1 Practice 6.1. Databases

> This practice is the next in the series of practices consisting of a Contacts application. This practice is based on the previous one. Before start it, finish the previous one, make the corrections you consider, if needed, and start coding.
>
> The inclusion of comments in the scripts will be valued.
>
> Before start, create a new git branch named `contacts-u6` and change to it:
>
> ```
> git branch contacts-u6
> git checkout contacts-u6
> ```
>
> Do all the practice in the contacts-u6 branch. Once finished, do the submission via GitHub Classroom as detailed in the end of the practice.

1.1 Exercise 1

Make a MySql database named `contacts` with an owner user named `contacts` and a table with the same name. The user must have all the permissions in the database. The table must have **exactly** the next schema:

- **id**: INT, auto-increment, primary key
- **title**: VARCHAR(5), default value “Mr.”
- **name**: VARCHAR(100), not null
- **surname**: VARCHAR(100), not null
- **birthDate**: DATE, not null
- **phone**: VARCHAR(15), not null
- **email**: VARCHAR(50), not null
- **favourite**: BOOLEAN, default value 0 (false)
- **important**: BOOLEAN, default value 0 (false)
- **archived**: BOOLEAN, default value 0 (false)

The output of the `describe contacts;` command should be:

```
mysql> describe contacts;
+-----------+--------------+------+-----+---------+----------------+
| Field     | Type         | Null | Key | Default | Extra          |
+-----------+--------------+------+-----+---------+----------------+
| id        | int          | NO   | PRI | NULL    | auto_increment |
| title     | varchar(5)   | NO   |     | Mr.     |                |
```

```
| name      | varchar(100) | NO   |     | NULL    |                |
| surname   | varchar(100) | NO   |     | NULL    |                |
| birthDate | date         | NO   |     | NULL    |                |
| phone     | varchar(15)  | NO   |     | NULL    |                |
| email     | varchar(50)  | NO   |     | NULL    |                |
| favourite | tinyint(1)   | NO   |     | 0       |                |
| important | tinyint(1)   | NO   |     | 0       |                |
| archived  | tinyint(1)   | NO   |     | 0       |                |
+-----------+--------------+------+-----+---------+----------------+
```

You can use the provided `contacts-dump.sql` file to create the database and insert some sample data, but you need to add the owner user.

1.2 Exercise 2

Make an interface, named **`IDbAccess.php`**, for your contacts. The class must have the next **public static** methods:

- getAll()
- select($id)
- insert($object)
- delete($object)
- update($object)

1.3 Exercise 3

Make a class named DB`Connection.php` with a public static method used to connect to the database. It will have all the data needed to connect to the database. Use persistent connections.

1.4 Exercise 4

Make a class named **`ContactRepository`**. This class must implement the `IDbAccess` interface and perform all the database operations to select, insert, update and delete contact entries, using **PDO** and **objects**.

1.5 Exercise 5

Modify your `contact_list.php` and `contact_form.php` scripts so that they store and retrieve the data from the database.

> In this practice we don't need the `data.php`, `functions.php` and `checkdata.php` files. Delete them and modify your scripts to not use them.

1.6 How to submit to GitHub Classroom

1. This task must be submitted to the same repository of the Practice 4.1. Remember to make a new branch before starting the practice.
2. Once you finish the task, make a commit with the comment "PRACTICE 6.1 SUBMISSION COMMIT", merge the branch into the main branch, and push it to GitHub. For example:

```
git commit -m "PRACTICE 6.1 SUBMISSION COMMIT"
git checkout main
git merge contacts-u6
git push
```

3. Before that, you can do the commits and pushes you want. If you change your code after your submission commit, make another commit and push with the same text in the message adding the corrections you've done. Make a pull-request if you want so i could know your code is submitted.
4. Once pushed to GitHub, create a tag with the name "Practice 6". Create a new tag if you make corrections after the first submission (ex. "Practice 6 corrected").

If you have any doubt in your task, you can push your code and ask me by email what's your problem.