**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**Чорноморський національний університет   
імені Петра Могили**

**Факультет комп’ютерних наук**

**Кафедра інтелектуальних інформаційних систем**

**ЗВІТ**

*з лабораторної роботи № 1*

Дисципліна «П ІС та ТЗІ»

Спеціальність 122 «Комп'ютерні науки»

122 – ЛР.ПЗ.01 – 402.1610203

Виконала: студентка групи 402

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ К. О. Бондаренко\_\_\_\_

(підпис, ініціали та прізвище)

\_\_\_\_\_\_\_\_\_\_\_\_ 16.09.2019\_\_\_\_\_\_\_

(дата)

Перевірив:

\_\_\_\_\_\_\_\_\_О. Р. Тогоєв \_\_\_\_\_

(підпис, ініціали та прізвище)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_16.09.2019\_\_\_\_\_\_\_\_\_\_

(дата)

**Миколаїв – 2019**

**Лабораторна робота №1**

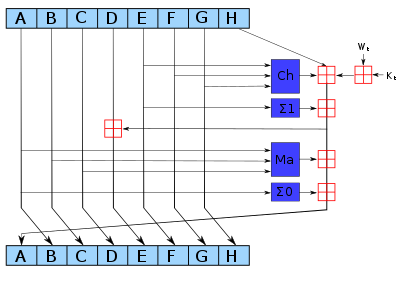
**Завдання для студентів:**

1. Розробити форму реєстрації з записом у БД за допомогою JS/PHP.
2. Зашифрувати дані форми SHA-256.
3. Оформити звіт.

**Теоретичні відомості:**

**SHA-256**

SHA-256 являє собою односпрямовану функцію для створення цифрових відбитків фіксованої довжини (256 біт, 32 байт) з вхідних даних розміром до 2,31 ексабайт (2⁶⁴ біт) і є окремим випадком алгоритму з сімейства криптографічних алгоритмів SHA-2 (Secure Hash Algorithm Version 2) опублікованим АНБ США в 2002 році.  
Хеш-функції сімейства SHA-2 побудовані на основі структури Меркле - Дамгарда.  
Оригінал тексту після доповнення розбивається на блоки, кожен блок - на 16 слів. Алгоритм пропускає кожен блок повідомлення через цикл з 64 ітераціями. На кожній ітерації 2 слова перетворюються, функцію перетворення задають інші слова. Результати обробки кожного блоку складаються, сума є значенням хеш-функції. Так як ініціалізація внутрішнього стану проводиться результатом обробки попереднього блоку, то немає можливості обробляти блоки паралельно. Графічне представлення однієї ітерації обробки блоку даних:



На поточний момент відомі методи для конструювання колізій до 31 ітерації. З огляду на алгоритмічної схожості SHA-2 з SHA-1 і наявності в останньої потенційних вразливостей прийнято рішення, що SHA-3 буде базуватися на зовсім іншому алгоритмі. 2 жовтня 2012 року NIST затвердив в якості SHA-3 алгоритм Keccak.

**Хід виконання роботи**

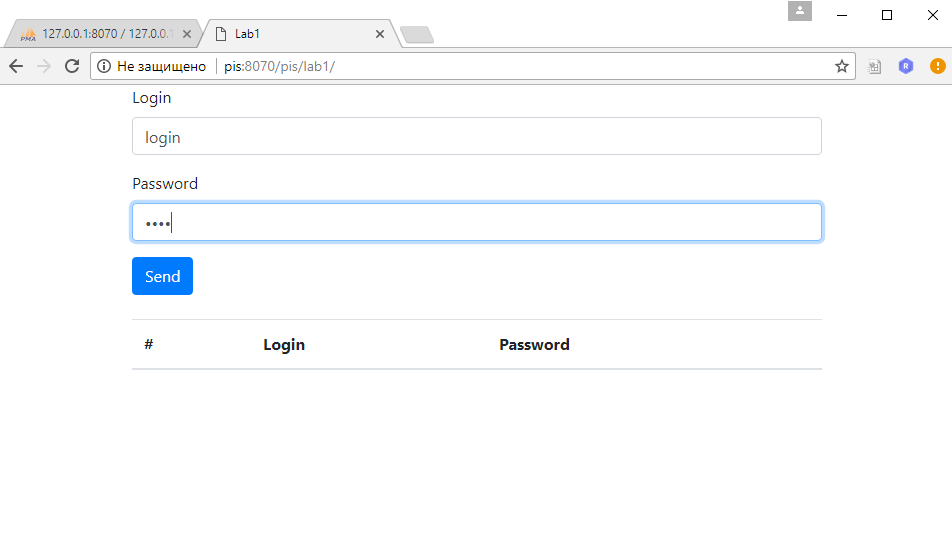
1. На головній сторінці потрібно ввести логін, пароль та натиснути кнопку Send (рис.1).

Рисунок 1 – Головна сторінка.

1. З’явиться в таблиці нижче логін та зашифрований пароль(рис.2).

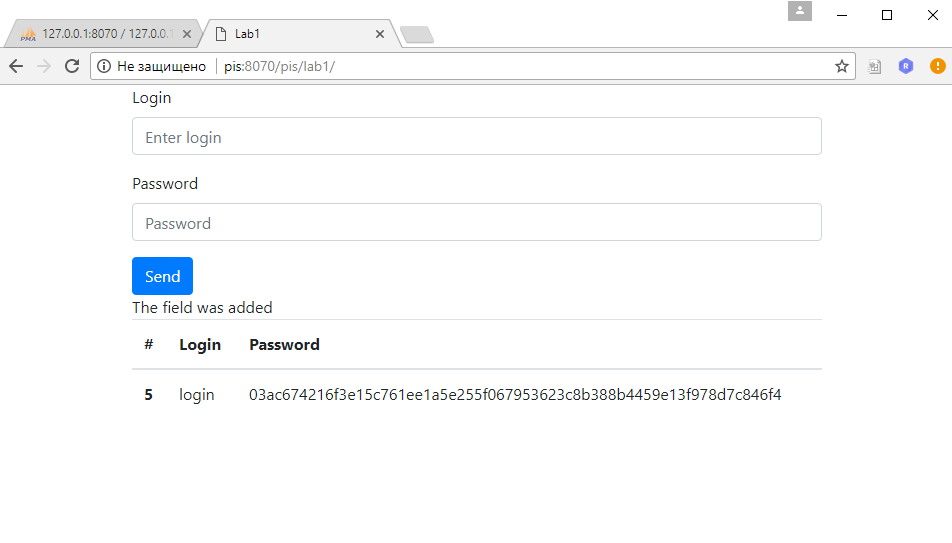


Рисунок 2 – Таблиця з користувачами.

**Реалізація**

index.php

**<?php  
  
include\_once 'config.php'**;  
  
**include\_once 'lib/MySql.php'**;  
**include\_once 'lib/Sql.php'**;  
  
$mySql = **new** MySql();  
  
**if**(**isset**($\_POST[**'btn'**]) && $\_POST[**'login'**] && $\_POST[**'password'**])  
{  
 $insert = $mySql->setTable(**'users'**)->setValues($\_POST[**'login'**])->setValues($\_POST[**'password'**])->insert();  
}  
  
$users = $mySql->setTable(**'users'**)->select();  
  
  
**include\_once 'templates/index.php'**;

config.php

**<?php**define(**'HOST'**,**'localhost'**);  
define(**'PORT'**, **'3307'**);  
define(**'USER\_NAME'**,**'root'**);  
define(**'USER\_PASS'**,**''**);  
define(**'DATABASE'**,**'pis'**);  
define(**'TABLE'**,**'users'**);  
**?>**

templates/index.php

**<?php  
?>**<!doctype html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport"  
 content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">  
 <meta http-equiv="X-UA-Compatible" content="ie=edge">  
 <title>Lab1</title>  
 <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"  
 integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T" crossorigin="anonymous">  
 <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"  
 integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM"  
 crossorigin="anonymous"></script>  
</head>  
<body>  
<div class="container">  
 <form method="post">  
 <div class="form-group">  
 <label for="login">Login</label>  
 <input type="text" class="form-control" id="login" name="login" placeholder="Enter login">  
 </div>  
 <div class="form-group">  
 <label for="password">Password</label>  
 <input type="password" class="form-control" name="password" id="password" placeholder="Password">  
 </div>  
 <button type="submit" name="btn" class="btn btn-primary">Send</button>  
 </form>  
 **<?php  
 echo** $insert.**'<br>'**;  
 **?>** <table class="table">  
 <thead>  
 <tr>  
 <th scope="col">#</th>  
 <th scope="col">Login</th>  
 <th scope="col">Password</th>  
 </tr>  
 </thead>  
 <tbody>  
 **<? foreach**($users **as** $user){**?>** <tr>  
 <th scope="row">**<?=**$user[**'id'**]**?>**</th>  
 <td>**<?=**$user[**'login'**]**?>**</td>  
 <td>**<?=**$user[**'password'**]**?>**</td>  
 </tr>  
 **<?**}**?>** </tbody>  
 </table>  
</div>  
  
</body>  
</html>

lib/Sql.php

**<?php  
  
class** Sql  
{  
 **protected** $fields;  
 **protected** $table;  
 **protected** $values;  
 **protected** $querySelect;  
 **protected** $queryInsert;  
  
 **function** \_\_construct()  
 {  
 $this->fields = **array**();  
 $this->values = **array**();  
 $this->table = **""**;  
 }  
  
  
 **public function** setFields($field)  
 {  
 **if** ($field != **"\*"** && $field != **""**) {  
 array\_push($this->fields, $field);  
 **return** $this;  
 } **else** {  
 **return false**;  
 }  
 }  
  
 **public function** setValues($value)  
 {  
 **if** ($value != **""**) {  
 array\_push($this->values, $value);  
 **return** $this;  
 } **else** {  
 **return false**;  
 }  
 }  
  
 **public function** getValues()  
 {  
 **return** $this->values;  
 }  
  
 **public function** setTable($table)  
 {  
 $this->table = $table;  
 **return** $this;  
 }  
  
 **public function** getTable()  
 {  
 **return** $this->table;  
 }  
  
  
 **public function** select()  
 {  
 $this->querySelect = **"SELECT id, login, password FROM "** . $this->table;  
 }  
  
 **function** insert()  
 {  
 $this->queryInsert = **"INSERT"** . **" INTO "** . $this->table . **"(login, password) "** . **"VALUES (:login, :password);"**;  
 }  
  
}  
  
**?>**

lib/MySql.php

**<?php  
include\_once 'Sql.php'**;  
**include\_once 'config.php'**;  
  
**class** MySql **extends** Sql  
{  
 **protected** $connection;  
 **protected** $obj;  
  
 **function** \_\_construct()  
 {  
 **parent**::*\_\_construct*();  
 $this->connection = **new** PDO(**"mysql:host="** . *HOST* . **";port="** . *PORT* . **";dbname="** . *DATABASE* /\*. ";charset=utf8\_unicode\_ci",\*/, *USER\_NAME*, *USER\_PASS*);  
 $this->connection->setAttribute(PDO::*ATTR\_ERRMODE*, PDO::*ERRMODE\_EXCEPTION*);  
 }  
  
 **public function** select()  
 {  
 **parent**::*select*();  
 $result = **array**();  
 $select = $this->connection->prepare($this->querySelect);  
  
 $select->execute();  
 $index = 0;  
 **while** ($row = $select->fetch(PDO::*FETCH\_ASSOC*)) {  
 $result[$index] = $row;  
 $index++;  
 }  
 **return** $result;  
 }  
  
 **function** insert()  
 {  
 **parent**::*insert*();  
 $result = **""**;  
 $insert = $this->connection->prepare($this->queryInsert);  
 $insert->bindParam(**':login'**, $this->values[0]);  
 $insert->bindParam(**':password'**, hash(**'sha256'**, $this->values[1]));  
 **if** ($insert->execute()) {  
 $result = **"The field was added"**;  
 } **else** {  
 $result = **"The field was NOT added"**;  
 }  
 **return** $result;  
 }  
  
}  
  
**?>**

**Висновки**: під час даної лабораторної роботи за допомогою PHP було розроблено сторінку з полями вводу логіна та пароля, що перетворює пароль у хеш за допомогою алгоритму SHA-256. Були закрыплены навички роботи з OpenServer, PhpMyAdmin та PhpStorm.