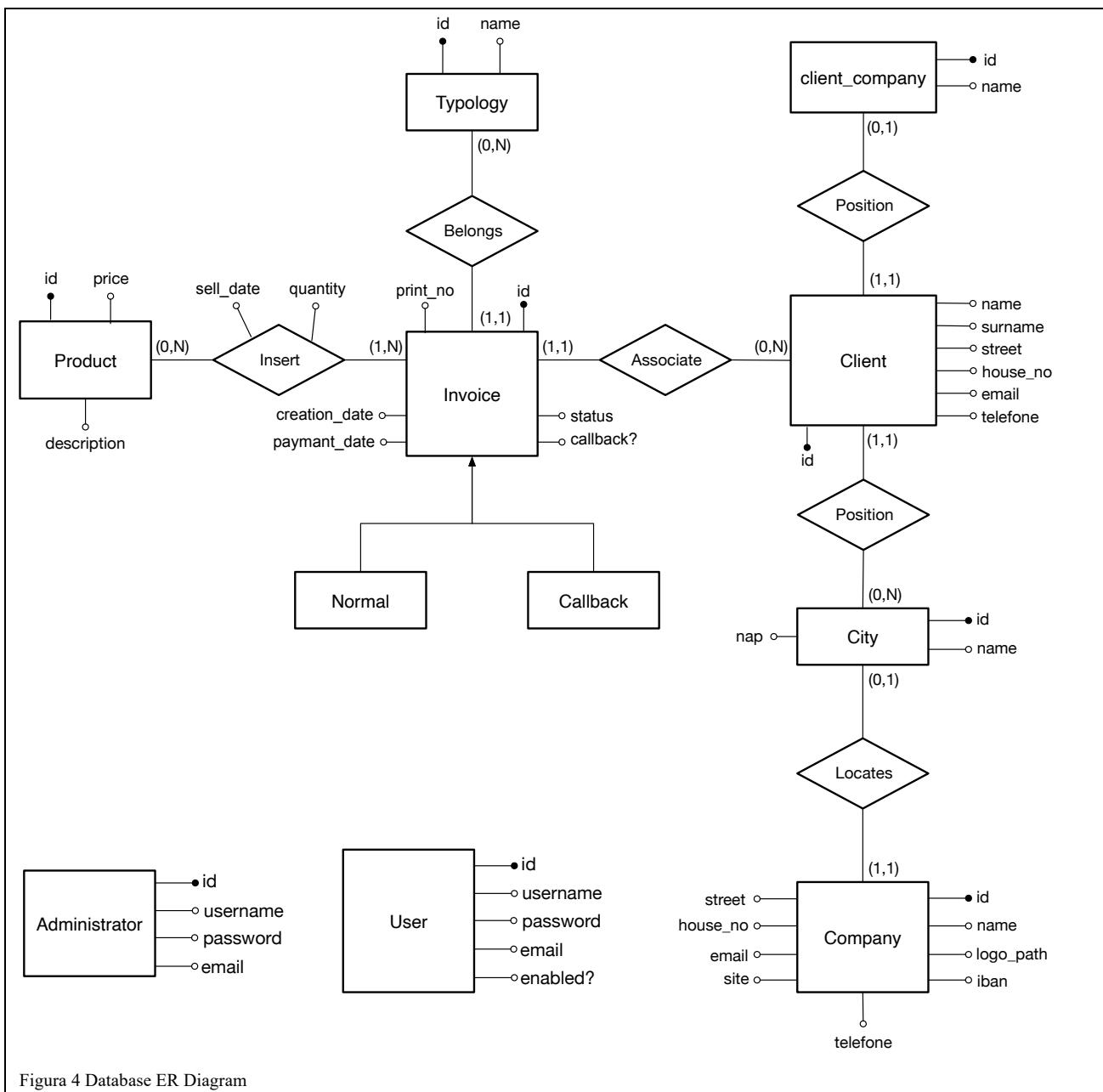


Figura 4 Database ER Diagram

3.2 Design delle interfacce

3.2.1 Design struttura fattura/richiamo

Per ogni fattura deve esserci il titolo “Fattura” e se si tratta di un richiamo il titolo “Richiamo N”, dove N è un numero variabile che identifica se si tratta del primo richiamo, secondo, ecc.

In parte è presente il logo dell’azienda e sotto di esso la data (quando è stata creata la fattura) e il numero della fattura.

Dopo sono presenti le informazioni dell’azienda.

Sotto sono presenti le informazioni del cliente, e successivamente una tabella con le informazioni dei prodotti/servizi offerti dall’azienda.

Infondo alla fattura c’è il totale, che è la somma di tutti gli importi di ogni singolo prodotto/servizio.

Fattura																															
Logo																															
<Nome azienda>		Data																													
<Indirizzo>		Fattura no.																													
<Telefono>																															
<Email, Web>																															
Cliente																															
<Nome del responsabile/ Nome cliente>																															
<Nome azienda>																															
<Indirizzo, NAP>																															
<Telefono>																															
<Email>																															
<table border="1"><thead><tr><th>Descrizione</th><th>Q.tà</th><th>Prezzo uni.</th><th>Importo</th></tr></thead><tbody><tr><td></td><td></td><td></td><td>0,00CHF</td></tr><tr><td></td><td></td><td></td><td>0,00CHF</td></tr><tr><td></td><td></td><td></td><td>0,00CHF</td></tr><tr><td></td><td></td><td></td><td>0,00CHF</td></tr><tr><td></td><td></td><td></td><td>0,00CHF</td></tr><tr><td></td><td></td><td></td><td>0,00CHF</td></tr></tbody></table>				Descrizione	Q.tà	Prezzo uni.	Importo				0,00CHF																				
Descrizione	Q.tà	Prezzo uni.	Importo																												
			0,00CHF																												
			0,00CHF																												
			0,00CHF																												
			0,00CHF																												
			0,00CHF																												
			0,00CHF																												
TOTALE _____ 0,00CHF																															

Figura 5: Design Struttura Fattura/Richiamo

3.2.2 Design interfaccia accesso

Quando si entra nel sito la prima scena che comparirà sarà questa, dove l'utente che entrerà potrà accedere con il proprio account, e se non possiede un account potrà registrarsi andando alla pagina di registrazione, schiacciando il link in blu “Registrati subito”.

Per accedere l'utente dovrà inserire il proprio **Username** e **Password**, e infine schiacciare il bottone **Accedi**. Se dopo aver schiacciato il bottone **Accedi** il sito trova che i dati inseriti non corrispondono o che non sono corretti, apparirà un errore sopra il campo con l'informazione sbagliata.

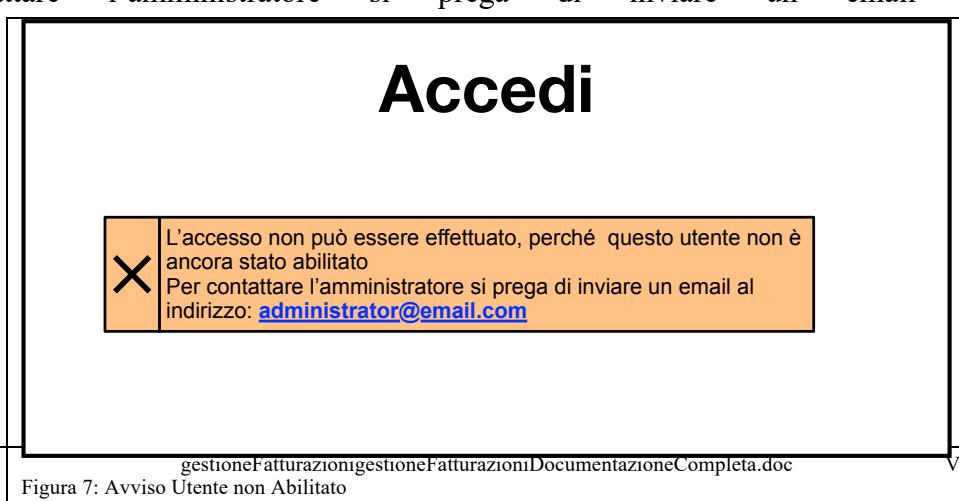
The screenshot shows a login form titled "Accedi". It has two input fields: "Username*" with placeholder "Bob..." and "Password*". Above the "Username" field, the word "Errore" is displayed in red. Below the fields is a blue link "Registrati subito". At the bottom is a large red button labeled "Accedi".

Figura 6: Design Interfaccia Accesso

Se un utente non è ancora stato accettato e prova a fare il login, non gli sarà permesso effettuarlo anche se le sue credenziali sono giuste, e sarà stampato un avviso:

“L’accesso non può essere effettuato, perché questo utente non è ancora stato abilitato

Per contattare l’amministratore si prega di inviare un email all’indirizzo:



administrator@email.com “

3.2.3 Design interfaccia registrazione

Quando un utente si vuole registrare andrà su questa pagina, all'interno sono presenti i campi di registrazione, e se si vuole andare alla pagina di **Accesso** basta schiacciare il link “Ho già un account”.

Una volta inserito tutti i dati correttamente, schiacciando il pulsante registrati l'utente sarà registrato, ma non potrà ancora accedere, e dovrà aspettare che l'**Administrator** gli dia il consenso.

The screenshot shows a registration form titled "Registrazione". It includes fields for "Username" (with an "Error" message), "Email", "Password", and "Conferma". A blue link "Ho già un account" is visible above the "Registrati" button. The "Registrati" button is highlighted with a red border.

Figura 8: Design Interfaccia Registrazione

3.2.4 Design interfaccia creazione fattura/richiamo

Questa interfaccia compare quando l'amministratore vuole creare una nuova fattura/richiamo, quando si tratta di una **fattura** il titolo sarà “Fattura” se no “Richiamo N” se si tratta di un **richiamo**.

In questa pagina le informazioni dell'azienda, la data e il numero della fattura saranno aggiunti in automatico, e le informazioni dell'azienda potranno essere anche modificati.

Tutte le altre informazioni saranno aggiunte manualmente da zero, a meno che si copia una fattura già esistente, in quel caso le informazioni saranno già tutte presenti e saranno solo da modificare se necessario.

Fattura

Path

<Nome azienda>

<Indirizzo>

<Telefono>

<Email, Web>

Data

Fattura no.

Tipologia: <Tipologia>

Cliente

<Nome del responsabile/ Nome cliente>

<Nome azienda>

<Indirizzo, NAP>

<Telefono>

<Email>

Descrizione	Q.tà	Prezzo uni.	Importo
			0,00CHF
			0,00CHF
			0,00CHF
			...

TOTALE 0,00CHF

Salva

Figura 9: Design Interfaccia Creazione Fattura/Richiamo

Peter Catania

gestioneFatturazionigestioneFatturazioniDocumentazioneCompleta.doc

Versione: 15.10.2019

3.2.5 Design lista fatture/richiami e filtri

Questa interfaccia rappresenta come la lista di fatture/richiami sarà visualizzata, e mostra anche come saranno i filtri per le fatture. Le fatture/richiami saranno visualizzate in ordine di creazione, le fatture/richiami in ritardo saranno visualizzate all'inizio.

I filtri si trovano in alto alla pagina, e permettono di restringere le fatture visualizzate per **Data**, **Cliente** e **Importo**.

A sinistra delle fatture/richiami saranno visualizzate delle icone che indicano di che cosa si tratta ad esempio una **fattura in ritardo**, sono presenti anche delle icone a destra, che se schiacciate permettono di eseguire una funzione riguardante le fatture/richiami. C'è ne una che permette di stampare in pdf una fattura o un richiamo, una che duplica una fattura/richiamo, e una di colore rosso che viene mostrata solo se una fattura è in ritardo e permette se ciacciata di creare un richiamo.

Se non ci sono fatture da visualizzare, al posto della lista apparirà il messaggio “Per il momento non è stata memorizzata nessuna fattura/richiamo”.

Fatture

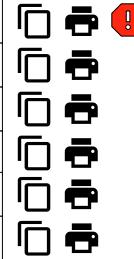
Filtri

Data: da al

Cliente: Numero Cliente
Marco Bernasconi, Via canale 2, 6600 Locarno

Importo: da a

Tipologia	Creazione	Pagata	No. Stampe	Importo
 Fattura Legna	21/07/19		5	50 CHF
 Richiamo cibo	24/07/19		5	80 CHF
 Fattura tubi	02/09/19		5	30 CHF
 Fattura tubi	13/04/19	06/10/19	5	20 CHF
 Fattura tegole	06/03/19	07/11/19	5	80 CHF
 Fattura dadi	03/03/19	06/10/19	5	22 CHF

Nuova Fattura

Figura 10: Design Lista Fatture/Richiami e Filtri

3.2.6 Design interfaccia gestione clienti

La gestione dei clienti è fatta in una pagina a parte, in questa pagina si possono aggiungere nuovi clienti e visualizzare tutti i clienti già memorizzati.

In alto è tutta una serie di campi che sono da riempire se si vuole creare un nuovo cliente. All'inizio c'è un *checkbox*, che se è spuntato significa che il cliente è un'azienda invece che un privato, se si tratta di un'azienda si dovrà anche riempire il campo **Nome Azienda** e se no il campo sarà disabilitato.

Per creare un nuovo cliente bisogna riempire tutti i campi richiesti, e infine schiacciare sul pulsante salva per aggiungere un nuovo cliente.

Se non ci sono clienti da visualizzare, al posto della lista apparirà il messaggio “Per il momento non è stato memorizzato nessun cliente”.

Clienti

Nuovo Cliente

Azienda:

Salva

Nome	Indirizzo	Telefono	Email
Rita Florence	Via arma 3, Lo...	+41 787344566	rita@gmail.com
GrandiAffari SA	Via nota 7, Lug...	+41 787344566	gr.f@gmail.com

Figura 11: Interfaccia Gestione Clienti

3.2.7 Design interfaccia gestione prodotti

La gestione dei prodotti è fatta su una pagina apposta, su questa pagina è presente una sezione in alto per aggiungere un nuovo prodotto e infondo la lista dei prodotti già creati.

Per creare un prodotto basta inserire tutte informazioni nei campi e infine schiacciare il tasto **Salva**.

Se non ci sono prodotti da visualizzare, al posto della lista apparirà il messaggio “Per il momento non è stato memorizzato nessun prodotto”.

Prodotti

Nuovo Prodotto

Descrizione ...

Prezzo 20 CHF...

Salva

Descrizione	Prezzo
Legname grezzo	120 CHF
Impalcature	200 CHF
Sacco di cemento 10Kg	60 CHF

Figura 12: Interfaccia Gestione Prodotti

3.2.8 Design interfaccia gestione utenti

Quando un utente si registra non può subito accedere, ma deve aspettare che l'amministratore abiliti il suo *account*, per fare questo l'amministratore può entrare in questa pagina.

In questa pagina gli utenti non abilitati verranno visualizzati tramite una lista, e per poterli abilitare basta mettere la spunta sul *checkbox Abilita*, e infine premere il pulsante **Salva**.

Utenti

Utenti non abilitati

Salva

Username	Email
Giulio.Protopet	giulio.proto@gmail.com
Gustavo.Inox	gusti.in@gmail.com
Davide.Umbri	d.umbri@gmail.com

Abilita
Abilita
Abilita

Figura 13: Interfaccia Gestione Utenti

Quando non ci sono utenti da abilitare, uscirà il messaggio “Non ci sono utenti da abilitare”.

Utenti

Utenti non abilitati

Salva

Non ci sono utenti da abilitare

Figura 14: Nessun utente da abilitare

3.3 Design delle classi

3.3.1 UML classi model

Questo UML specifica le classi che avranno il compito di Model (seguendo il pattern MVC), come si può vedere la maggior parte delle classi deve comunicare con il database, per questo queste classi hanno una connessione con il database (Invoices).

Tutte le classi possiedono i metodi necessari, che serviranno ad aiutare i Controller a svolgere tutte le operazioni necessarie.

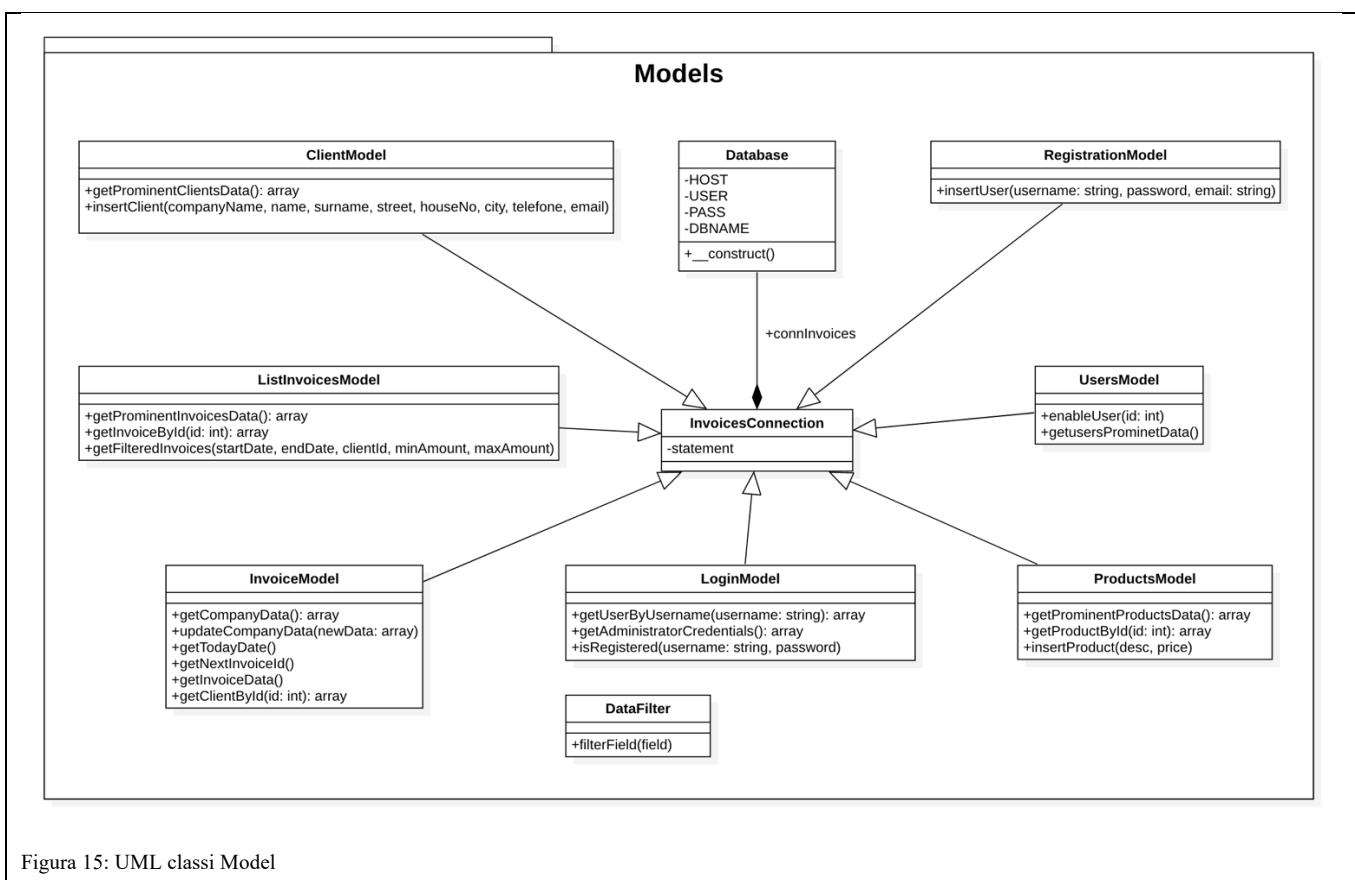
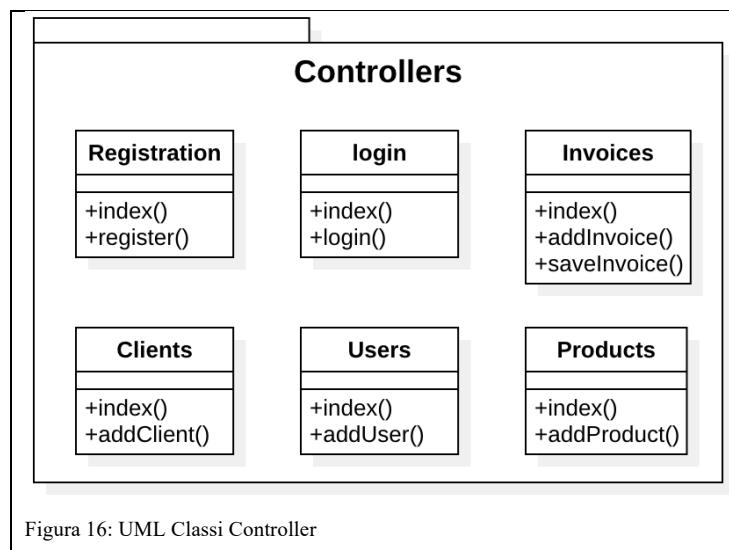


Figura 15: UML classi Model

3.3.2 UML classi controller

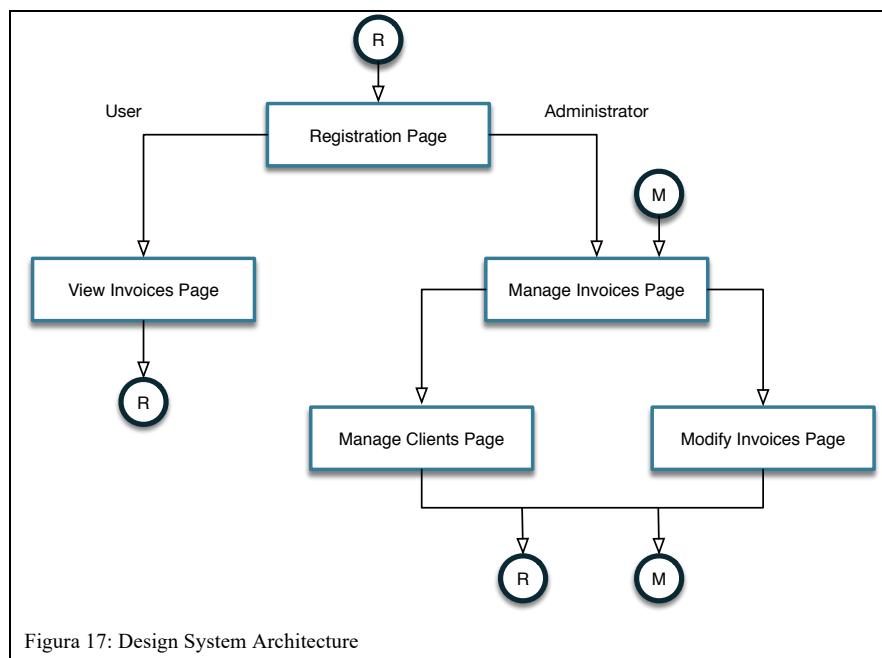
Questo UML specifica le classi che avranno il compito di Controller, cioè quelle che utilizzano le classi Model per effettuare delle operazioni su delle View (Pagine Web) precise. Ognuna di queste classi si occuperà di gestire le operazioni svolte su una pagina web, con l'eccezione di “Invoices” che si occupa sia di “[Interfaccia lista fatture/richiami](#)” sia di “[Interfaccia creazione fattura/richiamo](#)”.



3.4 Design procedurale

Questo è uno schema che mostra come logica in generale del sito, mostrando le fasi dalla prima all'ultima.

Sono descritte anche le differenti azioni che possono fare i diversi Utenti.



4 Implementazione

4.1 Classi di validazione dati

4.1.1 Classe Model Validator

La classe “Validator” fa parte delle classi [Model](#).

Al suo interno sono presenti diversi tipi di validazione sui dati o altri metodi che aiutano la validazione.

Questo metodo permette di filtrare in generale qualsiasi dato, togliendo qualsiasi carattere illegale che potrebbe comportare un inaspettato comportamento del sito.

```
/**  
 * Validate data of any kind.  
 *  
 * @param data The data to validate  
 * @return boolean The validated data  
 */  
private function generalValidation($data)  
{  
    $data = trim(stripslashes(htmlspecialchars($data)));  
    return $data;  
}
```

Questo metodo permette di sapere, se una email è valida strutturalmente e sintatticamente. Se ritorna “TRUE” allora significa che è valida, ma deve essere comunque filtrata in ogni caso.

```
/**  
 * Return true if the given email is valid, structurally and syntactically.  
 *  
 * @param email The email to check the validation  
 * @return boolean If the email is valid return true otherwise false  
 */  
function isValidEmail($email)  
{  
    return filter_var($email, FILTER_VALIDATE_EMAIL) &&  
        preg_match('/@.+\. /', $email);  
}
```

Questo metodo filtra i caratteri dell'email e se la filtrazione fallisce ritorna “FALSE”, invece se ci riesce ritorna il testo filtrato.

```
filter_var($email, FILTER_VALIDATE_EMAIL)
```

Questa parte invece verifica la struttura dell'email, verificando che i caratteri obbligatori ci siano e nella posizione giusta all'interno dell'email.

```
preg_match('/@.+\. /', $email);
```

Questo metodo ritorna l'email validata, grazie al metodo “filter_var(<variabile>,<tipoFiltro>)” che filtra ogni singolo carattere dell'email, e se la validazione fallisce ritorna “FALSE”.

```
/**  
 * Validate data of type string.  
 *  
 * @param email The email to validate  
 * @return boolean The validated email  
 */  
public function validateEmail($email)  
{  
    $validEmail = $this->generalValidation($email);  
    return filter_var($validEmail, FILTER_VALIDATE_EMAIL);  
}
```

Questo metodo permette di validare un numero intero, inserendo un qualsiasi numero ritorna un intero validato.

```
/**  
 * Validate data of type int.  
 *  
 * @param i The integer number to validate  
 * @return boolean The validated data  
 */  
public function validateInt($i)  
{  
    $validElement=$this->generalValidation($element);  
    return intval($validElement);  
}
```

Questo metodo permette di validare una stringa, inserendo un testo qualsiasi ritorna una stringa filtrata.

```
/**  
 * Validate data of type string.  
 *  
 * @param str The string to validate  
 * @return boolean The validated string  
 */  
public function validateString($str)  
{  
    $validStr = $this->generalValidation($str);  
  
    $pattern = '/^[-àèòùÀÈÒÙáéíóúýÁÉÍÓÚÝâéíóúÂÊÍÓÛñõÃÑÖäéöüýÄÉÍÖÜÝçÇßøÅåÆæœ]*$/';  
  
    if (!preg_match($pattern, $validStr))  
    {  
        $validStr = strval($validStr);  
    }  
    return $validStr;  
}
```

4.1.2 Funzioni base sulle informazioni

Per ogni pagina ho aggiunto dei buttoni che possono eseguire le informazioni principali sui dati del database. Le azioni principali che si possono eseguire con i buttoni sono la modifica, il salvataggio e l'eliminazione dei dati, invece per aggiungere i dati è presente un'interfaccia apposita all'inizio di ogni pagina.

Come esempio prendo la pagina per la gestione degli utenti, in questa pagina è presente un'interfaccia per aggiungere un nuovo utente e subito sotto la tabella con la lista degli utenti memorizzati sul database. Sopra alla tabella sono presenti i buttoni che permetto di salvare e modificare tutti insieme i dati presenti nel database. Invece in parte ad ogni riga sono presenti i buttoni per modificare, salvare e eliminare un utente. Se si schiaccia il bottone per modificare e dopo lo si rischiaccia i dati modificati non saranno salvati e i dati torneranno allo stato in cui erano prima della modifica. Se si schiaccia il bottone salva dopo la modifica i dati non si potranno più modificare fino a quando non si schiaccia di nuovo il bottone modifica. Per precauzione quando il bottone elimina viene schiacciato comparirà una piccola interfaccia, in cui si chiede la conferma per cancellare l'utente.

Nome Utente	Email	Abilitato
Sherryhalbara	sherryhalbara@gmail.com	<input type="checkbox"/>
RobertoMand	ro.manda@gmail.com	<input checked="" type="checkbox"/>

Queste azioni che vengono fatte sui dati sono state implementate in modo tale che quando vengono eseguite non si debba ricaricare la pagina, questo è stato possibile con l'utilizzo di AJAX per eseguire le azioni.

Le funzioni con **AJAX** sono state fatte in un file esterno apposito per ogni pagina. E in questo esempio permette di salvare le informazioni di un utente.

Questa funzione prende le informazioni che sono presenti all'interno della pagina, e li manda sotto forma di **JSON** ad un metodo presente nel controller della stessa view con un **POST**, in questo caso **updateUser** del controller **users.php**

Quando il metodo del controller è stato eseguito e arriva una risposta viene disabilitata la possibilità di modifica.

```
let URL = "<?= URL ?>";
/**
 * Update a user from its ID
 *
 * @param id The id of the user */
function updateUser(id) {
    var user = createUserArrayFromId(id); var jsonUser = JSON.stringify(user);
    $.ajax({
        type: "POST",
        url: URL + "users/updateUser", data: {
            user: jsonUser },
        success: function(response) { if (response) {
            $(".user-" + id + "-field").prop("disabled", true); }
        } });
}
```

Questo metodo aiuta il metodo precedente, permette di ricavare le informazioni di un certo utente dal suo ID, e le ritorna sotto forma di file JSON.

```
/**
 * Get the user representation with an JSON,
 * the JSON is mapped name => value. *
 * @param id The id of the user */
function createUserArrayFromId(id) { var user = {};
    $(".user-" + id + "-field").each(function(i, obj) {
        user[this.name] = this.value;
    });
    user["enabled"] = $("#enabled" + id).is(":checked");
    return user;
}
```

All'interno del metodo del controller che è stato richiamato dalla funzione AJAX, è presente il codice che permette di salvare i dati modificati all'interno del database. Riceve le informazioni tramite POST e subito dopo le salva utilizzando il metodo **updateUser()** all'interno della classe model principale del controller, in questo caso **UserModel**. Quello che ritorna questa funzione non viene letto ma serve alla funzione AJAX per sapere quando è stato eseguito il metodo.

```
**
 * Update a user,
 * with the informations contained in the upcoming POST request. *
 * @return void
 */
public function updateUser() {
    // the json containing the informations of the new user
    $user = json_decode($_POST['user'], true);
    $this->model('UserModel');

    $userModel = new UserModel();
    $userModel->updateUser($user['ids[]'], $user['usernames[]'], $user['emails[]'], $user['enabled']);

    echo "true";
}
```

4.1.3 Limitare l'accesso alle pagine

Nella classe Controller che fa parte delle classi centrali del pattern MVC, ho aggiunto alcuni metodi per permettere ad ogni controller di gestire alcune situazioni. Queste situazioni si presentano quando un metodo o più di un controller possono essere accessibili solo da particolari utenti, nel nostro caso sono presenti dei metodi che devono essere accessibili solo dall'amministratore.

Per fare questo ho creato questo metodo che consente di ritornare alla *homepage* se nessuno si è ancora registrato.

```
/**  
 * Check if the login data of a user or of the administrator * is saved in the session.  
 */  
private function existsLoginSessionData() {  
    return isset($_SESSION[USER_SESSION_DATA]) || isset($_SESSION[ADMINISTRATOR_SESSION_DATA]);  
}  
  
/**  
 * Redirect to the home page, if the login is not been effectuated by anyone.  
 */  
public function redirectToHomePagelfAnyonelsLogged() {  
    if (!$this->existsLoginSessionData()) {  
        header("Location: " . URL . "home/index");  
    }  
}
```

Ho implementato anche questo metodo, che permette di ritornare automaticamente alla *homepage*, se è registrato un **Utente** o nessuno si è ancora registrato.

```
/**  
 * Redirect to the home page, if is logged a user or anyone.  
 *  
 * @param session The current session saved on the server.  
 */  
public function redirectToHomePagelfUserOrAnyonelsLogged() {  
    if (  
        isset($_SESSION[USER_SESSION_DATA]) ||  
        !$this->existsLoginSessionData()  
    )  
        header("Location: " . URL . "home/index");  
}
```

5 Test

5.1 Protocollo di test

Test Case:	TC-001	Nome:	Controllo login
Riferimento:	REQ-01		
Descrizione:	Inserire credenziali per il login non corrette		
Prerequisiti:	Pagina di login, utente registrato nel database		
Procedura:	<ol style="list-style-type: none">1. Aprire il browser e immettere: http://localhost/Gestione-fatturazioni/src/invoices/home/index2. Inserire le credenziali, username e password non corretti3. Premere il bottone “Login”		
Risultati attesi:	Viene mostrato un messaggio di errore, comunica che le credenziali sono sbagliate.		

Test Case:	TC-002	Nome:	Controllo accesso alle pagine
Riferimento:	REQ-01		
Descrizione:	Effettuare il login con un utente e provare ad accedere alle pagine		
Prerequisiti:	Pagina di login, utente registrato nel database, tutte le pagine create		
Procedura:	<ol style="list-style-type: none">1. Aprire il browser e immettere: http://localhost/Gestione-fatturazioni/src/invoices/home/index2. Inserire le credenziali, username e password3. Premere il bottone “Login”4. Spostarsi sulla pagina: http://localhost/Gestione-fatturazioni/src/invoices/users/index5. Spostarsi sulla pagina: http://localhost/Gestione-fatturazioni/src/invoices/products/index6. Spostarsi sulla pagina: http://localhost/Gestione-fatturazioni/src/invoices/clients/index		
Risultati attesi:	Con il login iniziale si viene portati alla pagina gestione fatturazioni. Quando si cerca di accedere alle altre pagine si viene subito reindirizzati alla pagina gestione fatturazioni.		

Test Case:	TC-003	Nome:	Funzionamento logout
Riferimento:	REQ-01		
Descrizione:	Effettuare il login con un utente, dopo avere eseguito il logout provare ad accedere alle pagine.		
Prerequisiti:	Pagina di login, utente registrato nel database, tutte le pagine create		
Procedura:	<ol style="list-style-type: none">1. Aprire il browser e immettere: http://localhost/Gestione-fatturazioni/src/invoices/home/index2. Inserire le credenziali, username e password3. Premere il bottone “Login”4. Premere il bottone in alto a destra “Logout”5. Spostarsi sulla pagina: http://localhost/Gestione-fatturazioni/src/invoices/invoices/index6. Spostarsi sulla pagina: http://localhost/Gestione-fatturazioni/src/invoices/users/index7. Spostarsi sulla pagina: http://localhost/Gestione-fatturazioni/src/invoices/products/index8. Spostarsi sulla pagina: http://localhost/Gestione-fatturazioni/src/invoices/clients/index		
Risultati attesi:	Con il login iniziale si viene portati alla pagina gestione fatturazioni. Quando si cerca di accedere alle altre pagine si viene subito reindirizzati alla pagina di login.		

Test Case:	TC-004	Nome:	Controllo Registrazione
Riferimento:	REQ-01 REQ-02		
Descrizione:	Registrare un nuovo utente e accedere subito dopo.		
Prerequisiti:	Pagina di login, Pagina registrazione		
Procedura:	<ol style="list-style-type: none"> 1. Aprire il browser e immettere: http://localhost/Gestione-fatturazioni/src/invoices/registration/index 2. Registrare un nuovo utente 3. Accedere con l'utente appena creato 		
Risultati attesi:	<p>La registrazione viene effettuata con successo, dopo l'accesso si viene portati nella pagina :</p> <p>http://localhost/Gestione-fatturazioni/src/invoices/home/disableUser</p> <p>Dove viene mostrato un messaggio, specificando che l'utente non è abilitato.</p>		

Test Case:	TC-005	Nome:	Controllo Gestione Utenti
Riferimento:	REQ-03		
Descrizione:	Nella pagina gestione utenti, registrare un nuovo utente, abilitarlo e accedere subito dopo.		
Prerequisiti:	Pagina di login, Pagina gestione utenti		
Procedura:	<ol style="list-style-type: none"> 1. Fare il login con l'amministratore. 2. Andare al seguente indirizzo: http://localhost/Gestione-fatturazioni/src/invoices/users/index 3. Aggiungere un nuovo utente 4. Abilitare l'utente 5. Eseguire il logout 6. Accedere con l'utente creato 		
Risultati attesi:	L'accesso con il nuovo utente viene effettuata con successo, e si viene portati alla pagina gestione fatturazioni: http://localhost/Gestione-fatturazioni/src/invoices/invoices/index		

Test Case:	TC-006	Nome:	Controllo Gestione Prodotti
Riferimento:	REQ-04		
Descrizione:	Nella pagina gestione prodotti, registrare un nuovo prodotto e modificarlo.		
Prerequisiti:	Pagina di login, Pagina gestione prodotti		
Procedura:	<ol style="list-style-type: none"> 1. Fare il login con l'amministratore. 2. Andare al seguente indirizzo: http://localhost/Gestione-fatturazioni/src/invoices/products/index 3. Aggiungere un nuovo prodotto 4. Modificare il prodotto 5. Ricaricare la pagina 		
Risultati attesi:	<p>Il nuovo prodotto viene salvato nel database e viene subito mostrato nella lista. Quando viene modificato si vede subito la modifica. Quando si ricarica la pagina il prodotto compare ancora nella lista con gli stessi dati modificati.</p>		

Test Case:	TC-007	Nome:	Controllo gestione clienti
Riferimento:	REQ-05		
Descrizione:	Nella pagina gestione clienti, registrare un nuovo cliente e modificarlo.		
Prerequisiti:	Pagina di login, Pagina gestione clienti		
Procedura:	<ol style="list-style-type: none"> 1. Fare il login con l'amministratore. 2. Andare al seguente indirizzo: http://localhost/Gestione-fatturazioni/src/invoices/clients/index 3. Aggiungere un nuovo cliente 4. Modificare il cliente 5. Ricaricare la pagina 		
Risultati attesi:	<p>Il nuovo cliente viene salvato nel database e viene subito mostrato nella lista. Quando viene modificato si vede subito la modifica. Quando si ricarica la pagina il cliente compare ancora nella lista con gli stessi dati modificati.</p>		

Test Case:	TC-008	Nome:	Controllo lista fatture
Riferimento:	REQ-06		
Descrizione:	Nella pagina gestione clienti, registrare un nuovo cliente e modificarlo.		
Prerequisiti:	Pagina di login, Pagina gestione clienti		
Procedura:	<ol style="list-style-type: none">1. Fare il login con l'amministratore.2. Andare al seguente indirizzo: http://localhost/Gestione-fatturazioni/src/invoices/clients/index3. Aggiungere un nuovo cliente4. Modificare il cliente5. Ricaricare la pagina		
Risultati attesi:	Il nuovo cliente viene salvato nel database e viene subito mostrato nella lista. Quando viene modificato si vede subito la modifica. Quando si ricarica la pagina il cliente compare ancora nella lista con gli stessi dati modificati.		

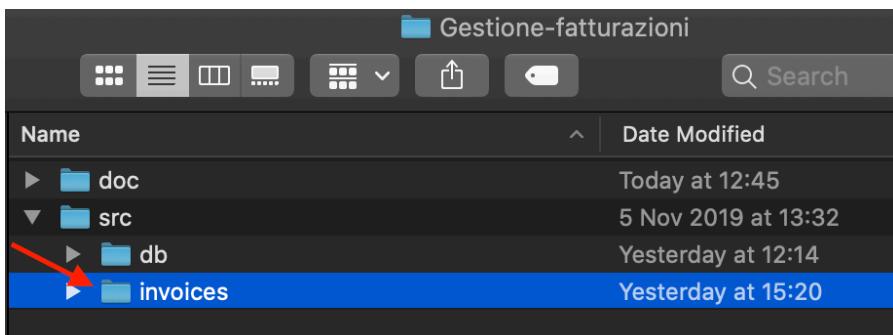
5.2 Risultati test

Test Case	Risultato	Riferimenti
TC-01	Superato	REQ-01
TC-02	Superato	REQ-01
TC-03	Superato	REQ-01
TC-04	Superato	REQ-02
TC-05	Superato	REQ-03
TC-06	Superato	REQ-04
TC-07	Superato	REQ-05
TC-08	Superato	REQ-06

5.3 Installazione Sito Web

Il mio progetto è stato sviluppato in locale sulla mia macchina, non l'ho trasferito su un server vero e proprio. Questa guida riguarda come installare il sito web dovunque esso sia.

Per prima cosa si deve mettere la cartella “invoices”, sul server o nella cartella del server locale.



Si deve modificare la prima costante (nel file **config.php**) in modo tale che l'URL contenga
<host:porta><Percorso della cartella invoices del sito>

Si deve modificare la seconda costante (nel file **config.php**) in modo tale che l'URL contenga
<Percorso della cartella app (all'interno di config.php) del sito>

```
/**  
 * Define the URL of the site  
 */  
define('URL', 'http://localhost:55777/Gestione-fatturazioni/src/invoices/');  
  
/**  
 * Define the ROOT of the site  
 */  
define('ROOT', '/Gestione-fatturazioni/src/invoices/app/');
```

5.4 Mancanze/limitazioni conosciute

Nelle pagine per la gestione degli utenti, prodotti e clienti durante modificare i dati che vengono modificati sono ottenuti tramite JS, e non vengono validati come durante l'aggiunta di un nuovo record, in cui vengono validati i dati prima di essere salvati.

Per ovviare a questo problema si potrebbe utilizzare la stessa interfaccia per l'aggiunta, che è già predisposta per effettuare la validazione dati.

Quando un utente viene creato nella pagina gestione utenti viene impostata anche la sua password. Ma dopo se si vuole modificarla non è possibile farlo.

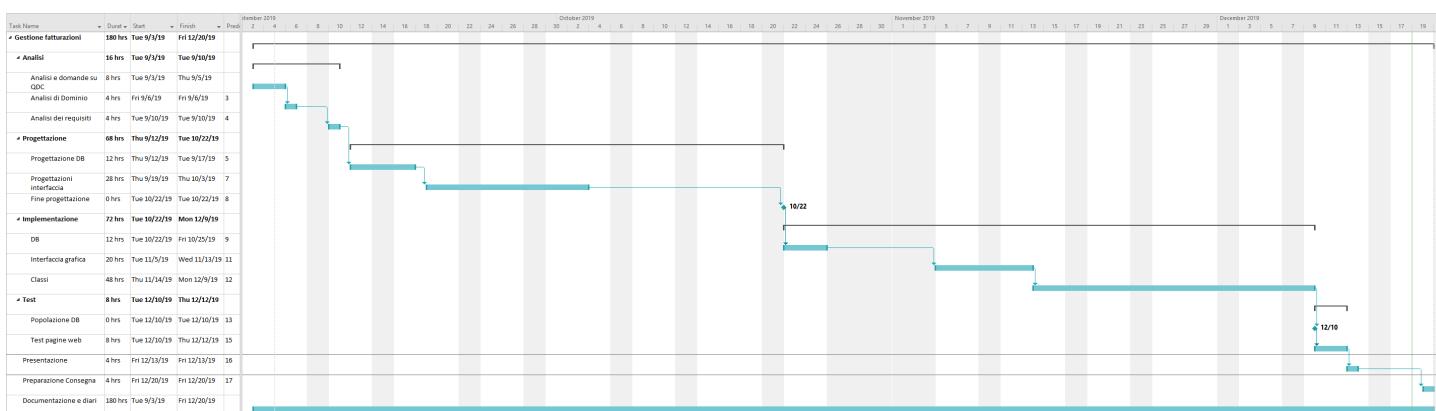
Non si possono aggiungere o modificare le fatture, le fatture non si possono copiare e non possono essere stampate.

6 Consuntivo

La pianificazione iniziale del progetto si è realizzata solo in parte. La parte iniziale della pianificazione è stata svolta come pianificato fino a finire la progettazione come previsto. Dopo c'è stata l'implementazione che è durata di più del previsto, durando qualche settimana in più. Questo ritardo a lasciato meno tempo per svolgere i test.

Durante il progetto ho dovuto implementare funzionalità non previste durante l'analisi dei requisiti, e quindi o dovuto riservare del tempo, il quale non era previsto nella pianificazione iniziale dell'implementazione. Questo ha comportato anche l'aggiunta di prerequisiti nei requisiti già esistenti.

Il Gantt consuntivo è presente anche come allegato in un formato più leggibile.



7 Conclusioni

Il prodotto finale ottenuto è un sito web MVC strutturato da diverse pagine web. L’interfaccia delle pagine web è minimalista e molto semplice, ottenendo uno stile ricercato e che aiuta a chi utilizza il sito di orientarsi meglio e senza confondersi. L’accesso alle pagine web è stato messo in sicurezza in modo tale che solo chi ha i permessi possa accederci, questo garantisce che i dati restino al sicuro e non possono essere modificati da chi non ne ha il permesso. Questo sito permette di gestire i dati delle fatture di piccole medie aziende, permettendo anche che tutti i dati possono essere visualizzati, modificati e eliminati in qualsiasi momento.

7.1 Sviluppi futuri

Un utente può registrarsi ma dopo non può in nessun modo modificare la propria password. Sarebbe una funzionalità ben voluta da un utente che ha dimenticato la password.

Durante la registrazione degli utenti vengo salvati i loro dati sul database prima che vengono accettati, quindi se alcuni utenti non verranno accettati i loro dati rimarranno nel database, quindi si potrebbe sviluppare un modo per eliminare quei dati in automatico. Eliminando questi dati non necessari, si potrebbe migliorare le performance del sito web o più precisamente del database, liberando anche dello spazio occupato inutilmente.

Per quanto riguarda la validazione viene effettuata su tutti i dati, ma non viene fatta in modo specifico su ogni diverso tipo di dato. Si potrebbe utilizzare un metodo più completo di validazione che verifichi tutti i criteri che quel tipo di dato deve rispettare. Magari durante la validazione si potrebbe impostare un messaggio di errore specifico, che esplicita il problema ritrovato nel dato immesso.

7.2 Considerazioni personali

Secondo me questo progetto era proporzionato al tempo a disposizione, anche se io non ho finito al completo il progetto. Non sono riuscito a finire l’implementazione perché non sono andato spedito con l’implementazione e anche perché le prime soluzioni adottate durante l’implementazione non erano spesso quelle finali, quindi secondo me non solo devo migliorare nell’implementazione ma che nella progettazione.

Questo progetto mia ha permesso di imparare a utilizzare il linguaggio php in modo diverso e più professionale. Ho imparato soprattutto come rendere lo sviluppo web più preciso e chiaro, in modo da implementare tutto il sistema con una precisa logica e struttura. Quello che ho imparato mi servirà non solo per sviluppare progetti simili, ma anche più complessi.

8 Sitografia

- <https://www.php.net>, *PHP Official Guide*, 18/10/2019
- <https://www.w3schools.com>, *W3schools*, 22/10/2019
- <https://getbootstrap.com>, *Bootstrap component library*, 22/10/2019
- <https://jquery.com>, *JQuery JavaScript library*, 22/10/2019
- <https://craftpip.github.io/jquery-confirm/>, *JQuery Confirm - JQuery Plugin*, 28/11/2019
- <https://notifyjs.jpillora.com>, *Notify.js JQuery Plugin*, 28/11/2019
- <http://designmodo.github.io/Flat-UI/>, *Flat UI User Interface Kit*, 29/11/2019

9 Glossario

Parola	Significato
DB	Database, bancadati in italiano
JS	JavaScript linguaggio script per siti web
checkbox	Input pagina web, spunta
MVC	Model View Controller, struttura sito web
JSON	Tipo di file, oggetti JS
POST	Metodo di invio die dati, nascosti
AJAX	Metodo per inviare dati, da js a php e vice versa

10 Allegati

- QDC
- Gantt Preventivo
- Gantt Consuntivo
- Diari di lavoro

INFORMAZIONI GENERALI

Allievo	Nome:	Cognome:
Luogo di lavoro	Aula 417 – Scuola d'Arti e Mestieri Trevano	
Orientamento	<input checked="" type="checkbox"/> 88602 Informatica aziendale	
Docente responsabile	Nome:Massimo	Cognome: Sartori
Periodo	martedì 3 settembre 2019 - venerdì 20 dicembre 2019	
Orario di lavoro	Secondo orario scolastico primo semestre	
Numero di ore lezione	174	
Pianificazione (in H o %)	Analisi: 10%	
	Implementazione: 50%	
	Test: 10%	
	Documentazione: 30%	

PROCEDURA

- L'allievo realizza il lavoro autonomamente sulla base del quaderno dei compiti ricevuto il 1 ° giorno.
- Il quaderno dei compiti è presentato, commentato e discusso con l'allievo. Con la sua firma, l'allievo accetta il lavoro proposto.
- L'allievo ha conoscenza della scheda di valutazione all'inizio del lavoro.
- L'allievo è responsabile dei suoi dati.
- In caso di problemi gravi, l'allievo avverte immediatamente il docente responsabile.
- L'allievo ha la possibilità di chiedere aiuto, ma deve menzionarlo nella documentazione.
- Alla fine del tempo a disposizione per la realizzazione del progetto, l'allievo deve inviare via e-mail il progetto al docente responsabile. In parallelo, una copia cartacea della documentazione dovrà essere fornita sempre al docente responsabile. Quest'ultima deve essere in tutto identica alla versione elettronica.

TITOLO

Gestione fatturazioni

HARDWARE E SOFTWARE DISPONIBILE

1 PC

PREREQUISITI

-

DESCRIZIONE DEL PROGETTO

L'obiettivo del progetto è quello di creare un sito nel quale si possano gestire tutte le fatturazioni di una piccola azienda.

In questo progetto ci saranno 2 utenti principali:

Amministratore

Dovrà dapprima poter gestire i clienti, con tutti i dati necessari (i clienti potranno essere sia persone fisiche che giuridiche).

I clienti potranno essere associati ad una determinata tipologia di fattura (ad esempio vendita legname oppure utilizzo strada).

Bisognerà poter associare i clienti alla o alle tipologie di fatture e creare le fatture vere e proprie. Queste fatture dovranno venir salvate i database e stampate (formato pdf).

Il numero di stampe di ogni fattura dovrà essere salvato e mostrato all'utente (per potersi ricordare se una determinata fattura è già stata stampata/inviata al cliente).

Bisognerà poter eseguire delle ricerche per poter trovare le fatture e/o richiami per data, per cliente e per importo minimo/massimo.

Visto che ogni anno le fatture saranno quasi sempre le stesse per ogni cliente, bisognerà disporre di un sistema che in automatico prenda la fattura dell'anno precedente per i clienti e la copi per l'anno in corso. Questo dovrà essere possibile farlo anche nel caso un cliente abbia più fatture.

Dovrà essere possibile segnare se una fattura è stata pagata e la data del pagamento. Se vi fossero ritardi dovranno apparire dei messaggi che indicheranno di inviare un richiamo. Anche il richiamo dovrà essere salvato e tracciato sul sito.

Utente

L'utente potrà solamente cercare le fatture e visualizzarle ma non potrà eseguire nessun'altra operazione. Se visualizzasse la fattura il numero di stampe non incrementerebbe.

RISULTATI FINALI

L'allievo è responsabile della consegna al docente responsabile di:

- Una pianificazione iniziale (entro la prima settimana)

- Una documentazione del progetto
- Un diario di lavoro
- Un prodotto finale

PUNTI TECNICI SPECIFICI VALUTATI

La griglia di valutazione definisce i criteri generali secondo cui il lavoro dell'allievo sarà valutato (documentazione, diario, rispetto dei standard, qualità, ...).

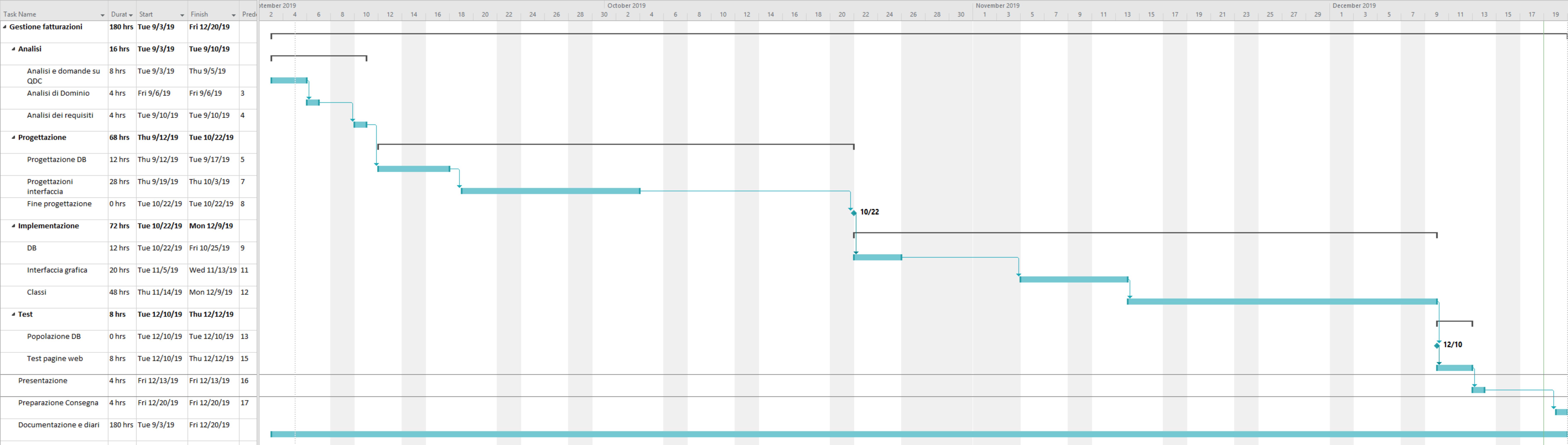
Inoltre, il lavoro sarà valutato sui seguenti 7 punti specifici (punti da A14 a A20):

1. 159 (*Analisi del problema (programmazione)*)
2. 162 (*Progettazione - Architettura del programma*)
3. 165 (*Implementazione della soluzione (programmazione)*)
4. 121 (*Ergonomia del programma*)
5. 166 (*Stile di codifica; Leggibilità del codice*)
6. 164 (*Codifica: Trattamento degli errori*)
7. 128 (*Identificazione delle entità necessarie conformemente al problema dato*)
8. 135 (*Documentazione DB, tabelle, ecc.*)

FIRMA

Allievo
(luogo e data)

Docente responsabile
(luogo e data)



Diario di lavoro

Luogo	Canobbio, SAM Trevano
Data	03/09/2019

Lavori svolti

Ho ricevuto il QDC del progetto e ho elaborato delle domande sulle cose dove non ero in chiaro, che ho mandato tramite email al docente responsabile.

Problemi riscontrati e soluzioni adottate

Punto della situazione rispetto alla pianificazione

Programma di massima per la prossima giornata di lavoro

Diario di lavoro

Luogo	Canobbio, SAM Trevano
Data	05/09/2019

Lavori svolti

Ho guardato insieme al mio formatore il QDC del progetto, e mi hanno spiegato più precisamente cosa andrò a fare. Durante la spigazione del formatore mi sono soffermato su alcune cose su cui non ero del tutto in chiaro.

Chiarimenti:

- Nel sito deve esserci solo un amministratore e deve essere creato manualmente.
- Nel sito i nuovi utenti si devono registrare, e solo dopo che l'amministratore li accetta potranno visualizzare le fatture.
- È l'amministratore che si occupa di registrare tutti i dati dei clienti e delle fatture sul sito.
- I clienti vengono salvati a parte, e solo dopo si potrà associarli ad una fattura.
- Deve essere presente la funzione copia, che permette di copiare una fattura.
- Di default dopo 30 giorni si deve notificare il ritardo di una fattura e deve esserci la possibilità di inviare il riporto tramite una funzione, la scadenza di una fattura si può modificare anche manualmente.

Ha iniziato a fare la documentazione, iniziando dall'analisi, e ho fatto il gantt.

Problemi riscontrati e soluzioni adottate

-

Punto della situazione rispetto alla pianificazione

-

Programma di massima per la prossima giornata di lavoro

-

Diario di lavoro

Luogo	Canobbio, SAM Trevano
Data	06/09/2019

Lavori svolti

Ho sistemato il gantt e l'ho inserito nella documentazione.

Sono andato avanti con la documentazione, ho finito l'analisi di dominio e ho iniziato l'analisi dei requisiti.

Ho inviato altre domande al mio formatore, per quanto riguarda i requisiti del progetto.

Problemi riscontrati e soluzioni adottate

-

Punto della situazione rispetto alla pianificazione

-

Programma di massima per la prossima giornata di lavoro

-

Diario di lavoro

Luogo	Canobbio, SAM Trevano
Data	10/09/2019

Lavori svolti

Ho ricevuto dei chiarimenti su dei requisiti di cui non ero completamente in chiaro, che sono:

- Una fattura potrà avere se necessario un numero identificativo, ma non è richiesto dal committente.
- Il cliente viene associato solo a alle sue fatture, ma ogni fattura viene associata ad una singola tipologia
- In una fattura deve esserci la possibilità di aggiungere il logo dell'azienda mandante.

Ho ricercato su internet cosa si intende per fattura, per capire meglio di cosa si occupa il mio lavoro.

La fattura è un documento molto importante per un'azienda, e assicura da un lato, come prova di fornitura di un servizio o prodotto, dall' altro assicura l'erogazione di un pagamento.

Scopi di una fattura:

1. Farti pagare per il proprio lavoro
2. Conoscere nei minimi dettagli le tue entrate e le tue uscite
3. Offre trasparenza sui prodotti e servizi a pagamento
4. È uno strumento di branding per la tua azienda

Contenuto di una fattura:

1. Intestazione della fattura
2. Linee di fatturazione
3. Riepilogo della fattura

L'intestazione contiene elementi necessari come il tuo indirizzo. Le righe della fattura contengono i beni e/o servizi da te forniti, compreso il loro prezzo.

Il riepilogo della fattura contempla:

- L'importo netto (IVA esclusa)
- L'aliquota e l'importo dell'IVA
- L'importo totale
- Le condizioni di pagamento
- Le opzioni di pagamento

Elementi obbligatori di una fattura:

Per essere giuridicamente valida, ogni fattura deve contenere una serie di elementi:

- La parola **fattura** deve essere menzionata in alto in modo chiaro.
- La **data** e un **numero di serie**: utilizza un numero diverso per ogni fattura.
- L'**indirizzo** completo di entrambe le parti
- I dati di pagamento: **numero di conto corrente bancario** o indirizzo PayPal
- Il tuo numero di partita **IVA** (o equivalente)
- La **data di consegna**
- Una descrizione della **natura** e della **quantità** di prodotti e/o servizi in modo che i tuoi clienti sappiano esattamente per cosa pagano:
- I diversi componenti di un progetto o prodotto e il **prezzo unitario** per ogni componente
- La tua **tariffa oraria** e il tempo dedicato a ciascuna fase o parte di un progetto
- L'**aliquota e l'importo dell'IVA**
- L'importo dovuto al **netto dell'IVA**

Preso da: <https://www.teamleadercrm.it/fattura>

Ho concluso l'analisi e la specifica dei requisiti e ho descritto dettagliatamente il Gantt della pianificazione.

Problemi riscontrati e soluzioni adottate

-

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione

Programma di massima per la prossima giornata di lavoro

-

Diario di lavoro

Luogo	Canobbio, SAM Trevano
Data	12/09/2019

Lavori svolti

Ho ricevuto dei chiarimenti su dei requisiti di cui non ero completamente in chiaro, che sono:

- Nelle fatture per migliorare l'esperienza d'uso si potrebbe far vedere i relativi richiami, quando si apre una fattura.
- Se una fattura è già stata stampata e si cerca di ristamparla, si potrebbe aggiungere un avviso che richiama l'attenzione all'utente, che quella fattura è già stata stampata.
- Un **Utente** può visualizzare sia le fatture che i richiami.

Ho visto alcuni esempi di **Use case** per capire bene come farne uno, visto che non abbiamo mai provato a farne, e infine l'ho fatto per il mio progetto.

Ho sistemato l'analisi dei requisiti, in modo che compaiano solo i requisiti che vuole il committente.

Problemi riscontrati e soluzioni adottate

-

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione

Programma di massima per la prossima giornata di lavoro

-

Diario di lavoro

Luogo	Canobbio, SAM Trevano
Data	13/09/2019

Lavori svolti

Ho costruito lo schema che descrive l'architettura del sistema, e un altro schema logico che descrive la fase di login nel sito, da parte di diversi utenti.

Ho iniziato a progettare il db, ho messo giu su carta uno schizzo e l'ho perfezionato fino ad arrivare ad uno schema completo.

Ho visto che su internet per salvare un immagine in un database non conviene, ma invece si dovrebbe salvare la **Path** dell'immagine nel database, invece l'immagine verrà salvata sul server che ospita il sito(in un apposita cartella).

Fonte: <https://stackoverflow.com/questions/6472233/can-i-store-images-in-mysql>

Ho descritto i vari dati che ci sono all'interno di una fattura.

Problemi riscontrati e soluzioni adottate

-

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione

Programma di massima per la prossima giornata di lavoro

-

Diario di lavoro

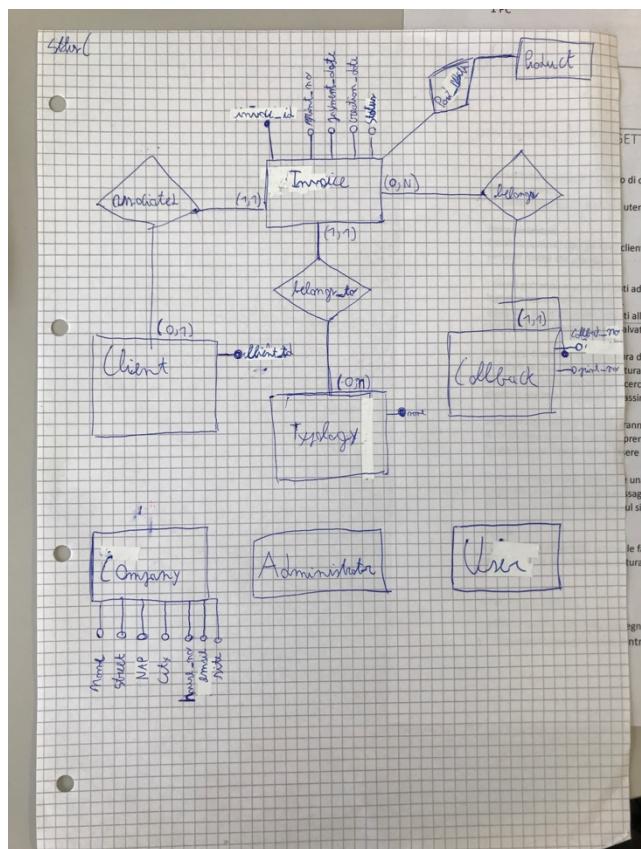
Luogo	Canobbio, SAM Trevano
Data	17/09/2019

Lavori svolti

Chiarimenti avvenuti tramme email:

- Per ogni fattura bisogna memorizzare la data di emissione, ma non quella di consegna perché non possiamo saperlo.
- Nelle fatture bisogna poter salvare diversi prodotti o servizi, con il loro corrispettivo prezzo.
- I dati dell'azienda, potrebbero essere salvati in una tabella nel database senza nessuna relazione con le altre.
- Per salvare o gestire le informazioni dell'azienda, si deve creare un'altra pagina in cui si possa fare questo.
- Bisogna allegare tutto quello che si fa nel diario; schemi, foto di cose fatte a mano, analisi che hai fatto ecc...; Bisogna poter trovare tutto quello si è fatto.

Ho fatto uno lo schema del database su carta per farmi un'idea di come è fatto:



Ho messo giù tutti i dati che contengono le tabelle del database:

La tabella Client contiene tutte le informazioni di un cliente, che dopo potrà essere associato a una fattura.

Client		
Nome del Dato	Tipo	Descrizione
client_id	(PK) int	Identificativo
name	varchar(50)	Nome
surname	varchar(50)	Cognome
street	varchar(100)	Via del indirizzo di fatturazione
house_no	varchar(6)	Numero civico del indirizzo di fatturazione
city	(FK) varchar(50)	Città del indirizzo di fatturazione

La tabella Typology contiene le varie tipologie di fatture, che vengono consegnate ai clienti.

Typology		
Nome del Dato	Tipo	Descrizione
name	(PK) varchar(100)	Il nome della tipologia

La tabella Product contiene tutti le informazioni dei prodotti contenuti in una fattura/richiamo.

Product		
Nome del Dato	Tipo	Descrizione
product_id	(PK) int	Identificativo
name	varchar(100)	Nome
price	decimal(19,4)	Prezzo
sell_date	date	Data di quando è stata effettuata la vendita

La tabella Invoice contiene tutte le informazioni di una fattura/richiamo.

Invoice		
Nome del Dato	Tipo	Descrizione
invoice_id	(PK) int	Identificativo
print_no	int	Numero di stampe
payment_date	date	Data di pagamento
creation_date	date	Data di creazione
status	varchar(8)	Specifica lo stato della fattura e può avere diversi valori, che sono: saved; printed; paid; expired; callback
logo_path	varchar(100)	Il percorso sul server per poter ottenere il logo dell'azienda
iban	varchar(30)	Il numero del conto dell'azienda
client	int (FK)	Cliente per il quale è stata emessa la fattura

La tabella Company contiene tutte le informazioni dell'azienda.

Company		
Nome del Dato	Tipo	Descrizione
name	(PK) varchar(50)	Nome
street	varchar(100)	Via di fatturazione dell'azienda
house_no	varchar(6)	Numero civico di fatturazione dell'azienda
email	varchar(100)	Email
site	varchar(100)	Sito web
city	(FK) varchar(50)	Città di fatturazione dell'azienda

La tabella Administrator contiene tutte le informazioni dell'amministratore del sito web.

Administrator		
Nome del Dato	Tipo	Descrizione
username	(PK) varchar(50)	Nome utente
password	binary(32)	Password

email	varchar(100)	Email
-------	--------------	-------

La tabella User contiene tutte le informazioni di un utente del sito web.

User		
Nome del Dato	Tipo	Descrizione
username	(PK) varchar(50)	Nome utente
password	binary(32)	Password
email	varchar(100)	Email
registered	tinyint	Specifica se l'utente è registrato, 1 è registrato 0 non lo è

La tabella City contiene, le informazioni di una città.

City		
Nome del Dato	Tipo	Descrizione
name	(PK) varchar(50)	Nome della città
nap	int(5)	Numero di avviamento postale

Problemi riscontrati e soluzioni adottate

-

Punto della situazione rispetto alla pianificazione

Un po' in ritardo con la pianificazione, bisogna fare lo schema del Database.

Programma di massima per la prossima giornata di lavoro

-

Diario di lavoro

Luogo	Canobbio, SAM Trevano
Data	19/09/2019

Lavori svolti		
Chiarimenti:		
<ul style="list-style-type: none">• Gli Utenti e l'Amministratore possono avere un email.• I clienti non devo essere associati ad una tipologia di fattura.		
Ho sistemato la struttura logica del Database:		
La tabella Client contiene tutte le informazioni di un cliente, che dopo potrà essere associato a una fattura.		
Client		
Nome del Dato	Tipo	Descrizione
id	(PK) int	Identificativo
name	varchar(50)	Nome
surname	varchar(50)	Cognome
street	varchar(100)	Via del indirizzo di fatturazione
house_no	varchar(6)	Numero civico del indirizzo di fatturazione
city_id	(FK) varchar(50)	Città del indirizzo di fatturazione
Typology		
Nome del Dato	Tipo	Descrizione
id	(PK) int	Identificativo
name	varchar(100)	Il nome della tipologia

La tabella Product contiene tutte le informazioni dei prodotti contenuti in una fattura/richiamo.

Product		
Nome del Dato	Tipo	Descrizione
id	(PK) int	Identificativo
name	varchar(100)	Nome
price	decimal(19,4)	Prezzo

La tabella Invoice contiene tutte le informazioni di una fattura/richiamo.

Invoice		
Nome del Dato	Tipo	Descrizione
id	(PK) int	Identificativo
print_no	Int(3)	Numero di stampe
payment_date	date	Data di pagamento
creation_date	date	Data di creazione
status	varchar(7)	Specifica lo stato della fattura e può avere diversi valori, che sono: saved; printed; paid; expired
callback	tinyint	Specifica se si tratta di una fattura o di un richiamo
client_id	(FK) int	Cliente per il quale è stata emessa la fattura/richiamo
typology_id	(FK) varchar(100)	La tipologia alla quale è associata la fattura/richiamo

La tabella Inserted contiene le informazioni che ci sono nel associazione di un prodotto con una fattura.

Inserted		
Nome del Dato	Tipo	Descrizione
product_id	(PK) int	Identificativo del prodotto associato
invoice_id	(PK) int	Identificativo della fattura/richiamo associato
sell_date	date	Data di quando è stata effettuata la vendita di un prodotto/servizio

La tabella Company contiene tutte le informazioni dell'azienda.

Company		
Nome del Dato	Tipo	Descrizione
id	(PK) int	Identificativo
name	varchar(50)	Nome
logo_path	varchar(100)	Il percorso sul server per poter ottenere il logo dell'azienda
iban	varchar(30)	Il numero del conto dell'azienda
street	varchar(100)	Via di fatturazione dell'azienda
house_no	varchar(6)	Numero civico di fatturazione dell'azienda
email	varchar(100)	Email
site	varchar(100)	Sito web
city_id	(FK) varchar(50)	Città di fatturazione dell'azienda

La tabella User contiene tutte le informazioni di un utente del sito web.

User		
Nome del Dato	Tipo	Descrizione
id	(PK) int	Identificativo
username	varchar(50)	Nome utente
password	binary(32)	Password
email	varchar(100)	Email
registered	tinyint	Specifica se l'utente è registrato, 1 è registrato 0 non lo è

La tabella Administrator contiene tutte le informazioni dell'amministratore del sito web.

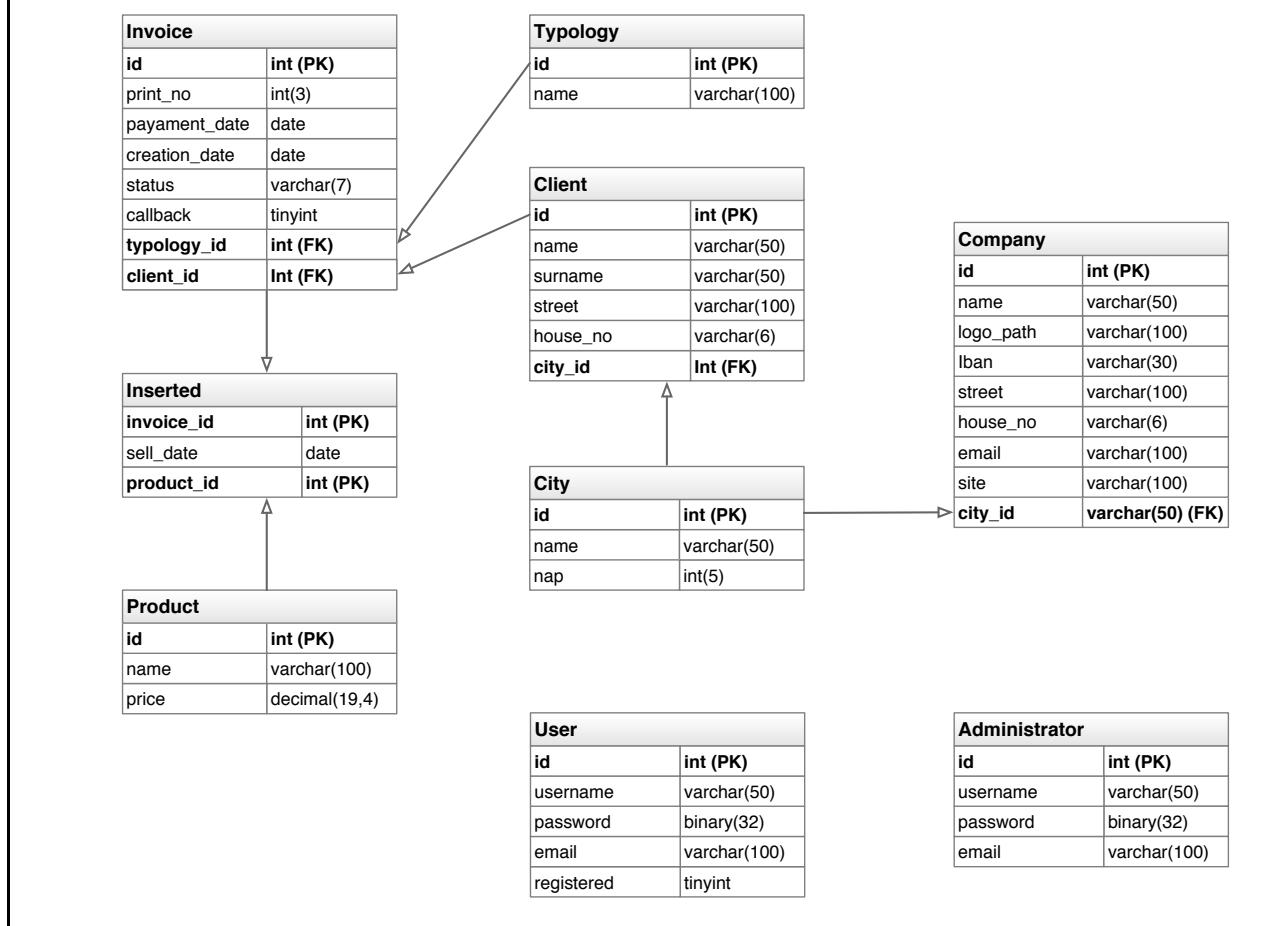
Administrator		
Nome del Dato	Tipo	Descrizione
id	(PK) int	Identificativo

username	varchar(50)	Nome utente
password	binary(32)	Password
email	varchar(100)	Email

La tabella City contiene le informazioni di una città.

City		
Nome del Dato	Tipo	Descrizione
id	(PK) int	Identificativo
name	varchar(50)	Nome della città
nap	int(5)	Numero di avviamento postale

Ho fatto il grafico che illustra la struttura dei dati del database:



Problemi riscontrati e soluzioni adottate

-

Punto della situazione rispetto alla pianificazione

Un po' in ritardo con la pianificazione, bisogna fare lo schema ER del Database.

Programma di massima per la prossima giornata di lavoro

-

Diario di lavoro

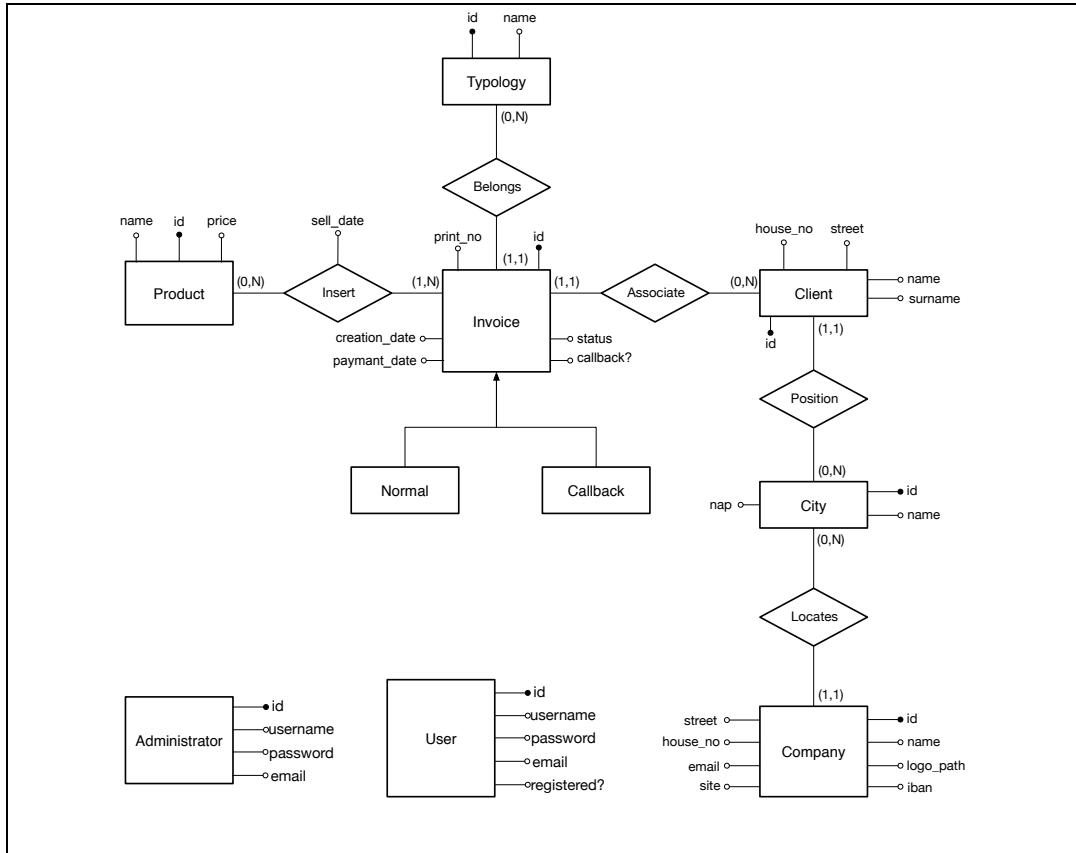
Luogo	Canobbio, SAM Trevano
Data	20/09/2019

Lavori svolti

Ho fatto il diagramma ER del database:

Questo è il diagramma ER che descrive il database, e fa vedere le varie tabelle come sono relazionate fra loro, ma anche le varie colonne che possidono al loro interno.

Da questo schema si può comprendere come verranno inseriti i dati e a cosa serviranno.



Ho fatto il mockup della pagina iniziale di login.

Questo è il mockup dell'interfaccia che avrà la pagina di login, che sarà la prima pagina visualizzata quando si entrerà nel sito.

Quando l'utente si trova a questa schermata, può loggarsi facilmente inserendo il suo username e password e infine schiacciando sul bottone **Login**.

The mockup shows a large, bold 'Login' title at the top. Below it is a 'Username*' field containing 'Bob...' with a red 'Error' message above it. A 'Password*' field is below it. At the bottom is a red 'Login' button. To the right of the input fields is a 'Sing up now' link.

Problemi riscontrati e soluzioni adottate

-

Punto della situazione rispetto alla pianificazione

Un po' in ritardo con la pianificazione, bisogna fare lo schema ER del Database.

Programma di massima per la prossima giornata di lavoro

-

Diario di lavoro

Luogo	Canobbio, SAM Trevano
Data	24/09/2019

Lavori svolti

Ho tradotto in italiano il design della pagina di login e ho ampliato la sua descrizione:

Quando si entra nel sito la prima schermata che comparirà sarà questa, dove l'utente che entrerà potrà accedere con il proprio account, e se non possiede un account potrà registrarsi andando alla pagina di registrazione, schiacciando il link in blu “Registrati subito”.

Per accedere l'utente dovrà inserire il proprio Username e Password, e infine schiacciare il bottone Accedi. Se dopo aver schiacciato il bottone Accedi il sito trova che i dati inseriti non corrispondono o che non sono corretti, apparirà un errore sopra la campo con l'informazione sbagliata.

The screenshot shows a login form with the following elements:

- Accedi**: Large title at the top center.
- Username***: Label followed by an input field containing "Bob...". Above the input field, the word "Errore" is displayed in red.
- Password***: Label followed by an empty input field.
- Registrati subito**: Link in blue text below the password field.
- Accedi**: Large red button at the bottom center.

Caption: Figura 1 Progettazione dell'interfaccia di login

Ho fatto la progettazione di sarà una fattura, cioè come saranno disposte tutte le informazioni:

Per ogni fattura deve esserci il titolo “Fattura” e se si tratta di un richiamo il titolo “Richiamo N”, dove N è un numero variabile che identifica se si tratta del primo richiamo, secondo, ecc. Subito sotto al titolo è presente il numero della fattura e la data di quando è stata creata la fattura.

Dopo sono presenti le informazioni dell’azienda con in parte il logo corrispondente. Sotto sono presenti le informazione del cliente cioè il destinatario della fattura, e le informazioni dei prodotti/servizi offerti dall’azienda.

<h1>Fattura</h1>											
N. 32 Del 05/09/2019											
Nome Azienda											
Indirizzo											
Email											
Telefono											
Site											
IBAN											
Destinatario											
Codice: N											
Nome											
Indirizzo											
<table border="1"><thead><tr><th>Data</th><th>Codice</th><th>Prodotto</th><th>Prezzo</th></tr></thead><tbody><tr><td>05/09/2019</td><td>22</td><td>Legname</td><td>100 CHF</td></tr></tbody></table>				Data	Codice	Prodotto	Prezzo	05/09/2019	22	Legname	100 CHF
Data	Codice	Prodotto	Prezzo								
05/09/2019	22	Legname	100 CHF								
Total 100CHF											

Figura 2: Progettazione Fattura

Problemi riscontrati e soluzioni adottate

-

Punto della situazione rispetto alla pianificazione

Un po' in ritardo con la pianificazione.

Programma di massima per la prossima giornata di lavoro

-

Diario di lavoro

Luogo	Canobbio, SAM Trevano
Data	24/09/2019

Lavori svolti

Siccome la progettazione della struttura di una fattura non era molto compatta, ho cercato su internet un esempio da cui possa prendere spunto.

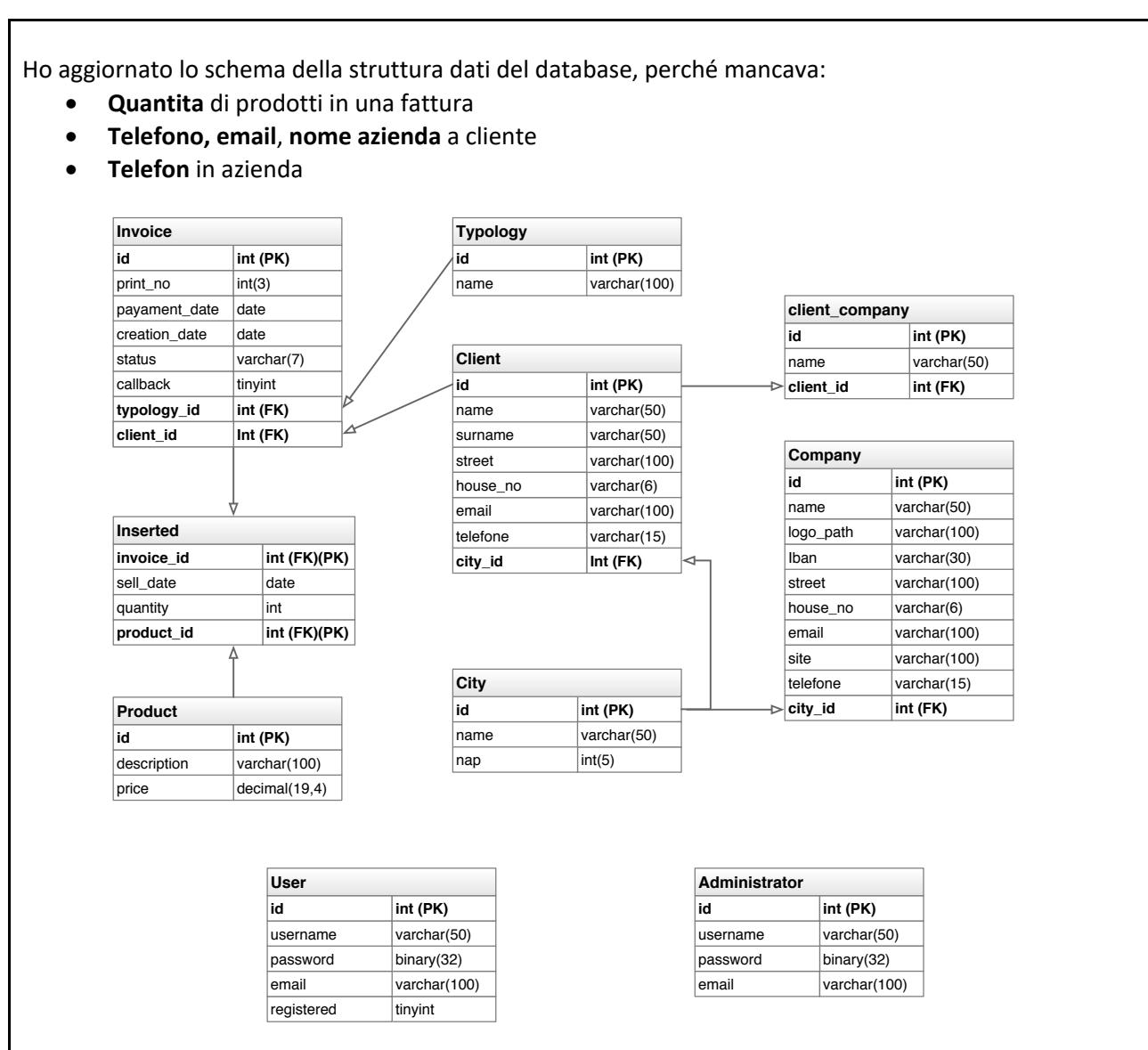
Il modello di fattura che mi è piaciuto di più:

The image shows a template for an Italian invoice (Fattura). The layout is as follows:

- Header:** Fattura (Invoice) and a placeholder for LOGO.
- Company Information:** Fields for Nome Azienda, Indirizzo, Tel, Email, Web.
- Client Information:** Fields for Nome del contatto, Ragione Sociale, Indirizzo, CAP, Tel, Email.
- Delivery Address:** Fields for Nome, Dipartimento, Indirizzo, Tel.
- Payment Terms:** Placeholder text: <Modalità di pagamento; Ad esempio: pagamento in 15 giorni>.
- Table:** A table for itemized charges with columns: Descrizione (Description), Q.tà (Quantity), Prezzo uni. (Unit Price), and Importo (Amount). All rows show 0.00.
- Annotations:** A placeholder for annotations: <Immettere eventuali annotazioni qui>.
- Subtotal and Taxes:** A table showing partial totals, discounts, VAT amounts, and shipping costs, all listed at 0.00.
- Total:** The final total is shown as **TOTALE FATTURA 0.00€**.

Basandomi sul modello trovato ho rieseguito la progettazione della fattura:

Fattura		Logo																																	
<Nome azienda> <Indirizzo> <Telefono> <Email, Web>		Data																																	
		Fattura no.																																	
Cliente																																			
<Nome del responsabile/ Nome cliente> <Nome azienda> <Indirizzo, NAP> <Telefono> <Email>																																			
<table border="1"><thead><tr><th>Descrizione</th><th>Q.tà</th><th>Prezzo uni.</th><th>Importo</th></tr></thead><tbody><tr><td></td><td></td><td></td><td>0,00CHF</td></tr><tr><td></td><td></td><td></td><td>0,00CHF</td></tr><tr><td></td><td></td><td></td><td>0,00CHF</td></tr><tr><td></td><td></td><td></td><td>0,00CHF</td></tr><tr><td></td><td></td><td></td><td>0,00CHF</td></tr><tr><td></td><td></td><td></td><td>0,00CHF</td></tr><tr><td colspan="3">TOTALE</td><td>0,00CHF</td></tr></tbody></table>				Descrizione	Q.tà	Prezzo uni.	Importo				0,00CHF	TOTALE			0,00CHF																				
Descrizione	Q.tà	Prezzo uni.	Importo																																
			0,00CHF																																
			0,00CHF																																
			0,00CHF																																
			0,00CHF																																
			0,00CHF																																
			0,00CHF																																
TOTALE			0,00CHF																																



Problemi riscontrati e soluzioni adottate

-

Punto della situazione rispetto alla pianificazione

Un po' in ritardo con la pianificazione.

Programma di massima per la prossima giornata di lavoro

-

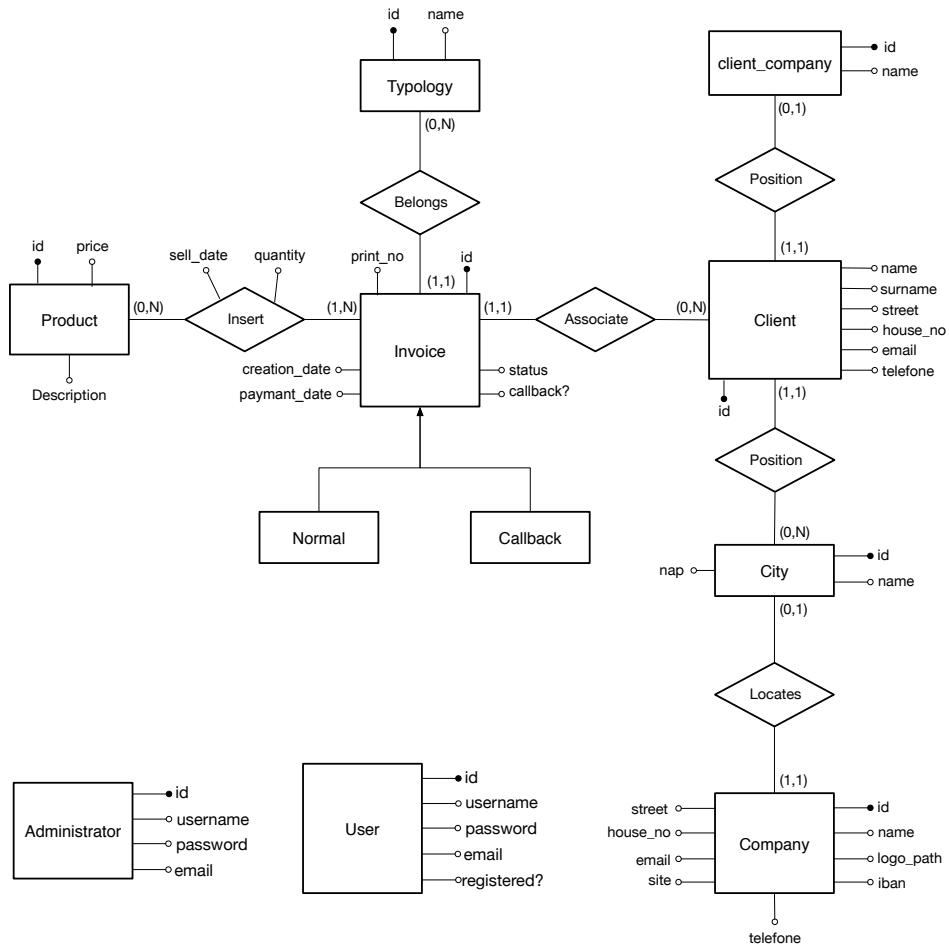
Diario di lavoro

Luogo	Canobbio, SAM Trevano
Data	24/09/2019

Lavori svolti

Ho sistemato il diagramma ER del database visto che la struttura è stata modificata lo scorso giorno di lavoro.

Diagramma ER:



Ho fatto il design della pagina di registrazione:

Quando un utente si vuole registrare andrà su questa pagina, all'interno sono presenti i campi di registrazione, e se si vuole andare alla pagina di **Accesso** basta schiacciare il link "Ho già un account". Una volta inserito tutti i dati correttamente, schiacciando il pulsante registrati l'utente sarà registrato, ma non potrà ancora accedere, e dovrà aspettare che l'**Administrator** gli dia il consenso.

Registrazione

Error

Username

Password

Conferma

Ho già un account

Registrati

Alle 15.00 sono andato nel aula 420 perché quelle di **Lingue e stage all estero** ci dovevano parlare del soggiorno che avevamo svolto all estero questa estate, sono tornato in classe alle 15:45.

Problemi riscontrati e soluzioni adottate

-

Punto della situazione rispetto alla pianificazione

Un po' in ritardo con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Creare l'interfaccia di gestione fatture, lista fatture, gestione clienti, gestione prodotti

Diario di lavoro

Luogo	Canobbio, SAM Trevano
Data	01/10/2019

Lavori svolti

Ho specificato meglio cosa succede all'interno della pagina per la creazione di una fattura, e anche come è strutturata:

Questa interfaccia compare quando l'amministratore vuole creare una nuova fattura/richiamo, quando si tratta di una **fattura** il titolo sarà “Fattura” se no “Richiamo N” se si tratta di un **richiamo**.

In questa pagina le informazioni dell'azienda, la data e il numero della fattura saranno aggiunti in automatico, e le informazioni dell'azienda potranno essere anche modificati.

Tutte le altre informazioni saranno aggiunte manualmente da zero, a meno che si copia una fattura già esistente, in quel caso le informazioni saranno già tutte presenti e saranno solo da modificare se necessario.

Ho trovato una mancanza nel sito web che potrebbe compromettere la performance del sito o più precisamente del database, occupando anche spazio inutilmente.

Durante la registrazione degli utenti vengo salvati i loro dati sul database prima che vengono accettati, quindi se alcuni utenti non verranno accettati i loro dati rimarranno nel database, quindi si potrebbe sviluppare un modo per eliminare quei dati in automatico.

Ho ideato come sarà il design della lista delle fatture sul sito, e i filtri che serviranno per filtrare le fattura elencate.

Questa interfaccia rappresenta come la lista di fatture/richiami sarà visualizzata, e mostra anche come saranno i filtri per le fatture. Le fatture/richiami saranno visualizzate in ordine di creazione, le fatture/richiami in ritardo saranno visualizzate all'inizio.

I filtri si trovano in alto alla pagina, e permettono di restringere le fatture visualizzate per Data, Cliente e Importo.

A sinistra delle fatture/richiami saranno visualizzate delle icone che indicano di che cosa si tratta ad esempio una fattura in ritardo, sono presenti anche delle icone a destra, che se schiacciate permettono di eseguire una funzione riguardante le fatture/richiami. C'è ne una che permette di stampare in pdf una fattura o un richiamo, una che duplica una fattura/richiamo, e una di colore rosso che viene mostrata solo se una fattura è in ritardo e permette se ciacciata di creare un richiamo.

Fatture

Filtri

Data: da al

Cliente: Numero Cliente

Marco Bernasconi, Via canale 2, 6600 Locarno

Importo: da a

Tipologia	Creazione	Pagata	No. Stampe	Importo			
	Fattura Legna	21/07/19		5	50 CHF		
	Richiamo cibo	24/07/19		5	80 CHF		
	Fattura tubi	02/09/19		5	30 CHF		
	Fattura tubi	13/04/19	06/10/19	5	20 CHF		
	Fattura tegole	06/03/19	07/11/19	5	80 CHF		
	Fattura dadi	03/03/19	06/10/19	5	22 CHF		

Nuova Fattura

Problemi riscontrati e soluzioni adottate

-

Punto della situazione rispetto alla pianificazione

Un po' in ritardo con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Creare le interfacce per la gestione clienti e prodotti, creare gli uml per le classi.

Diario di lavoro

Luogo	Canobbio, SAM Trevano
Data	03/10/2019

Lavori svolti

Ho sistemato l'interfaccia della pagina di accesso e ho aggiunto come dovrebbe apparire l'avviso, quando un utente non abilitato cerca di fare l'accesso:

The screenshot shows a login form with a red error message at the top. The form includes fields for 'Username*' and 'Password*', both with placeholder text. A 'Registrati subito' link and a red 'Accedi' button are also present.

Accedi

Username* Bob...

Password*

Registrati subito

Accedi

Se un utente non è ancora stato accettato e prova a fare il login, non gli sarà permesso effettuarlo anche se le sue credenziali sono giuste, e sarà stampato un avviso:
“L'accesso non può essere effettuato, perché questo utente non è ancora stato abilitato
Per contattare l'amministratore si prega di inviare un email al indirizzo: administrator@email.com”

The screenshot shows a login form with a red error message box containing a large 'X'. The message states that the user cannot log in because they are not yet activated, and provides an email address for contact.

Accedi

X L'accesso non può essere effettuato, perché questo utente non è ancora stato abilitato
Per contattare l'amministratore si prega di inviare un email al indirizzo: administrator@email.com

Ho aggiornato l'interfaccia di registrazione perché mancava il campo dell'email.

Registrazione

Username Error

Email

Password

Conferma

[Ho già un account](#)

Registrati

Ho fatto il design dell'interfaccia gestione clienti:

La gestione dei clienti è fatta in una pagina a parte, in questa pagina si possono aggiungere nuovi clienti e visualizzare tutti i clienti già memorizzati.

In alto è tutta una serie di campi che sono da riempire se si vuole creare un nuovo cliente. All'inizio c'è un checkbox, che se è spuntato significa che il cliente è un'azienda invece che un privato, se si tratta di un'azienda si dovrà anche riempire il campo Nome Azienda e se no il campo sarà disabilitato.
Per creare un nuovo cliente bisogna riempire tutti i campi richiesti, e infine schiacciare sul pulsante salva per aggiungere un nuovo cliente.

Clienti

Nuovo Cliente

Azienda: Nome Azienda

Nome Cognome

Via Nap

Città Telefono

Email

Salva

Nome	Indirizzo	Telefono	Email
Rita Florence	Via arma 3, Lo...	+41 787344566	rita@gmail.com
GrandiAffari SA	Via nota 7, Lug...	+41 787344566	gr.f@gmail.com

Ho fatto il design dell'interfaccia gestione prodotti:

La gestione dei prodotti è fatta su una pagina apposta, su questa pagina è presente una sezione in alto per aggiungere un nuovo prodotto e infondo la lista dei prodotti già creati.

Per creare un prodotto basta inserire tutte informazioni nei campi e infine schiacciare il tasto **Salva**.

Come funzioni aggiuntive si potrebbe aggiungere delle icone in parte alla lista di prodotti o anche clienti, queste icone permetterebbero di modificare il prodotto/cliente già inserito, oppure potrebbero esserci delle icone che permettono di eliminare e copiare un prodotto/cliente.

Prodotti

Nuovo Prodotto

Descrizione

Prezzo

Salva

Descrizione	Prezzo
Legname grezzo	120 CHF
Impalcature	200 CHF
Sacco di cemento 10Kg	60 CHF

Problemi riscontrati e soluzioni adottate

-

Punto della situazione rispetto alla pianificazione

Un po' in ritardo con la pianificazione.

Programma di massima per la prossima giornata di lavoro

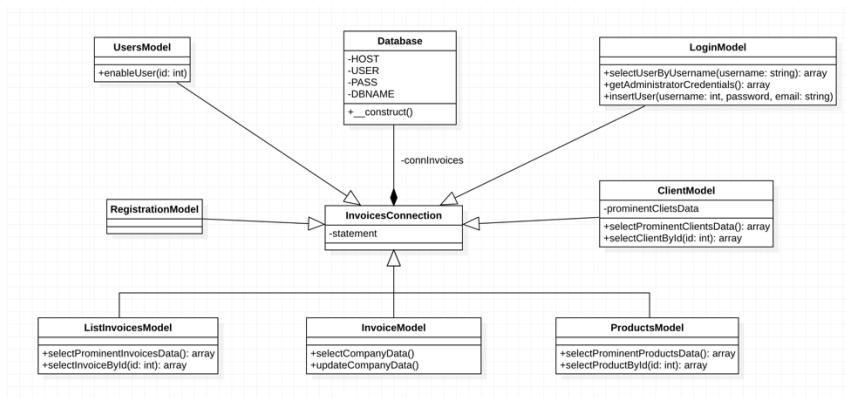
Creare gli uml per le classi che interfacciano il database e anche quelle per permettere alcune funzionalità

Diario di lavoro

Luogo	Canobbio, SAM Trevano
Data	04/10/2019

Lavori svolti

Ho iniziato a creare l'uml per le classi che andro ad implementare:



Ho fatto l'intefaccia di gestione per gli utenti, per poter abilitare gli utenti che si registrano.

Quando un utente si registra non può subito accedere, ma deve aspettare che l'amministratore abiliti il suo *account*, per fare questo l'amministratore può entrare in questa pagina.

In questa pagina gli utenti non abilitati verranno visualizzati tramite una lista, e per poterli abilitare basta mettere la spunta sul *checkbox* Abilita, e infine premere il pulsante Salva.

Utenti

Utenti non abilitati

Salva

Username	Email	
Giulio.Protopet	giulio.proto@gmail.com	<input checked="" type="checkbox"/> Abilita
Gustavo.Inox	gusti.in@gmail.com	<input type="checkbox"/> Abilita
Davide.Umbri	d.umbri@gmail.com	<input type="checkbox"/> Abilita

Quando non ci sono utenti da abilitare, uscirà il messaggio “Non ci sono utenti da abilitare”.

Utenti

Utenti non abilitati

Salva

Non ci sono utenti da abilitare

Problemi riscontrati e soluzioni adottate

-

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Creare gli uml per le classi che interfacciano il database e anche quelle per permettere alcune funzionalità

Diario di lavoro

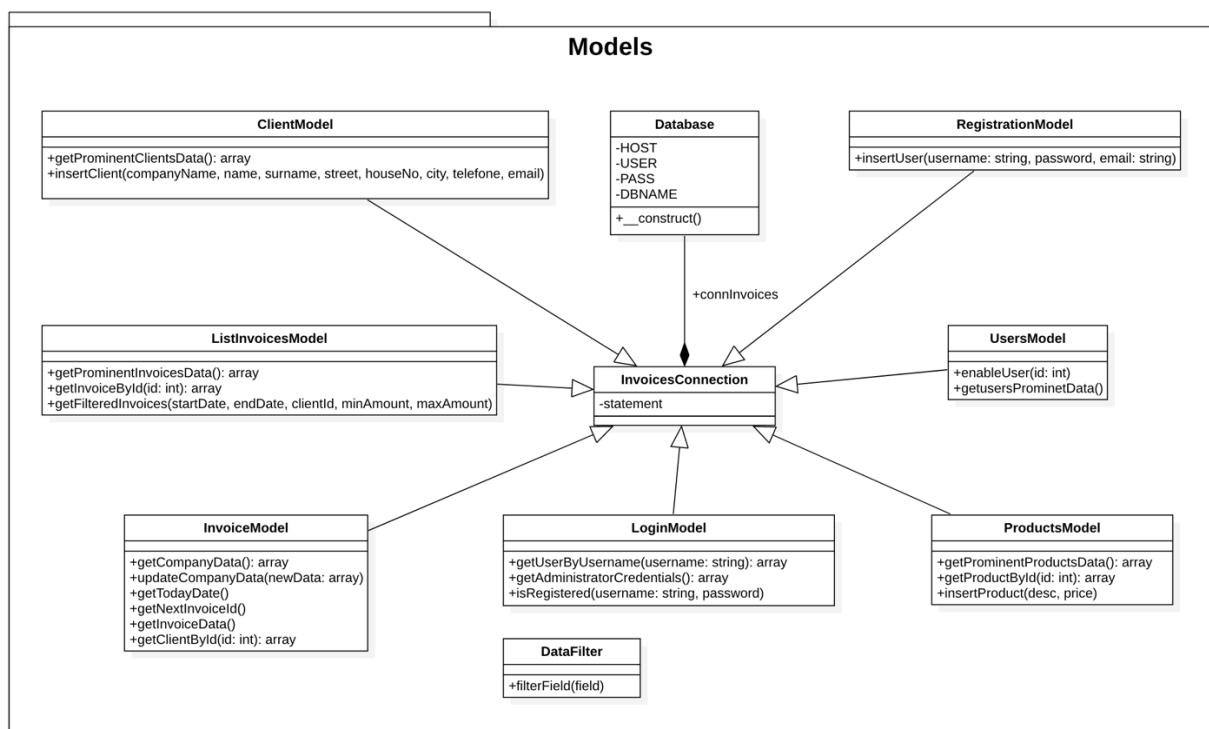
Luogo	Canobbio, SAM Trevano
Data	08/10/2019

Lavori svolti

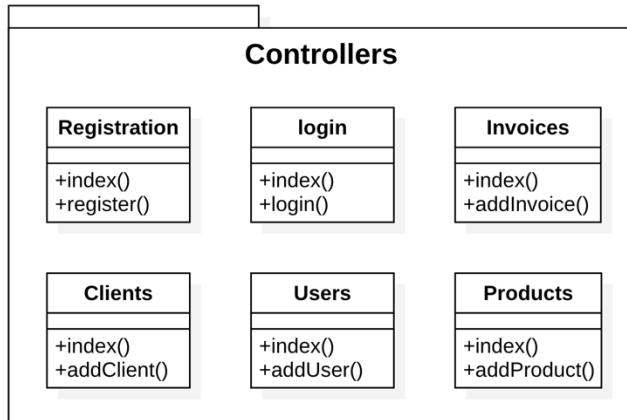
Ho continuato l'uml delle classi in php che svolgono il compito di **Model** seguendo il *pattern MVC*

Questo UML specifica le classi che avranno il compito di Model (seguendo il pattern MVC), come si può vedere la maggior parte delle classi deve comunicare con il database, per questo queste classi hanno una connessione con il database (Invoices).

Tutte le classi possiedono i metodi necessari, che serviranno ad aiutare i Controller a svolgere tutte le operazioni necessarie.



Ho fatto l'uml delle classi in php che svolgono la funzione di **Controller** segunedo il pattern **MVC**.



Problemi riscontrati e soluzioni adottate

-

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Creare gli uml per le classi mancanti

Diario di lavoro

Luogo	Canobbio, SAM Trevano
Data	10/10/2019

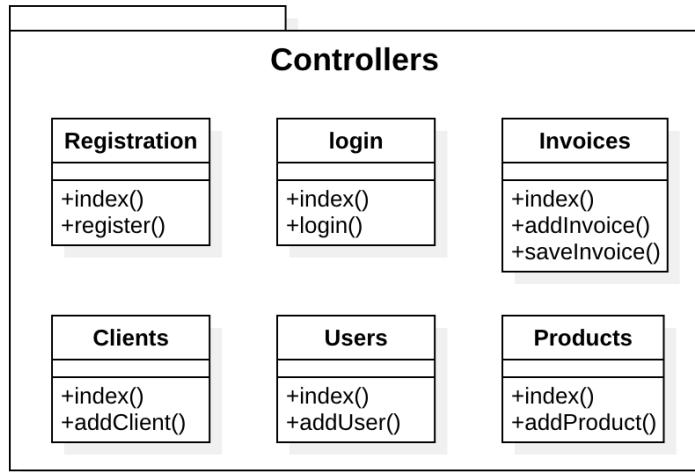
Lavori svolti

Ho messo apposto la documentazione, sistemando alcune imprecisioni o piccole mancanze.

Ho sistemato l'uml delle classi Controller, e ho descritto quello che serviva specificare.

Questo UML specifica le classi che avranno il compito di Controller, cioè quelle che utilizzano le classi Model per effettuare delle operazioni su delle View (Pagine Web) precise.

Ognuna di queste classi si occupa di gestire le operazioni svolte su una pagina web, con l'eccezione di "Invoices" che si occupa sia di "Interfaccia Lista Fatture/Richiami" sia di "Interfaccia Creazione Fattura/Richiamo".



Ho finito di implementare il database.

Problemi riscontrati e soluzioni adottate

Creando il database ho scoperto che impostare una colonna di tipo int specificando la grandezza non sarà più possibile nelle prossime versioni di MySql, quindi nei casi dove usavo questo metodo ho specificato il tipo int senza specificare la grandezza, come avevo fatto per il NAP della citta "int(5)".

<https://gdsotirov.blogspot.com/2019/07/new-features-in-mysql-8017.html>

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Aggiugere la tabella che descrive le colonne della tabella “client_company”.

Aggiornare lo schema della struttura dati del database, e lo schema ER.

Aggiungere nelle interfacce il campo “tipologia di fattura”, perché manca.

Iniziare l’implementazione sito web e definire pattern MVC.

Diario di lavoro

Luogo	Canobbio, SAM Trevano
Data	11/10/2019

Lavori svolti

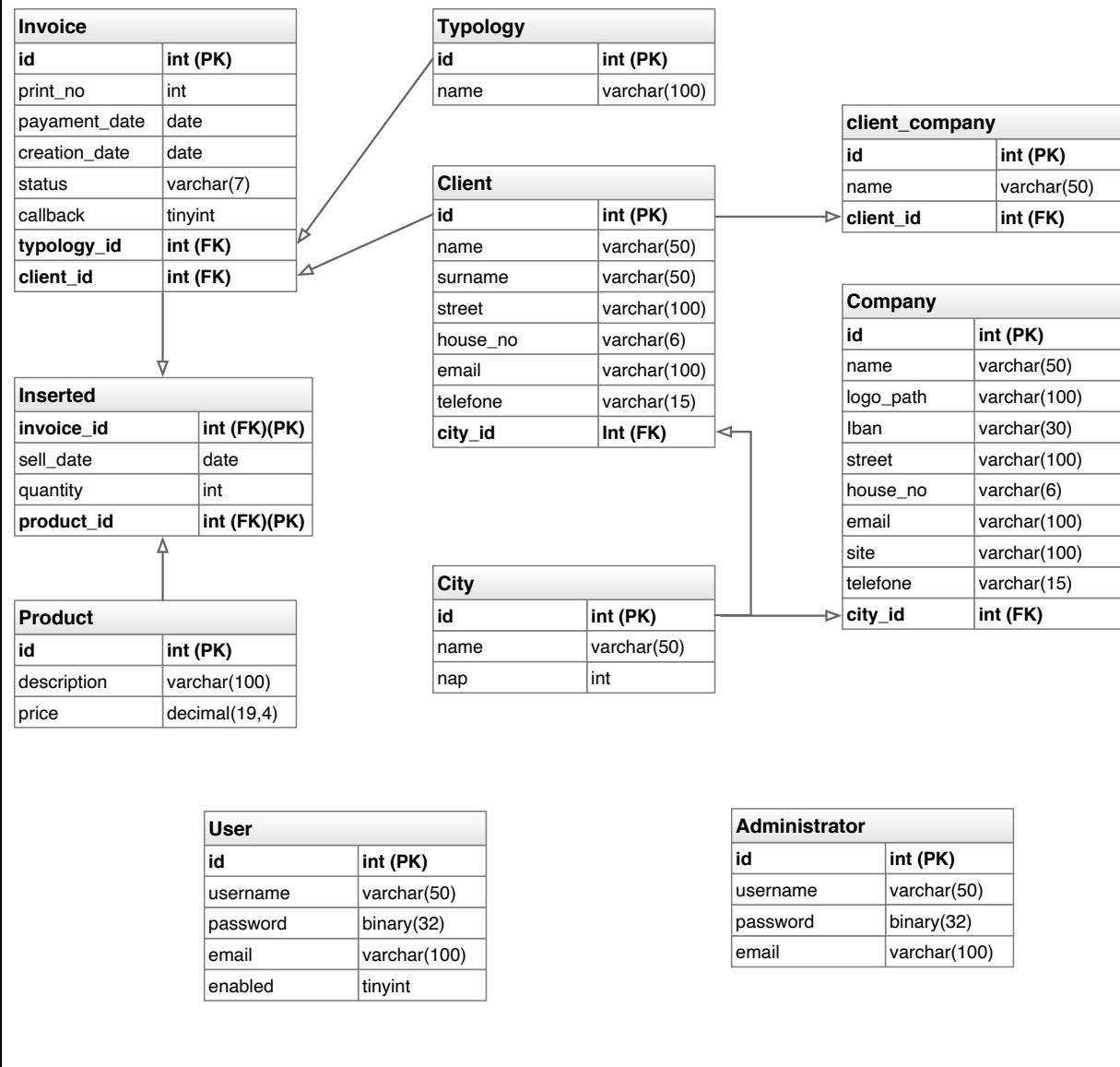
Le prime due ore il professore Valsangiacomo ci ha presentato e spiegato LPI.

Ho la tabella che descrive la tabella “client_company”, e ho sistemato anche le altre tabelle.

La tabella client_company contiene il nome dell'azienda del cliente, se si tratta di un cliente giuridico.

Client_company		
Nome del Dato	Tipo	Descrizione
id	(PK) int	Identificativo di questa tabella.
name	varchar(50)	Nome dell'azienda del cliente.
client_id	(FK) int	Identificativo del cliente associato.

Ho aggiornato lo schema della struttura dati del database, e lo schema ER.



Problemi riscontrati e soluzioni adottate

nussuno

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Aggiungere nelle interfacce il campo "tipologia di fattura", perché manca.

Iniziare l'implementazione sito web e definire pattern MVC.

Diario di lavoro

Luogo	Canobbio, SAM Trevano
Data	15/10/2019

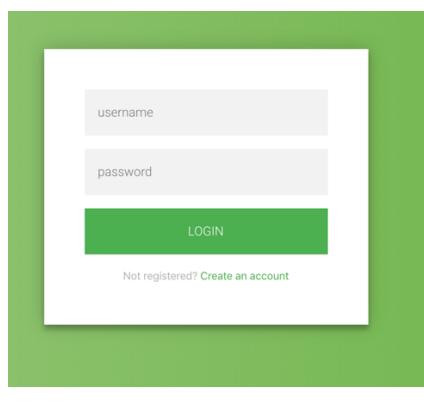
Lavori svolti

Ho aggiunto nel interfaccia, che quando si crea una fattura si puo associargli una tipologia:

The screenshot shows a software interface for creating an invoice ('Fattura'). It includes fields for customer information ('Nome azienda', 'Cognome', 'Via', 'Cap'), a 'Tipologia' dropdown, and a 'Data' field. A table lists items with columns 'Descrizione', 'Q.tà', 'Prezzo uni.', and 'Importo'. At the bottom is a red 'Salva' button.

Ho iniziato l'implementazione sito web e definire pattern MVC.

Ho iniziato anche a creare l'interfaccia di login



Problemi riscontrati e soluzioni adottate

nussuno

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Continuare l'implementazione, completare l'interfaccia di login e di registrazione

Diario di lavoro

Luogo	Canobbio, SAM Trevano
Data	17/10/2019

Lavori svolti

Sono adato a cercare un template di una navbar che soddisfasse le mie necessità:

link: <https://bootsnipp.com/snippets/nNX3a>

Ho fatto in modo che i tutti campi del login sono obbligatori, utilizzando **required** di html, quindi si potrebbe fare dei controlli personalizzati per rendere la gestione degli errori più coerente con il resto, che viene fatto con php.

Per le icone da utilizzare sul sito ho scelto quelle di “Font Awesome”

Link: <https://fontawesome.com/how-to-use/on-the-web/setup/hosting-font-awesome-yourself>

Per salvare il percorso assoluto di certi file o librerie, uso le costanti di PHP. Possono essere utilizzate dovunque in qualsiasi file PHP senza importare nessuna classe. E facilitano il tutto soprattutto quando vengono aggiornate o spostati dei file/librerie.

```
/**  
 * jQuery URL  
 *  
 * Needed for Bootstrap and oder JS files  
 * Depends on VENDER_URL  
 * @link https://jquery.com  
 */  
define('JQUERY_URL', VENDOR_URL.'jquery/jquery-3.4.1.min.js');
```

Per utilizzare tali costanti, basta stamparle con php all'interno dei tag “<link>” per i CSS e dentro i tag “<Script>”

```
<!-- Navbar CSS -->  
<link rel="stylesheet" type="text/css" href="<?php echo CSS_URL?>navbar.css">
```

```
<!-- jQuery -->  
<script src="<?php echo JQUERY_URL?>"></script>
```

Problemi riscontrati e soluzioni adottate

nussuno

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Continuare l'implementazione, completare l'interfaccia di login e di registrazione

Diario di lavoro

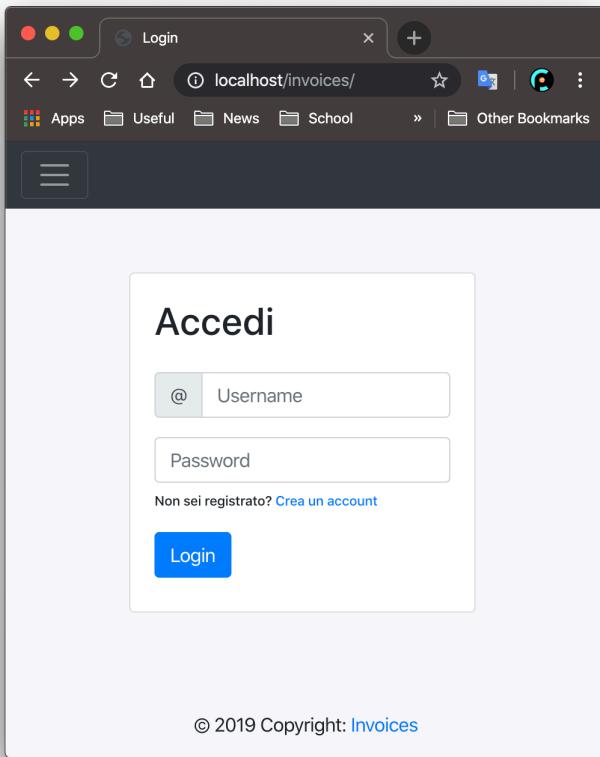
Luogo	Canobbio, SAM Trevano
Data	18/10/2019

Lavori svolti

Ho sistemato il codice html e ho utilizzato bootstrap per migliorare lo stile della pagina di login.

Ho aggiunto il footer e ho migliorato l'aspetto del form di registrazione.

Ho sistemato la navbar e i suoi vari link.



Problemi riscontrati e soluzioni adottate

nussuno

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Continuare l'implementazione, completare l'interfaccia di login e di registrazione

Diario di lavoro

Luogo	Canobbio, SAM Trevano
Data	22/10/2019

Lavori svolti

Ho sistemato il pattern MVC, aggiungendo qualche altra classe.

Dentro alla cartella del progetto è presente il pattern MVC, che è definito all'interno della cartella app.

Al difuori della cartella app, ci sono i file:

- **".htaccess"** che regola che cosa può essere visualizzato dall'esterno

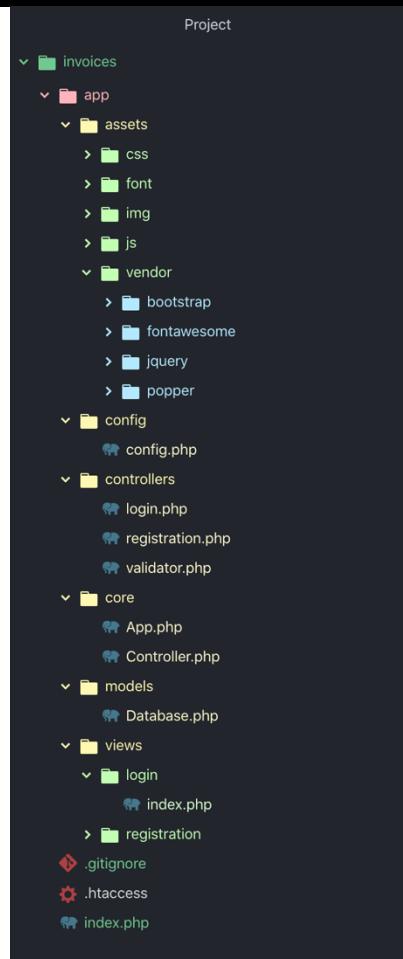
```
#Prevents access to MVC important folders like "App",
#this is very important for security
Options -Indexes
```

```
# With this you can name every page "index.php" just "index"
Options -MultiViews
RewriteEngine On
```

```
# Allows you to pinpoint the index.php files,
# we want to call on the "RewriteRule"
RewriteBase /invoices/
```

```
# If the condition is true, it will rewrite directory, file, link
RewriteCond %{REQUEST_FILENAME} !-d
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-l
```

```
# Provides the App class to take the url string
RewriteRule ^(.+)\$ index.php?url=\$1 [QSA,L]
```



- **"Index.php"** È il primo file che viene caricato, ha il compito di caricare le classi all'interno della cartella "app/core/" e le classi all'interno di "app/config/" e di avviare il sistema.

```
// Import the config file
require 'app/config/config.php';

// Import the application main classes
require_once 'app/core/App.php';
require_once 'app/core/Controller.php';

// Start the application
$app = new App;
```

Problemi riscontrati e soluzioni adottate

nussuno

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Continuare l'implementazione, completare l'interfaccia di registrazione; documentare il codice

Diario di lavoro

Luogo	Canobbio, SAM Trevano
Data	24/10/2019

Lavori svolti

All'interno della classe **Model** “Validator” sono presenti diversi tipi di validazione sui dati o altri metodi che aiutano la validazione.

Questo metodo permette di filtrare in generale qualsiasi dato, togliendo qualsiasi carattere illegale che potrebbe comportare un inaspettato comportamento del sito.

```
/**  
 * Validate data of any kind.  
 *  
 * @param data The data to validate  
 * @return boolean The validated data  
 */  
private function generalValidation($data)  
{  
    $data = trim(stripslashes(htmlspecialchars($data)));  
    return $data;  
}
```

Ho cercato una soluzione su internet, di come sapere se una email è valida o non.

Link: <https://stackoverflow.com/questions/5855811/how-to-validate-an-email-in-php>

Questo metodo permette di sapere, se una email è valida strutturalmente e sintatticamente. Se ritorna “**TRUE**” allora significa che è valida, ma deve essere comunque filtrata in ogni caso.

```
/**  
 * Return true if the given email is valid, structurally and syntactically.  
 *  
 * @param email The email to check the validation  
 * @return boolean If the email is valid return true otherwise false  
 */  
function isValidEmail($email)  
{  
    return filter_var($email, FILTER_VALIDATE_EMAIL) &&  
        preg_match('/@.+\. /', $email);  
}
```

Questo metodo filtra i caratteri dell'email e se la filtrazione fallisce ritorna “**FALSE**”, invece se ci riesce ritorna il testo filtrato.

```
filter_var($email, FILTER_VALIDATE_EMAIL)
```

Questa parte invece verifica la struttura dell'email, verificando che i caratteri obbligatori ci siano e nella posizione giusta all'interno dell'email.

```
preg_match('/@.+\. /', $email);
```

Questo metodo ritorna l'email validata, grazie al metodo "filter_var(<variabile>,<tipoFiltro>)" che filtra ogni singolo carattere dell'email, e se la validazione fallisce ritorna "FALSE".

```
/**  
 * Validate data of type string.  
 *  
 * @param email The email to validate  
 * @return boolean The validated email  
 */  
public function validateEmail($email)  
{  
    $validEmail = $this->generalValidation($email);  
    return filter_var($validEmail, FILTER_VALIDATE_EMAIL);  
}
```

Questo metodo permette di validare un numero intero, inserendo un qualsiasi numero ritorna un intero validato.

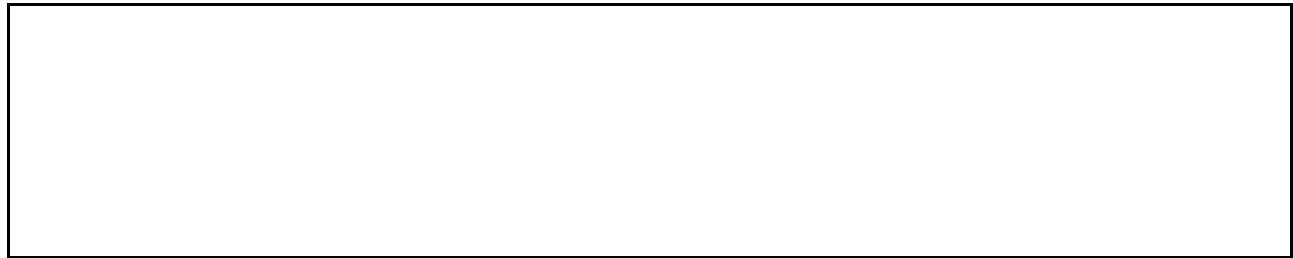
```
/**  
 * Validate data of type int.  
 *  
 * @param i The integer number to validate  
 * @return boolean The validated data  
 */  
public function validateInt($i)  
{  
    $validElement=$this->generalValidation($element);  
    return intval($validElement);  
}
```

Questo metodo permette di validare una stringa, inserendo un testo qualsiasi ritorna una stringa filtrata.

```
/**  
 * Validate data of type string.  
 *  
 * @param str The string to validate  
 * @return boolean The validated string  
 */  
public function validateString($str)  
{  
    $validStr = $this->generalValidation($str);  
  
    $pattern = '/^([A-Za-z0-9_-àèìòùÀÈÌÒÙáéíóúýÁÉÍÓÚÝâéîôûÂÊÎÔÛâñõÃÑÕâéïöüýÄËÏÖÜÝççßøÅåÆæœé]*)$/';  
  
    if (!preg_match($pattern, $validStr))  
    {  
        $validStr = strval($validStr);  
    }  
    return $validStr;  
}
```

Ho fatto questa *query* sul database, per inserire le credenziali dell'amministratore nel DB.

```
insert into administrator values(null,'Administrator',sha2('Invoices2019',256),'administrator@gmail.com');
```



Problemi riscontrati e soluzioni adottate

Ho cambiato il tipo di dato in cui è memorizzata una password nel database, sia per la tabella “User” che della tabella “Administrator”, ora si tratta di un char(64). Questo cambiamento è stato necessario perché il tipo binary(32) è troppo piccolo, per memorizzare una hash codificata in sha256.

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Continuare l’implementazione, completare l’interfaccia di registrazione e login; documentare il codice

Diario di lavoro

Luogo	Canobbio, SAM Trevano
Data	25/10/2019

Lavori svolti

Ho aggiunto alcuni utenti per verificare il comportamento del login.

```
insert into user values
(null,"PeterCatania",sha2('Password&1',256),'petercat721@gmail.com', 0),
(null,"Giacomo2000",sha2('Password&1',256),'gicaomo@gmail.com', 0),
(null,"RobertoMand",sha2('Password&1',256),'ro.mand@gmail.com', 1),
(null,"SherryHaibara",sha2('Password&1',256),'ai.home@gmail.com', 1);
```

Ho fatto in modo che quando un utente registrato ma non abilitato prova a fare login, sarà portato ad un'altra pagina con un messaggio di errore, e schiacciando sul bottone X potrà ritornare al login.

Questa parte è la parte di login, che controlla quello che ha inserito l'utente, a seconda del tipo di utente che è, verrà indirizzato in una pagina diversa.

```
session_start();
// Effettuate the login, if is submit a POST request
if (isset($_POST['login']))
{
    // Import the Validator Model class, and inizialize a new instance.
    $this->model('Validator');
    $validator = new Validator();

    // get the validated username, and the hash of the password, from login form
    $username = $validator->validateString($_POST['username']);
    $password = hash('sha256', $_POST['password']);

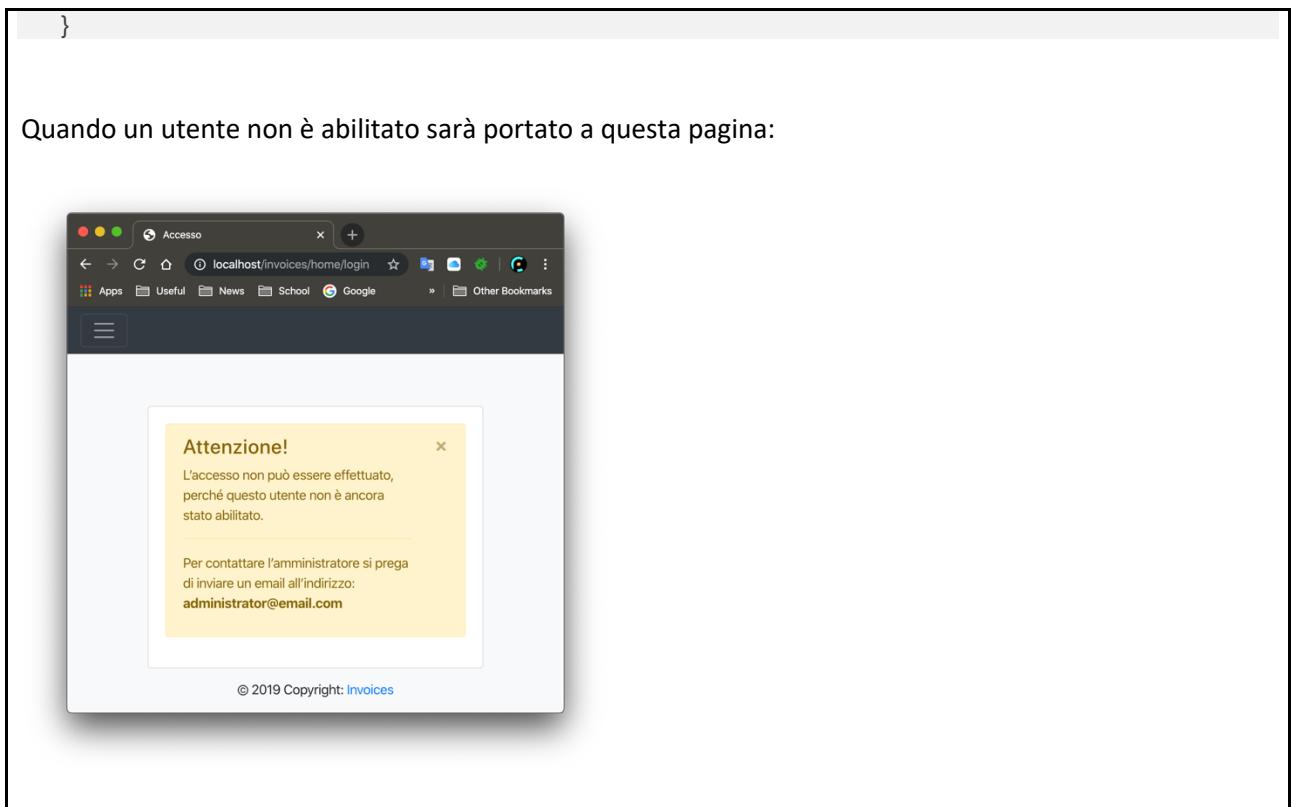
    // Import the LoginModel Model class, and inizialize a new instance.
    $this->model('LoginModel');
    $loginModel = new LoginModel();

    // get the user with the same credentials from the login
    $user = $loginModel->getUserByUsernameAndPassword(
        $username,
        $password
    );

    // if the query have find the user with the login credentials.
    if($user)
    {
        // get the first array, in this case is our user
        $user = $user[0];

        // save the user data in the session
        $_SESSION['user'] = $user;

        if($user['enabled'] == 0)
        {
            $this->view('home/disableUser');
        }else
        {
            $this->view('invoices/index');
        }
    }
}
```



Problemi riscontrati e soluzioni adottate

nessuno

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione.

Programma di massima per la prossima giornata di lavoro

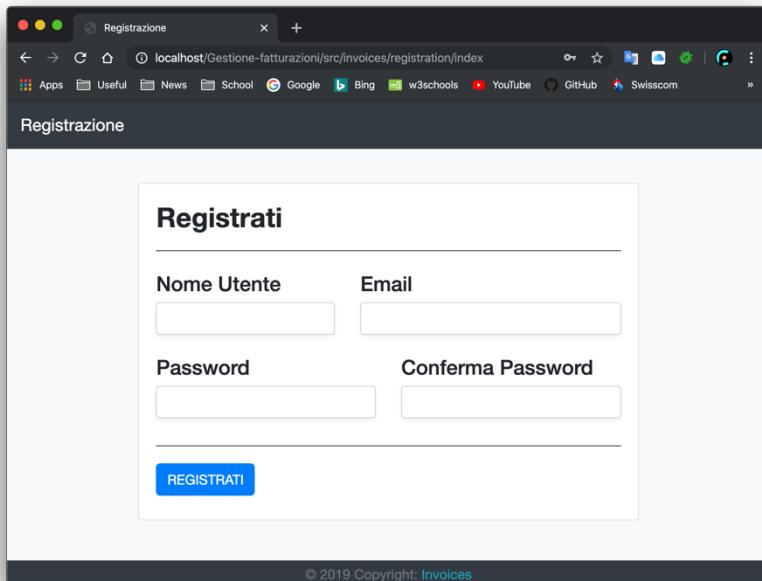
Continuare l'implementazione, completare l'interfaccia di registrazione; documentare il codice

Diario di lavoro

Luogo	Canobbio, SAM Trevano
Data	05/11/2019

Lavori svolti

Ho creato l'interfaccia per la pagina di registrazione, che è stata fatta seguendo la progettazione.



Ho creato due nuove costanti in PHP, che indicano il nome delle classi che vengono utilizzate per definire se un **input HTML** è valido o invalido. Se un input contiene la classe **invalid** varrà mostrato il corrispettivo messaggio di errore.

```
/**  
 * Define the CSS classes, that indicate that a input is valid or invalid.  
 * This classes show the messages contained in the HTML.  
 */  
define('VALID','is-valid');  
define('INVALID','is-invalid');
```

Problemi riscontrati e soluzioni adottate

nessuno

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione.

Programma di massima per la prossima giornata di lavoro

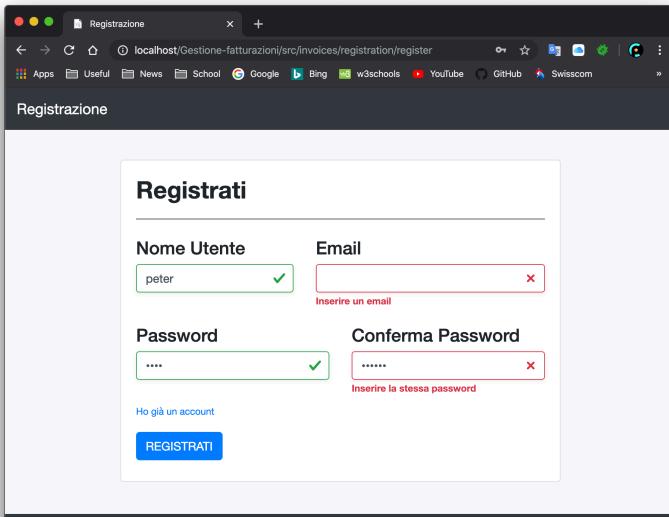
Continuare l'implementazione, completare l'interfaccia di registrazione; documentare il codice

Diario di lavoro

Luogo	Canobbio, SAM Trevano
Data	05/11/2019

Lavori svolti

**Ho continuato la pagina di registrazione,
ho fatto in modo che venissero controllati i campi:**



La prima cosa è prendere i valori che sono arrivati con il **Submit** del form di registrazione, e poi validare il loro contenuto e trasformare le password in hash

```
// get the validated field, and the hash of the passwords, from login form
$username = $validator->validateString($_POST['username']);
$_SESSION['username'] = $username;
$email = $validator->validateEmail($_POST['email']);
$_SESSION['email'] = $email;
$password = hash('sha256', $_POST['password']);
$_SESSION['password'] = $_POST['password'];
$confirmedPassword = hash('sha256', $_POST['confirmedPassword']);
$_SESSION['confirmedPassword'] = $_POST['confirmedPassword'];
```

Dopo arriva un controllo che si occupa di capire quali campi siano validi e quali non, se i campi sono vuoti o non contengono i valori adeguati sono considerati invalidi, e la password e anche considerata invalida quando non corrisponde a quella di conferma.

```
if (empty($username)) {
    $_SESSION['usernameCSSValidityClass'] = INVALID;
} else {
    $_SESSION['usernameCSSValidityClass'] = VALID;
}

if (empty($email)) {
    $_SESSION['emailCSSValidityClass'] = INVALID;
} else {
    $_SESSION['emailCSSValidityClass'] = VALID;
}
if (
$confirmedPassword ==
'e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855'
) {
    $_SESSION['passwordCSSValidityClass'] = INVALID;
} else {
    $_SESSION['passwordCSSValidityClass'] = VALID;
}

if (
$confirmedPassword ==
'e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855' ||
$confirmedPassword != $password
) {
    $_SESSION['confirmedPasswordCSSValidityClass'] = INVALID;
} else {
    $_SESSION['confirmedPasswordCSSValidityClass'] = VALID;
}
```

Alla fine viene mostrato il messaggio di errore, inserendo tramite PHP una classe CSS che indica che quel campo è valido o non , e quindi deve mostrare il messaggio di invalidità se necessario.

```
class="form-control mt-1 shadow-sm <?= $_SESSION['emailCSSValidityClass'] ?>"
```

Problemi riscontrati e soluzioni adottate

nessuno

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Continuare l'implementazione, completare l'interfaccia di registrazione; documentare il codice

Diario di lavoro

Luogo	Canobbio, SAM Trevano
Data	08/11/2019

Lavori svolti

Ho continuato e finito la registrazione

Se tutti i campi delle registrazioni sono validi, viene inserito il nuovo utente nel database e torna alla pagina di login.

Se invece i campi non sono validi vengono mostrati gli errori.

```
if (!$allFieldAreValid) {
    // return to the default page for registration
    $this->view('registration/index');
} else {
    // import the Validator Model class, and initialize a new instance
    $this->model('RegistrationModel');
    $registrationModel = new RegistrationModel();

    // insert the user data in the database
    $registrationModel->insertUser($username, $password, $email);

    // unset the no longer needed var, memorised in the Session
    unset($_SESSION['username']);
    unset($_SESSION['email']);
    unset($_SESSION['password']);
    unset($_SESSION['confirmedPassword']);
    unset($_SESSION['usernameCSSValidityClass']);
    unset($_SESSION['emailCSSValidityClass']);
    unset($_SESSION['passwordCSSValidityClass']);
    unset($_SESSION['confirmedPasswordCSSValidityClass']);

    // return to the login form
    $this->view('home/index');
}
```

Ho commentato il codice e ho iniziato la gestione utenti.

Problemi riscontrati e soluzioni adottate

nessuno

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Continuare l'implementazione, finire lista utenti e gestione,

Diario di lavoro

Luogo	Canobbio, SAM Trevano
Data	12/11/2019

Lavori svolti

Ho messo apposto alcuni elementi nelle view.

Ho creato ogni singolo controller e view che sono necessari per il sito web.

Ho creato una classe che implementa funzioni utili al controllo: se un utente ha fatto login o non.

Ho implementato un metodo per fare il log out.

```
/**  
 * Effetuate the logout  
 */  
public function logout()  
{  
    // unset all saved session variables  
    $_SESSION = array();  
  
    // require the default page  
    $this->view('home/index');  
}
```

Problemi riscontrati e soluzioni adottate

nessuno

Punto della situazione rispetto alla pianificazione

Un po in ritardo con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Continuare l'implementazione, finire lista utenti e gestione,

Diario di lavoro

Luogo	Canobbio, SAM Trevano
Data	14/11/2019

Lavori svolti

Ho completato la pagina per gestire gli utenti che non sono stati abilitati.

Nome Utente	Email	
Angelo	Corinto@gmail.com	<input type="checkbox"/> Abilita
Angelo	Corinto@gmail.com	<input type="checkbox"/> Abilita
Piero123	piro.Fagiani@gmail.com	<input type="checkbox"/> Abilita
MiaVaccaro-5	mia.vaccaro@gmail.com	<input type="checkbox"/> Abilita

Prendo gli utenti dal database e poi li passo alla view che li mostra, e li permette di abilitarli.

```
// instance a new object of the model class "UsersModel"  
$this->model("UsersModel");  
$usersModel = new UsersModel();  
  
// the array that contain the Users saved in the database.  
$users = $usersModel->getNotEnabledUsers();  
  
// require the default page  
$this->view('users/index', ['users' => $users]);
```

Per abilitarli eseguo una query che abilita tutti gli utenti, con gli stessi id degli utenti che sono presenti. nel checkbox spuntati.

```
if (isset($_POST['enable'])) {  
    //get the array that contain the ids of the users to enable  
    $userIdToEnable = $_POST['userIdToEnable'];  
  
    // instance a new object of the model class "UsersModel"  
    $this->model("UsersModel");  
    $usersModel = new UsersModel();  
  
    //enable all users by id  
    foreach ($userIdToEnable as $userId) {  
        $usersModel->enableUserById($userId);  
    }
```

Problemi riscontrati e soluzioni adottate

nessuno

Punto della situazione rispetto alla pianificazione

Un po' in ritardo con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Continuare l'implementazione, continuare con la pagina di gestione dei prodotti.

Diario di lavoro

Luogo	Canobbio, SAM Trevano
Data	15/11/2019

Lavori svolti

Nella classe Controller che fa parte delle classi centrali del pattern MVC, ho aggiunto alcuni metodi per permettere ad ogni controller di gestire alcune situazioni. Queste situazioni si presentano quando un metodo o più di un controller possono essere accessibili solo da particolari utenti, nel nostro caso sono presenti dei metodi che devono essere accessibili solo dall'amministratore.

Per fare questo ho creato questo metodo che consente di ritornare alla *homepage* se nessuno si è ancora registrato.

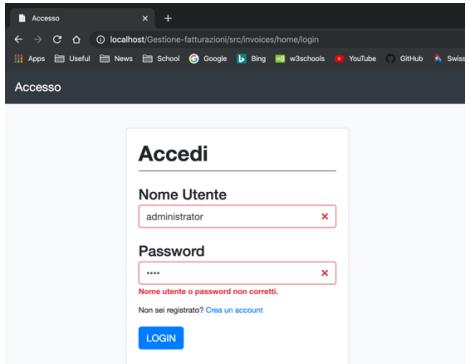
```
/*
 * Check if the login data of a user or of the administrator
 * is saved in the session.
 */
private function existsLoginSessionData()
{
    return isset($_SESSION[USER_SESSION_DATA]) ||
           isset($_SESSION[ADMINISTRATOR_SESSION_DATA]);
}

/*
 * Redirect to the home page, if the login is not been effectuated by anyone.
 */
public function redirectToHomePagelfAnyonelsLogged()
{
    if (!$this->existsLoginSessionData()) {
        header("Location: " . URL . "home/index");
    }
}
```

Ho implementato anche questo metodo, che permette di ritornare automaticamente alla *homepage*, se è registrato un **Utente** o nessuno si è ancora registrato.

```
/*
 * Redirect to the home page, if is logged a user or anyone.
 *
 * @param session The current session saved on the server.
 */
public function redirectToHomePagelfUserOrAnyonelsLogged()
{
    if (
        isset($_SESSION[USER_SESSION_DATA]) ||
        !$this->existsLoginSessionData()
    ) {
        header("Location: " . URL . "home/index");
    }
}
```

Nella pagina di registrazione mostro un errore, se l' username o la password non corrispondono a nessun Utente registrato nel database.



Problemi riscontrati e soluzioni adottate

nessuno

Punto della situazione rispetto alla pianificazione

Un po' in ritardo con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Continuare l'implementazione, continuare con la pagina di gestione dei prodotti.

Diario di lavoro

Luogo	Canobbio, SAM Trevano
Data	19/11/2019

Lavori svolti

Ho fatto la pagina che gestisce i prodotti, e che può aggiungere nuovi prodotti

Prodotti

Nuovo prodotto

Descrizione

Prezzo

SALVA

Descrizione	Prezzo
Legna	0.0000
Comignolo	50.0000

Questo codice permette di prendere i prodotti salvati sul database

```
/**  
 * Get the saved products in the database.  
 */  
public function getProducts()  
{  
    // prepare the query, that get all the not enabled users  
    $selectProducts = 'select id,description,price from product';  
    $stmt = $this->connInvoices->prepare($selectProducts);  
  
    // the query statement is executed and returned  
    $stmt->execute();  
    return $stmt->fetchAll(PDO::FETCH_ASSOC);  
}
```

Questo codice permette di inserire un nuovo utente all'interno del database

```
/**  
 * Insert a new product in the database.  
 *  
 * @param description the description of the new product  
 * @param price the price of the new product  
 */  
public function saveProduct($description, $price)  
{  
    // prepare the query, to insert a new product in the database  
    $selectUserById = "insert into product (id, description, price) values (null, :description, :price)";  
    $stmt = $this->connInvoices->prepare($selectUserById);  
  
    // insert in the query the id of the user  
    $stmt->bindParam(':description', $description);  
    $stmt->bindParam(':price', $price);  
  
    // the query statement is executed  
    $stmt->execute();  
}
```

Problemi riscontrati e soluzioni adottate

nessuno

Punto della situazione rispetto alla pianificazione

Un po' in ritardo rispetto alla pianificazione.

Programma di massima per la prossima giornata di lavoro

Continuare l'implementazione, continuare con la pagina di gestione dei clienti.

Diario di lavoro

Luogo	Canobbio, SAM Trevano
Data	21/11/2019

Lavori svolti

Ho iniziato ha implementare le pagine per gestire i clienti

Ho implementato un metodo che fa parte della classe model “ClientModel”, che permette di aggiungere un nuovo cliente all’interno del database.

```
/*
 * Insert a new client in the database, also it might be a company.
 *
 * @param clientName the new name of the client, or the responsible of the company
 * @param clientSurname the new surname of the client, or the responsible of the company
 * @param street the new street of the client address
 * @param houseNo the new client house_no of the client address
 * @param city the new city of the client address
 * @param nap the new nap of the client address
 * @param telephone the new telephone of the client
 * @param email the new email of the client
 * @param companyName the new company name of the client, if the client is a company
 */
public function saveClient(
    $clientName,
    $clientSurname,
    $street,
    $houseNo,
    $city,
    $nap,
    $telephone,
    $email,
    $companyName = null
) {
    // prepare the query, to insert a new city in the database
    $insertCity = "insert into city (id, name, nap) values (null, :city, :nap)";
    $stmt = $this->connInvoices->prepare($insertCity);

    // insert in the query, the data of the new city
    $stmt->bindParam(':city', $city);
    $stmt->bindParam(':nap', $nap);

    // the query statement that insert the new city is executed
    $stmt->execute();

    // get the id of the new city
    $selectNewCityId = "select id from city where nap = :nap";
    $stmt = $this->connInvoices->prepare($selectNewCityId);
    $stmt->bindParam(':nap', $nap);
    $cityId = $stmt->execute();

    // prepare the query, to insert a new client in the database
    $insertClient = "
        insert into client (id, name, surname, street, house_no, telephone, email)
        values (null, :clientName, :clientSurname, :street, :house_no, :telephone, :email)
    ";
    $stmt = $this->connInvoices->prepare($insertClient);

    // insert in the query, the data of the new client
}
```

```
$stmt->bindParam(':clientName', $clientName);
$stmt->bindParam(':clientSurname', $clientSurname);
$stmt->bindParam(':street', $street);
$stmt->bindParam(':house_no', $houseNo);
$stmt->bindParam(':telephone', $telephone);
$stmt->bindParam(':email', $email);

// the query statement that insert the new client is executed
$stmt->execute();

if ($CompanyName) {
    //prepare the query, to insert a new company name in the database
    $insertCompanyName = "
        insert into client_company (id, name)
        values (null, :companyName)
    ";
    $stmt = $this->connInvoices->prepare($insertCompanyName);

    // insert in the query, the new company name
    $stmt->bindParam(':companyName', $CompanyName);

    // the query statement that insert the new company name is executed
    $stmt->execute();
}
}
```

Problemi riscontrati e soluzioni adottate

1)

Quando un utente dopo aver eseguito un'azione tramite un metodo che usa una richiesta HTTP POST, e cerca di ricaricare la pagina eseguirà l'invio di una nuova richiesta http POST, questo provoca l'inserimento di duplicati nel *database* se l'azione eseguita è un *insert*. Può provocare diversi problemi duplicando il prodotto ottenuto dalla esecuzione della richiesta.

Alla fine di ogni operazione eseguita con il contenuto della richiesta di POST, ho inserito questa funzione che impedisce di eseguire lo stesso metodo del controller, visto che reindirizza la pagina al metodo di default, cioè "index".

```
// redirect to the default method of the products page
$this->redirectToPage('products');
```

Questo metodo è all'interno della classe Controller che a sua volta viene estesa da tutte le classi di tipo Controller; questo è il contenuto del metodo:

```
/**
 * Redirect to a page that is related with the given controller, and the method.
 */
public function redirectToPage(
    $controller = DEFAULT_CONTROLLER,
    $method = DEFAULT_METHOD
) {
    header("Location: " . URL . $controller . "/" . $method);
    exit;
}
```

2)

ho aggiunto questi due metodi disponibili per qualunque classe Controller, che si occupano rispettivamente, di evitare che un utente non registrato possa vedere pagine che non può, e un utente registrato che non sia l'amministratore possa entrare in una pagina in cui non dovrebbe avere accesso.

```
// prevents that anyone that is not logged enter this page, and execute this method  
$this->redirectToHomePagelfAnyonelsLogged();
```

```
// prevents that users accounts can access this page, and execute this method  
$this->redirectToUserDefaultPermittedPagelfUserlsLogged();
```

Il contenuto di questi metodi sono rispettivamente:

```
/**  
 * Redirect to the home page, if the login is not been effectuated by anyone.  
 */  
public function redirectToHomePagelfAnyonelsLogged()  
{  
    if (!$this->existsLoginSessionData()) {  
        $this->redirectToPage();  
    }  
}  
  
/**  
 * Redirect to the the user default permitted page,  
 * that a user can access, if a user is logged.  
 *  
 * @param session the current session saved on the server.  
 */  
public function redirectToUserDefaultPermittedPagelfUserlsLogged()  
{  
    if (  
        !isset($_SESSION[USER_SESSION_DATA])  
    ) {  
        $this->redirectToPage('invoices');  
    }  
}
```

3)

Ho inserito un trigger nel database MySQL, che evita che le stesse città vengano inserite nella tabella "city", dove vengono salvati i nomi delle città e i loro nap. Questo trigger esegue un controllo del insert e se individua una città con lo stesso nome, l'insert non sarà eseguito ma verrà restituito un codice e un messaggio di errore.

```
DROP TRIGGER IF EXISTS `before_city_insert`;

DELIMITER $$

CREATE TRIGGER before_city_insert
BEFORE INSERT
ON city FOR EACH ROW
BEGIN
    DECLARE sameCityCount INT;

    SELECT COUNT(*)
    INTO sameCityCount
    FROM city c
    WHERE NEW.nap = nap;

    IF sameCityCount > 0 THEN
        SET @message_text = CONCAT('This city already exists, with this nap: ',NEW.nap);
        SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = @message_text;
    END IF;

END $$

DELIMITER ;
```

Punto della situazione rispetto alla pianificazione

Un po' in ritardo rispetto alla pianificazione.

Programma di massima per la prossima giornata di lavoro

Continuare l'implementazione, continuare con la pagina di gestione dei clienti.

Diario di lavoro

Luogo	Canobbio, SAM Trevano
Data	21/11/2019

Lavori svolti

Sono andato avanti con la gestione dei clienti

Ho fatto in modo che venga salvato un utente quando viene schiacciato "Salva"

Clienti Fatture Prodotti Utenti LOGOUT

Clienti

Nuovo cliente

Cliente Aziendale

Nome Azienda
Nome Cliente
Cognome Cliente

Via
Numero Civico

Città
NAP

Telefono
Email

SALVA

Non è stato salvato nessun cliente

Quando viene aggiunto un cliente viene subito mostrato sotto nella lista.

Questo codi permette di inserire un nuovo cliente all'interno del database.

```
/**  
 * Insert a new client in the database, also it might be a company.  
 *  
 * @param clientId the new id of the client  
 * @param companyName the new name of the client, or the responsible of the company  
 * @param clientSurname the new surname of the client, or the responsible of the company  
 * @param street the new street of the client address  
 * @param houseNo the new client house_no of the client address  
 * @param city the new city of the client address  
 * @param nap the new nap of the client address  
 * @param telephone the new telephone of the client  
 * @param email the new email of the client  
 * @param companyName the new company name of the client, if the client is a company  
 */  
public function saveClient(  
$clientId,  
$companyName,  
$clientSurname,  
$street,  
$houseNo,
```

```
$city,
$nap,
$telephone,
$email,
$CompanyName = null
){
// prepare the query, to insert a new city in the database
$insertCity = "insert into city (id, name, nap) values (null, :city, :nap)";
$stmt = $this->connInvoices->prepare($insertCity);

// insert in the query, the data of the new city
$stmt->bindParam(':city', $city);
$stmt->bindParam(':nap', $nap);

// the query statement that insert the new city is executed
$stmt->execute();

// get the id of the new city
$selectNewCityId = "select id from city where nap = :nap";
$stmt = $this->connInvoices->prepare($selectNewCityId);
$stmt->bindParam(':nap', $nap);
$cityId = $stmt->execute();

// prepare the query, to insert a new client in the database
$insertClient = "
    insert into client (id, name, surname, street, house_no, telephone, email, city_id)
    values (null, :clientName, :clientSurname, :street, :house_no, :telephone, :email, :cityId)
";
$stmt = $this->connInvoices->prepare($insertClient);

// insert in the query, the data of the new client
$stmt->bindParam(':clientName', $clientName);
$stmt->bindParam(':clientSurname', $clientSurname);
$stmt->bindParam(':street', $street);
$stmt->bindParam(':house_no', $houseNo);
$stmt->bindParam(':telephone', $telephone);
$stmt->bindParam(':email', $email);
$stmt->bindParam(':cityId', $cityId);

// the query statement that insert the new client is executed
$stmt->execute();

if ($CompanyName) {
    //prepare the query, to insert a new company name in the database
    $insertCompanyName = "
        insert into client_company (id, name)
        values (null, :companyName)
    ";
    $stmt = $this->connInvoices->prepare($insertCompanyName);

    // insert in the query, the new company name
    $stmt->bindParam(':companyName', $CompanyName);

    // the query statement that insert the new company name is executed
    $stmt->execute();
}
}
```

Problemi riscontrati e soluzioni adottate

nessuno

Punto della situazione rispetto alla pianificazione

Nome Progetto: Gestione fatturazioni

2

Un po' in ritardo rispetto alla pianificazione.

Programma di massima per la prossima giornata di lavoro

Continuare l'implementazione, iniziare gestione fatturazioni.

Diario di lavoro

Luogo	Canobbio, SAM Trevano
Data	26/11/2019

Lavori svolti

Ho sistemato la pagina per la gestione degli utenti, compreso di grafica e ottimizzazione

Ho iniziato la gestione delle fatturazioni, e ho sistemato alcuni commenti in alcune classi e alcune view.

Questa funzione esegue una query che permette di leggere tutti i clienti salvati sul database, anche se alcune volte il nome dell'azienda può non esserci, e quindi il cliente non è associato a nessun nome di azienda.

Questa query ritornerà tutti i clienti, ma se un cliente non ha associato un nome di una azienda questo campo risulterà uguale a Null.

```
/*
 * Get the saved clients in the database.
 */
public function getClients()
{
    // prepare the query, that get the saved clients
    $selectClients = '
        SELECT cl.id, cl.name AS clientName, cl.surname AS clientSurname, cl.street, cl.house_no AS
        houseNo, ci.name AS city, ci.nap, cl.telephone, cl.email,
        CASE
            WHEN EXISTS (
                SELECT co.name
                FROM client_company co
                WHERE co.client_id = cl.id
            )
            THEN (
                SELECT co.name
                FROM client_company co
                WHERE co.client_id = cl.id
            )
            ELSE null
        END AS companyName
        FROM client cl, city ci
        WHERE cl.city_id = ci.id;
    ';

    $stmt = $this->connInvoices->prepare($selectClients);

    // the query statement is executed and returned
    $stmt->execute();
    return $stmt->fetchAll(PDO::FETCH_ASSOC);
}
```

Problemi riscontrati e soluzioni adottate

nessuno

Punto della situazione rispetto alla pianificazione

Un po' in ritardo rispetto alla pianificazione.

Programma di massima per la prossima giornata di lavoro

Continuare l'implementazione, iniziare gestione fatturazioni.

Diario di lavoro

Luogo	Canobbio, SAM Trevano
Data	28/11/2019

Lavori svolti

Oggi ho sistemato dei metodi per validare e ne ho aggiunti altri, in modo che i dati rimangono in un formato più leggibile e professionale.

Ho anche sistemato la gestione dei clienti.

Ho sistemato la gestione degli utenti, dei prodotti e dei clienti.

Problemi riscontrati e soluzioni adottate

nessuno

Punto della situazione rispetto alla pianificazione

Un po' in ritardo rispetto alla pianificazione.

Programma di massima per la prossima giornata di lavoro

Continuare l'implementazione, iniziare gestione fatturazioni.

Diario di lavoro

Luogo	Canobbio, SAM Trevano
Data	29/11/2019

Lavori svolti

Ho anche sistemato la gestione dei Utenti.

Ho sistemato ulteriormente la gestione degli utenti, dei prodotti e dei clienti.

Problemi riscontrati e soluzioni adottate

nessuno

Punto della situazione rispetto alla pianificazione

Un po' in ritardo rispetto alla pianificazione.

Programma di massima per la prossima giornata di lavoro

Continuare l'implementazione, iniziare gestione fatturazioni.

Diario di lavoro

Luogo	Canobbio, SAM Trevano
Data	03/12/2019

Lavori svolti

Ho sviluppate alcune funzioni con Ajax per poter effettuare operazioni sul database e non ogni volta ricaricare la pagina.

Es:

Questa funzione permette di aggiornare il contenuto di un utente, cioè salvare il cambiamento sul database. In Questo modo operazione non passa per il controller ma passa su una pagina php apposita che si occupa di eseguire solo questa operazione.

```
function updateUser(id) {  
    var user = createUserArrayFromId(id);  
    console.log(user);  
    var jsonString = JSON.stringify(user);  
    console.log(jsonString);  
  
    $.ajax({  
        type: "POST",  
        url: "/Gestione-fatturazioni/src/invoices/app/models/updateUser.php",  
        data: jsonString,  
  
        success: function(response) {  
            console.log(response);  
            $(".user-" + id + "-field").prop("disabled", true);  
        }  
    });  
}
```

Sono andato avanti a sistemare il contenuto delle pagine users, products, e clients.

Problemi riscontrati e soluzioni adottate

nessuno

Punto della situazione rispetto alla pianificazione

Un po' in ritardo rispetto alla pianificazione.

Programma di massima per la prossima giornata di lavoro

Continuare l'implementazione, iniziare gestione fatturazioni e finire di sistemare le altre pagine.

Diario di lavoro

Luogo	Canobbio, SAM Trevano
Data	06/12/2019

Lavori svolti

Per ogni pagina ho aggiunto dei buttoni che possono eseguire le informazioni principali sui dati del database. Le azioni principali che si possono eseguire con i buttoni sono la modifica, il salvataggio e l'eliminazione dei dati, invece per aggiungere i dati è presente un'interfaccia apposita all'inizio di ogni pagina.

Come esempio prendo la pagina per la gestione degli utenti, in questa pagina è presente un'interfaccia per aggiungere un nuovo utente e subito sotto la tabella con la lista degli utenti memorizzati sul database.

Sopra alla tabella sono presenti i buttoni che permetto di salvare e modificare tutti insieme i dati presenti nel database. Invece in parte ad ogni riga sono presenti i buttoni per modificare, salvare e eliminare un utente.

Se si schiaccia il bottone per modificare e dopo lo si rischiaccia i dati modificati non saranno salvati e i dati torneranno allo stato in cui erano prima della modifica.

Se si schiaccia il bottone salva dopo la modifica i dati non si potranno più modificare fino a quando non si schiaccia di nuovo il bottone modifica.

Per precauzione quando il bottone elimina viene schiacciato comparirà una piccola interfaccia, in cui si chiede la conferma per cancellare l'utente.

Utenti

Nuovo Utente

Nome Utente	Email
<input type="text"/>	<input type="text"/>
Password	Conferma Password
<input type="password"/>	<input type="password"/>
Aggiungi	

Lista Utenti

Salva Tutto
 Modifica Tutto

Nome Utente	Email	Abilitato	
Sherryhaibara	sherryhaibara@gmail.com	<input type="radio"/>	
RobertoMand	ro.manda@gmail.com	<input checked="" type="checkbox"/>	

Queste azioni che vengono fatte sui dati sono state implementate in modo tale che quando vengono eseguite non si debba ricaricare la pagina, questo è stato possibile con l'utilizzo di AJAX per eseguire le azioni.

Le funzioni con **AJAX** sono state fatte in un file esterno apposito per ogni pagina. E in questo esempio permette di salvare le informazioni di un utente.

Questa funzione prende le informazioni che sono presenti all'interno della pagina, e li manda sotto forma di **JSON** ad un metodo presente nel controller della stessa view con un **POST**, in questo caso **updateUser** del controller **users.php**

Quando il metodo del controller è stato eseguito e arriva una risposta viene disabilitata la possibilità di modifica.

```
let URL = "<?= URL ?>";

/**
 * Update a user from its ID
 *
 * @param id The id of the user
 */
function updateUser(id) {
  var user = createUserArrayFromId(id);
  var jsonUser = JSON.stringify(user);

  $.ajax({
    type: "POST",
    url: URL + "users/updateUser",
    data: {
      user: jsonUser
    },
    success: function(response) {
      if (response) {
        $(".user-" + id + "-field").prop("disabled", true);
      }
    }
  });
}
```

Questo metodo aiuta il metodo precedente, permette di ricavare le informazioni di un certo utente dal suo ID, e le ritorna sotto forma di file JSON.

```
/**
 * Get the user representation with an JSON,
 * the JSON is mapped name => value.
 *
 * @param id The id of the user
 */
function createUserArrayFromId(id) {
  var user = {};

  $(".user-" + id + "-field").each(function(i, obj) {
    user[this.name] = this.value;
  });

  user["enabled"] = $("#enabled" + id).is(":checked");

  return user;
}
```

All'interno del metodo del controller che è stato richiamato dalla funzione AJAX, è presente il codice che permette di salvare i dati modificati all'interno del database. Riceve le informazioni tramite POST e subito dopo le salva utilizzando il metodo **updateUser()** all'interno della classe model principale del controller, in questo caso **UserModel**. Quello che ritorna questa funzione non viene letto ma serve alla funzione AJAX per sapere quando è stato eseguito il metodo.

```
/**  
 * Update a user,  
 * with the informations contained in the upcoming POST request.  
 *  
 * @return void  
 */  
public function updateUser()  
{  
    // the json containing the informations of the new user  
    $user = json_decode($_POST['user'], true);  
  
    $this->model('UserModel');  
    $UserModel = new UserModel();  
    $UserModel->updateUser($user['ids[]'], $user['usernames[]'], $user['emails[]'], $user['enabled']);  
  
    echo "true";  
}
```

Problemi riscontrati e soluzioni adottate

nessuno

Punto della situazione rispetto alla pianificazione

Un po' in ritardo rispetto alla pianificazione.

Programma di massima per la prossima giornata di lavoro

Continuare l'implementazione, iniziare gestione fatturazioni.

Diario di lavoro

Luogo	Canobbio, SAM Trevano
Data	10/12/2019

Lavori svolti

Oggi ho sistemato il codice e i commenti di un po' di classi.

Ho inserito una nuova costante all'interno della classe controller, cioè quella che estendono tutti i controller, questa costante contiene il nome del controller.

```
/**  
 * @var string The controller name of this controller class.  
 */  
protected $controllerName;  
  
/**  
 * Controller empty constructor.  
 */  
public function __construct()  
{  
    $this->controllerName = get_class($this);  
}  
/**  
 * @var string The controller name of this controller class.  
 */  
protected $controllerName;  
  
/**  
 * Controller empty constructor.  
 */  
public function __construct()  
{  
    $this->controllerName = get_class($this);  
}
```

Problemi riscontrati e soluzioni adottate

nessuno

Punto della situazione rispetto alla pianificazione

Un po' in ritardo rispetto alla pianificazione.

Programma di massima per la prossima giornata di lavoro

Continuare l'implementazione, finire l'implementazione.

Diario di lavoro

Luogo	Canobbio, SAM Trevano
Data	12/12/2019

Lavori svolti

Ho sistemato le varie view presenti sul sito al nuovo stile usando ancora bootstrap, ma utilizzando anche un'altra libreria esterna per dare lo stile alla grafica:

link: <http://desigmodo.github.io/Flat-UI/>

Ho aggiunto anche delle notifiche quando è stata completata un'azione con successo, per fare questo ho utilizzato una libreria:

link: <https://notifyjs.jpillora.com/>

Quando si vuole eliminare un record dal database comparirà una conferma per procedere con l'eliminazione, questo l'ho fatto utilizzando una libreria:

Link: <https://craftpip.github.io/jquery-confirm/>

Problemi riscontrati e soluzioni adottate

nessuno

Punto della situazione rispetto alla pianificazione

Un po' in ritardo rispetto alla pianificazione.

Programma di massima per la prossima giornata di lavoro

Continuare l'implementazione, finire l'implementazione.

Diario di lavoro

Luogo	Canobbio, SAM Trevano
Data	13/12/2019

Lavori svolti

Ho sistemato il codice per la validazione e ogginti validazioni mancanti.
Ho sistemato il codice in diverse classi e sono andato avanti con l'implementazione.

Problemi riscontrati e soluzioni adottate

nessuno

Punto della situazione rispetto alla pianificazione

Un po' in ritardo rispetto alla pianificazione.

Programma di massima per la prossima giornata di lavoro

Continuare l'implementazione, finire l'implementazione.

Diario di lavoro

Luogo	Canobbio, SAM Trevano
Data	17/12/2019

Lavori svolti

Ho sistemato la documentazione e sono andato avanti con essa ho fatto il test e l'abstract.
Ho sistemato il codice e sono andato avanti con l'implementazione.

Problemi riscontrati e soluzioni adottate

nessuno

Punto della situazione rispetto alla pianificazione

Un po' in ritardo rispetto alla pianificazione.

Programma di massima per la prossima giornata di lavoro

Continuare l'implementazione, finire l'implementazione e mostrare demo.

Diario di lavoro

Luogo	Canobbio, SAM Trevano
Data	19/12/2019

Lavori svolti

Ho sistemato gli utilimi problemi sul sito web.
Ho mostrato la demo al mio formatore.

Problemi riscontrati e soluzioni adottate

nessuno

Punto della situazione rispetto alla pianificazione

In ritardo

Programma di massima per la prossima giornata di lavoro

Cosegna progetto

Diario di lavoro

Luogo	Canobbio, SAM Trevano
Data	20/12/2019

Lavori svolti

Ho finito la documentazione e ho consegnato il progetto.

Problemi riscontrati e soluzioni adottate

-

Punto della situazione rispetto alla pianificazione

-

Programma di massima per la prossima giornata di lavoro

-