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 e 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 impedische 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 la 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 diponibili 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 acesso.

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

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

Il contenuto di questi metodi sono rispettivamente:

```
/**
 * Redirect to the home page, if the login is not been effectuated by anyone.
 */
public function redirectToHomePageIfAnyoneIsLogged()
{
	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 redirectToUserDefaultPermittedPageIfUserIsLogged()
{
	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.