Министерство науки и высшего образования Российской Федерации
Севастопольский государственный университет
Кафедра ИС

Отчет
по лабораторной работе №9
«Исследование возможностей хранения данных на стороне сервера. Работа с файлами. Работа с реляционными СУБД.»
по дисциплине
«ВЕБ-ТЕХНОЛОГИИ»

Выполнил студент группы ИС/б-17-2-о
Горбенко К. Н.
Проверил
Овчинников А.Л.

Севастополь
2020

1 ЦЕЛЬ РАБОТЫ

Изучить возможности хранения данных на стороне сервера: работу с файлами и СУБД MySQL из PHP, приобрести практические навыки организации хранения данных на стороне сервера в файлах, в базах данных MySQL, а также овладение навыками постраничного вывода данных.

2 ЗАДАНИЕ НА РАБОТУ

1. Разработать базовый класс BaseActiveRecord для работы с базой данных, который реализует паттерн ActiveRecord (данный класс разместить в папке /my_site/app/core).

2. Для всех моделей, которые будут использоваться при выполнении данной лабораторной работы, создать классы, наследующие BaseActiveRecord. Для каждого из классов определить все поля и названия таблиц.

3. Создать новую страницу "Гостевая книга". Страница должна содержать форму ввода (Фамилия, Имя, Отчество, E –mail, Текст отзыва), а также таблицу сообщений, оставленных пользователями.

4. Реализовать страницу "Загрузка сообщений гостевой книги содержащую форму загрузки подготовленного заранее файла messages.inc на сервер.

5. Реализовать на странице "Тест по дисциплине"сохранение ответов пользователей и правильности ответов в разработанную таблицу базы данных MySQL, с возможностью просмотра сохраненных данных.

6. Разработать страницу «Редактор Блога», позволяющую добавлять записи Блога. Страница должна содержать форму добавления записи Блога и список выдаваемых постранично записей отсортированных в порядке убывания даты.

7. Разработать страницу «Мой Блог», содержащую упорядоченные в порядке убывания даты добавления, выдаваемые постранично данные.

8. Реализовать возможность добавления записей на страницу «Мой Блог» из файла формата CSV.

3 ХОД РАБОТЫ

3.1 Разработка класса BaseActiveRecord

```
1 abstract class BaseActiveRecord {
2     protected static $pdo;
3     private static $tablename;
4     private static $className;
5     private static $dbfields = array();
6 
7     protected $id;
8 
9     public function __construct() {
10         static::setupConnection();
11     }
12 
13     private static function setupConnection() {
14         static::$pdo = PortfolioPdo::getInstance();
15     }
16 
17     public static function find($id) {
18         static::setupConnection();
19 
20         $sql = "SELECT * FROM " . static::$tablename . " WHERE id=$id";
21         $stmt = static::$pdo->prepare($sql);
22         $stmt->setFetchMode(PDO::FETCH_CLASS, static::$className);
23         $stmt->execute();
24         $result = $stmt->fetch();
25 
26         if (!$result) {
27             throw new Exception("Entity " . static::$className . " was not
                   found.");
28         }
29 
30         return $result;
31     }
32 
33     public static function findAll() {
34         static::setupConnection();
35 
36         $sql = "SELECT * FROM " . static::$tablename;
37         $stmt = static::$pdo->prepare($sql);
38         $stmt->setFetchMode(PDO::FETCH_CLASS, static::$className);
39         $stmt->execute();
40 
41         return $stmt->fetchAll();
42     }
43 
```

```
44      public static function findByPage($offset, $rowsPerPage) {
45          static::setupConnection();
46
47          $sql = "SELECT * FROM " . static::$tablename . " ORDER BY createdAt
                DESC LIMIT " . "$offset , $rowsPerPage";
48          $stmt = static::$pdo->prepare($sql);
49          $stmt->setFetchMode(PDO::FETCH_CLASS, static::$className);
50          $stmt->execute();
51
52          return $stmt->fetchAll();
53      }
54
55      public static function getCount() {
56          static::setupConnection();
57
58          $sql = "SELECT COUNT(*) FROM " . static::$tablename;
59          $stmt = static::$pdo->query($sql);
60          $result = $stmt->fetch(PDO::FETCH_ASSOC);
61
62          return current($result);
63      }
64
65      private static function getFieldTypes() {
66          $stmt = static::$pdo->query("SHOW FIELDS FROM " . static::$tablename);
67          $fieldTypesTableRows = $stmt->fetchAll(PDO::FETCH_ASSOC);
68
69          $fieldNameToType = array();
70          foreach ($fieldTypesTableRows as $row) {
71              $fieldNameToType[$row["Field"]] = $row["Type"];
72          }
73
74          return $fieldNameToType;
75      }
76
77      private function wrapWithQuotesIfNeeded($fieldValue, $fieldType) {
78          if (substr($fieldType, 0, 7) == "varchar" || substr($fieldType, 0, 8)
                == "datetime") {
79              return "'" . $fieldValue . "'";
80          }
81          return $fieldValue;
82      }
83
84      public function save() {
85          $data = array();
86          $fieldTypes = static::getFieldTypes();
87
88          foreach (static::$dbfields as $fieldName) {
89              $data[$fieldName] = $this->wrapWithQuotesIfNeeded($this->{
```

```
                $fieldName}, $fieldTypes[$fieldName]);
90          }
91
92          return $this->saveInternal($data);
93      }
94
95      private function saveInternal($data) {
96          $values = implode(",", $data);
97          $fields = implode(",", static::$dbfields);
98
99          $stmt = static::$pdo->prepare("INSERT INTO " . static::$tablename .  "
                ($fields) VALUES ($values)");
100         $stmt->execute();
101
102         return static::$pdo->lastInsertId();
103     }
104
105     public function delete() {
106         $sql = "DELETE FROM " . static::$tablename . " WHERE id = " .$this->id;
107         $stmt = static::$pdo->query($sql);
108
109         $stmt->execute();
110     }
111 }
```

3.2 Модели

Модель ответа:

```
1  class Answer extends BaseActiveRecord {
2      public static $tablename = "answers";
3      public static $className = "Answer";
4      public static $dbfields = array("createdAt", "question1Answer", "
           question2Answer", "fullTextAnswer", "studentFullName", "email");
5
6      public string $createdAt;
7      public int $question1Answer;
8      public int $question2Answer;
9      public string $fullTextAnswer;
10     public string $studentFullName;
11     public string $email;
12
13     public function __construct() {
14         parent::__construct();
15     }
16 }
```

Модель записи блога:

```
1 class BlogEntry extends BaseActiveRecord {
2     public static $tablename = "blog";
3     public static $className = "BlogEntry";
4     public static $dbfields = array("createdAt", "subject", "message", "
          photoName", "name");
5
6     public function __construct() {
7         parent::__construct();
8     }
9
10    public string $createdAt;
11    public string $subject;
12    public string $message;
13    public string $photoName;
14    public string $name;
15 }
```

3.3 Реализация гостевой книги

Класс GuestBookMessagesProvider:

```
1 class GuestBookMessagesProvider implements IGuestBookMessagesProvider {
2     private string $fileName;
3
4     public function __construct() {
5         $this->fileName = $_SERVER["DOCUMENT_ROOT"] . "/messages.inc";
6     }
7
8     public function saveEntry(GuestBookEntry $entry) {
9         $handle = fopen($this->fileName, "a") or die("Cannot open file $this->
              fileName");
10        fwrite($handle, $this->stringifyGuestBookEntry($entry));
11        fclose($handle);
12    }
13
14    public function getAllEntries() {
15        if (!file_exists($this->fileName)) {
16            $handle = fopen($this->fileName, 'w');
17            fclose($handle);
18        }
19        $fileContent = file_get_contents($this->fileName);
20        $stringifiedEntries = preg_split("/;/", $fileContent, null,
              PREG_SPLIT_NO_EMPTY);
21
22        return array_map([$this, "parseGuestBookEntry"], $stringifiedEntries);
23    }
24
25    private function stringifyGuestBookEntry(GuestBookEntry $entry) {
```

```
26          $normalizedName = $this->stripCommasAndSemicolons($entry->name);
27          $normalizedEmail = $this->stripCommasAndSemicolons($entry->email);
28          $normalizedMessage = $this->stripCommasAndSemicolons($entry->message);
29          $formattedDate = $entry->date->format("d.m.Y");
30
31          return "$normalizedName,$normalizedEmail,$normalizedMessage,
                $formattedDate;";
32      }
33
34      private function stripCommasAndSemicolons($line) {
35          return str_replace([",", ";"], [""], $line);
36      }
37
38      private function parseGuestBookEntry(string $stringifiedEntry):
            GuestBookEntry {
39          $entryProperties = explode(",", $stringifiedEntry);
40
41          return new GuestBookEntry($entryProperties[0], $entryProperties[1],
                $entryProperties[2], DateTime::createFromFormat("d.m.Y",
                $entryProperties[3]));
42      }
43
44      public function importGuestBook($tempFileName) {
45          $newFileContent = file_get_contents($tempFileName);
46          file_put_contents($this->fileName, $newFileContent);
47      }
48
49      public function verifyGuestBookContents(string $contents): bool {
50          return preg_match("/^([A-Za-z]+ [A-Za-z]+ [A-Za-z]+,[a-zA-Z0-9+_.-]+@[a
                -zA-Z0-9.-]+,.*,\d{1,2}\.\d{1,2}\.\d{4};)+$/", $contents) == 1;
51      }
52  }
```

Форма для импорта гостевой книги:

```
1  <form action="Import" method="post" enctype="multipart/form-data">
2      <ol>
3          <li>
4              <label for="file-input">Please select a valid file</label>
5              <input type="file" class="field-long" id="file-input" name="
                   GuestBook" accept=".inc" />
6          </li>
7          <li>
8              <input type="submit" value="Submit" />
9          </li>
10     </ol>
11 </form>
```

Валидация импорта:

```
1 public function validate() {
2     $fileName = $_FILES["GuestBook"]["tmp_name"];
3     $fileSize = $_FILES["GuestBook"]["size"];
4     $fileContent = file_get_contents($fileName);
5
6     $isFileNameProvided = !empty($fileName);
7     $isSizeWithinLimit = $fileSize <= $this->fileSizeLimit;
8     $isFileContentValid = $this->guestBookMessageProvider->
          verifyGuestBookContents($fileContent);
9     $isFileValid = $isFileNameProvided && $isSizeWithinLimit &&
          $isFileContentValid;
10
11    return new ValidationResult($isFileValid, $isFileValid ? array() : array(
          array(1 => "File content is not valid.")));
12 }
```

Результат представлен на рисунке 1.

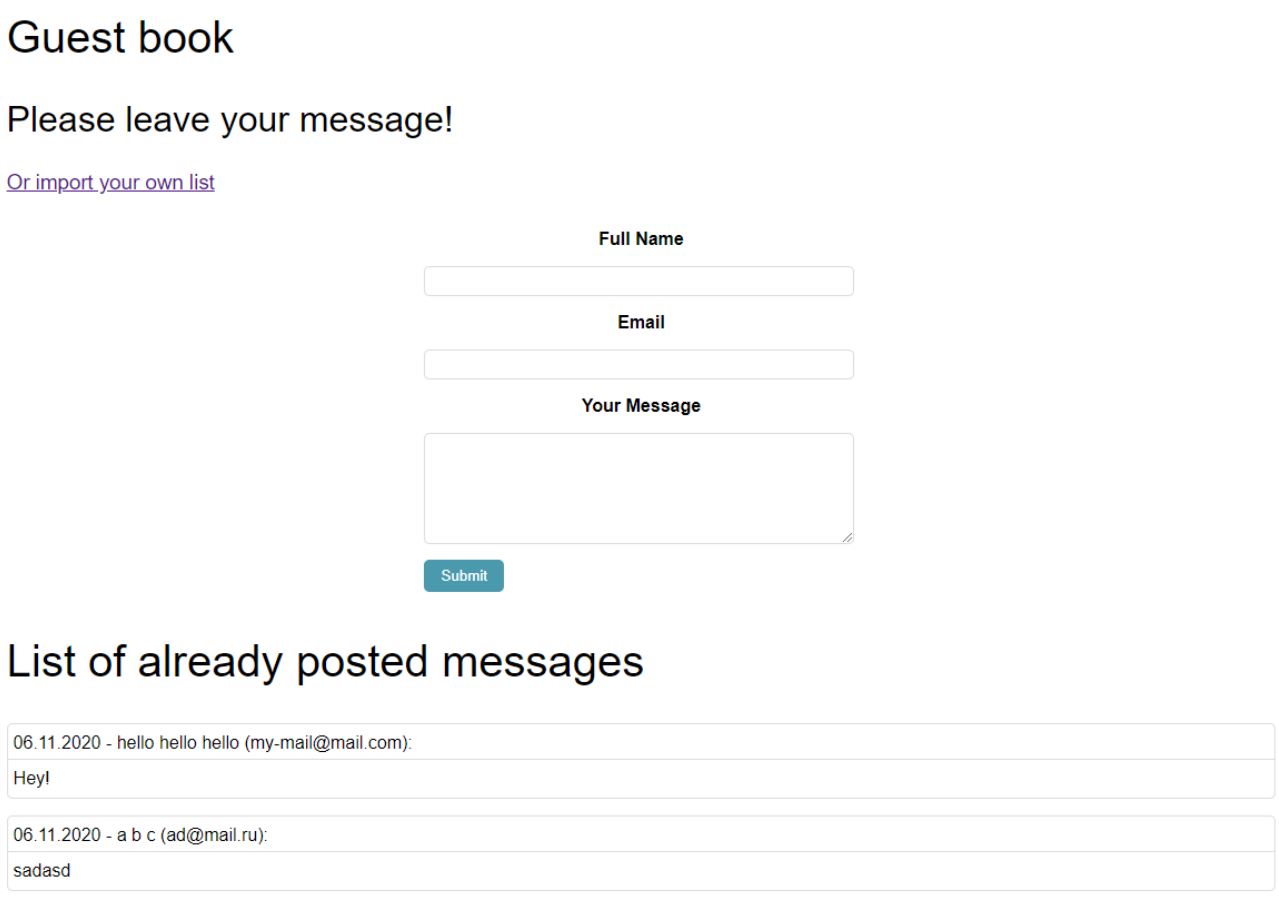

Рисунок 1 – Реализация гостевой книги

3.4 Сохранение ответов в базе данных

Изменения в контроллере:

```
class TestController {
    private $pageSize = 10;

    public function __construct(IContainer $container) { }

    public function index() {
        $viewModel = new ValidationViewModel(true, array());
        ViewRenderer::render("Views/Test/Index.php", "Test", $viewModel);
    }

    public function answers() {
        $page = isset($_GET['page'])
            ? $_GET['page']
            : 1;

        $offset = PaginationHelper::getOffset($page, $this->pageSize);
        $answers = Answer::findByPage($offset, $this->pageSize);

        $totalPages = PaginationHelper::getTotalPages(Answer::getCount(), $this
            ->pageSize);
        $paginationViewModel = new PaginationViewModel($page, $totalPages, "/
            Test/Answers");
        $viewModel = new AnswersViewModel($answers, $paginationViewModel);

        ViewRenderer::render("Views/Test/Answers.php", "Answers", $viewModel);
    }

    public function testPost() {
        $validator = new TestRequestValidator($_POST);
        $validationResult = $validator->validate();

        if ($validationResult->isValid) {
            $answer = new Answer();
            $answer->createdAt = date("Y-m-d H:i:s");
            $answer->studentFullName = $_POST["Name"];
            $answer->email = $_POST["Email"];
            $answer->question1Answer = $_POST["Question1"];
            $answer->question2Answer = $_POST["Question2"];
            $answer->fullTextAnswer = $_POST["Question3"];
            $answer->save();

            $testVerification = new TestRequestVerification($_POST);
            $viewModel = new SuccessfulTestViewModel($testVerification->verify
                ());
            ViewRenderer::render("Views/Test/Success.php", "Test", $viewModel);
            return;
        }

```

```
46          $viewModel = new ValidationViewModel($validationResult->isValid,
                $validationResult->errors);
47          ViewRenderer::render("Views/Test/Index.php", "Test", $viewModel);
48      }
49  }
```

Постраничный вывод данных:

```
 1  private static function renderLinkToCurrentPage(PaginationViewModel $model) {
 2      static::renderLinkToPage($model->currentPage, $model->baseUrl, false, true)
            ;
 3  }
 4
 5  private static function renderLinkToPreviousPage(PaginationViewModel $model) {
 6      static::renderLinkToPage($model->currentPage - 1,
 7                               $model->baseUrl,
 8                               $model->currentPage != 1,
 9                               true,
10                               "Previous");
11  }
12
13  private static function renderLinkToNextPage(PaginationViewModel $model) {
14      static::renderLinkToPage($model->currentPage + 1,
15                               $model->baseUrl,
16                               $model->currentPage < $model->totalPages,
17                               true,
18                               "Next");
19  }
20
21  public static function renderPagination(PaginationViewModel $model) {
22      static::renderLinkToPreviousPage($model);
23      static::renderLinkToPage($model->currentPage - 2,
24                               $model->baseUrl,
25                               $model->currentPage >= 3,
26                               $model->currentPage >= 3);
27      static::renderLinkToPage($model->currentPage - 1,
28                               $model->baseUrl,
29                               $model->currentPage >= 2,
30                               $model->currentPage >= 2);
31      static::renderLinkToCurrentPage($model);
32      static::renderLinkToPage($model->currentPage + 1,
33                               $model->baseUrl,
34                               $model->currentPage + 1 <= $model->totalPages,
35                               $model->currentPage + 1 <= $model->totalPages);
36      static::renderLinkToPage($model->currentPage + 2,
37                               $model->baseUrl,
38                               $model->currentPage + 2 <= $model->totalPages,
39                               $model->currentPage + 2 <= $model->totalPages);
40      static::renderLinkToNextPage($model);
```

```
41 }
```

Результат представлен на рисунке 2.

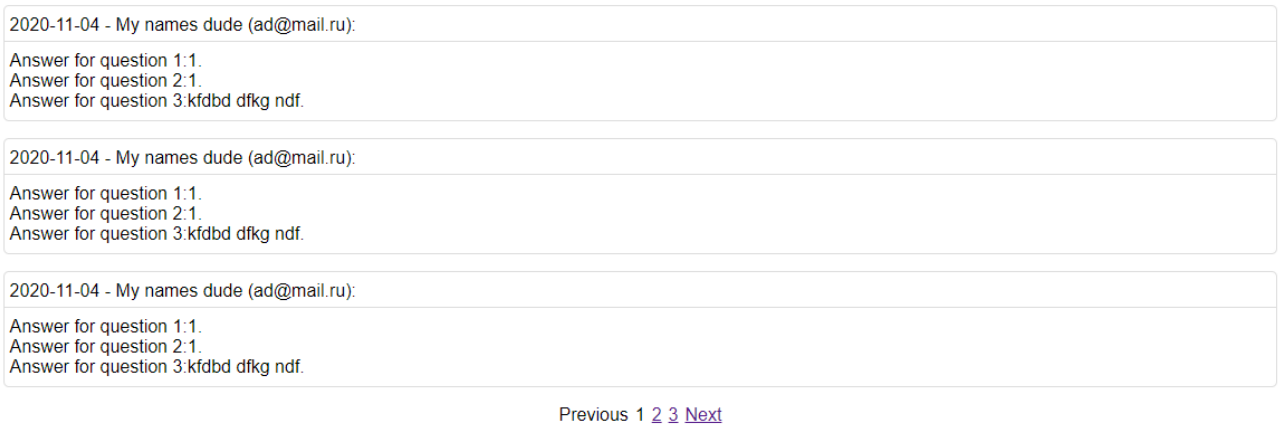

Рисунок 2 – Реализация сохранения ответов в БД

3.5 Реализация блога

Создание контроллера:

```
1 class BlogController {
2     private $pageSize = 10;
3
4     public function __construct(IContainer $container) { }
5
6     public function index() {
7         $page = isset($_GET['page'])
8             ? $_GET['page']
9             : 1;
10
11        $offset = PaginationHelper::getOffset($page, $this->pageSize);
12        $answers = BlogEntry::findByPage($offset, $this->pageSize);
13
14        $totalPages = PaginationHelper::getTotalPages(BlogEntry::getCount(),
              $this->pageSize);
15        $paginationViewModel = new PaginationViewModel($page, $totalPages, "/
              Blog/Index");
16        $viewModel = new BlogEntriesViewModel($answers, $paginationViewModel);
17
18        ViewRenderer::render("Views/Blog/Index.php", "Blog", $viewModel);
19    }
20
21    public function post() {
22        $viewModel = new ValidationViewModel(true, array());
23        ViewRenderer::render("Views/Blog/Post.php", "Post", $viewModel);
24    }
```

```
25
26      public function uploadPost() {
27          $validator = new BlogRequestValidator($_POST, $_FILES);
28          $validationResult = $validator->validate();
29
30          if ($validationResult->isValid) {
31              $picturePath = $_FILES["Picture"]["tmp_name"];
32              $pictureName = $_FILES["Picture"]["name"];
33              $picture = file_get_contents($picturePath);
34              file_put_contents($_SERVER["DOCUMENT_ROOT"] . "/client-side/images
                    /" . $pictureName, $picture);
35
36              $blogEntry = new BlogEntry();
37              $blogEntry->createdAt = date("Y-m-d H:i:s");
38              $blogEntry->name = $_POST["Name"];
39              $blogEntry->message = $_POST["Message"];
40              $blogEntry->photoName = $pictureName;
41              $blogEntry->subject = $_POST["Subject"];
42              $blogEntry->save();
43
44              $this->index();
45              return;
46          }
47
48          $viewModel = new ValidationViewModel($validationResult->isValid,
                $validationResult->errors);
49          ViewRenderer::render("Views/Blog/Post.php", "Post", $viewModel);
50      }
51  }
```

Результат представлен на рисунке 3.

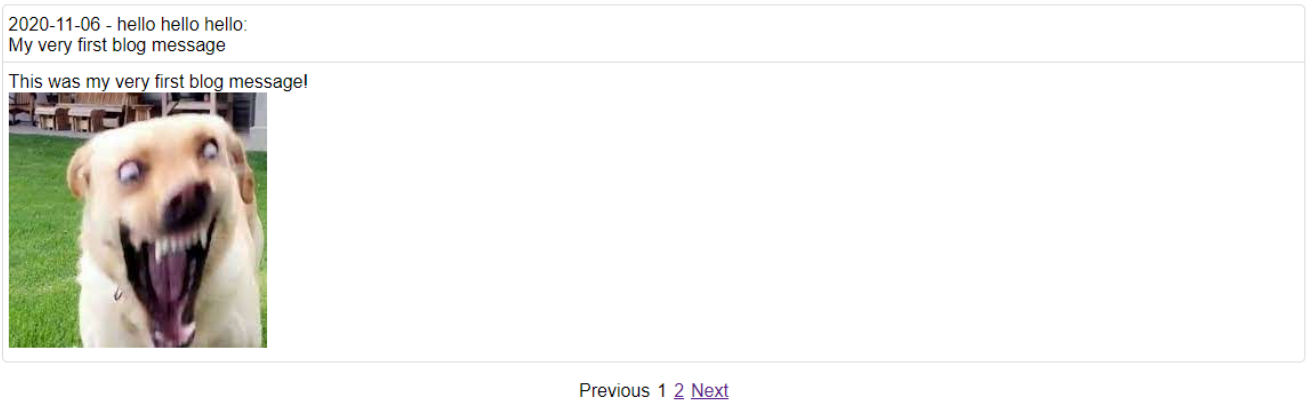

Рисунок 3 – Реализация блога

Форма для добавления записей в блог представлена на рисунке 4.

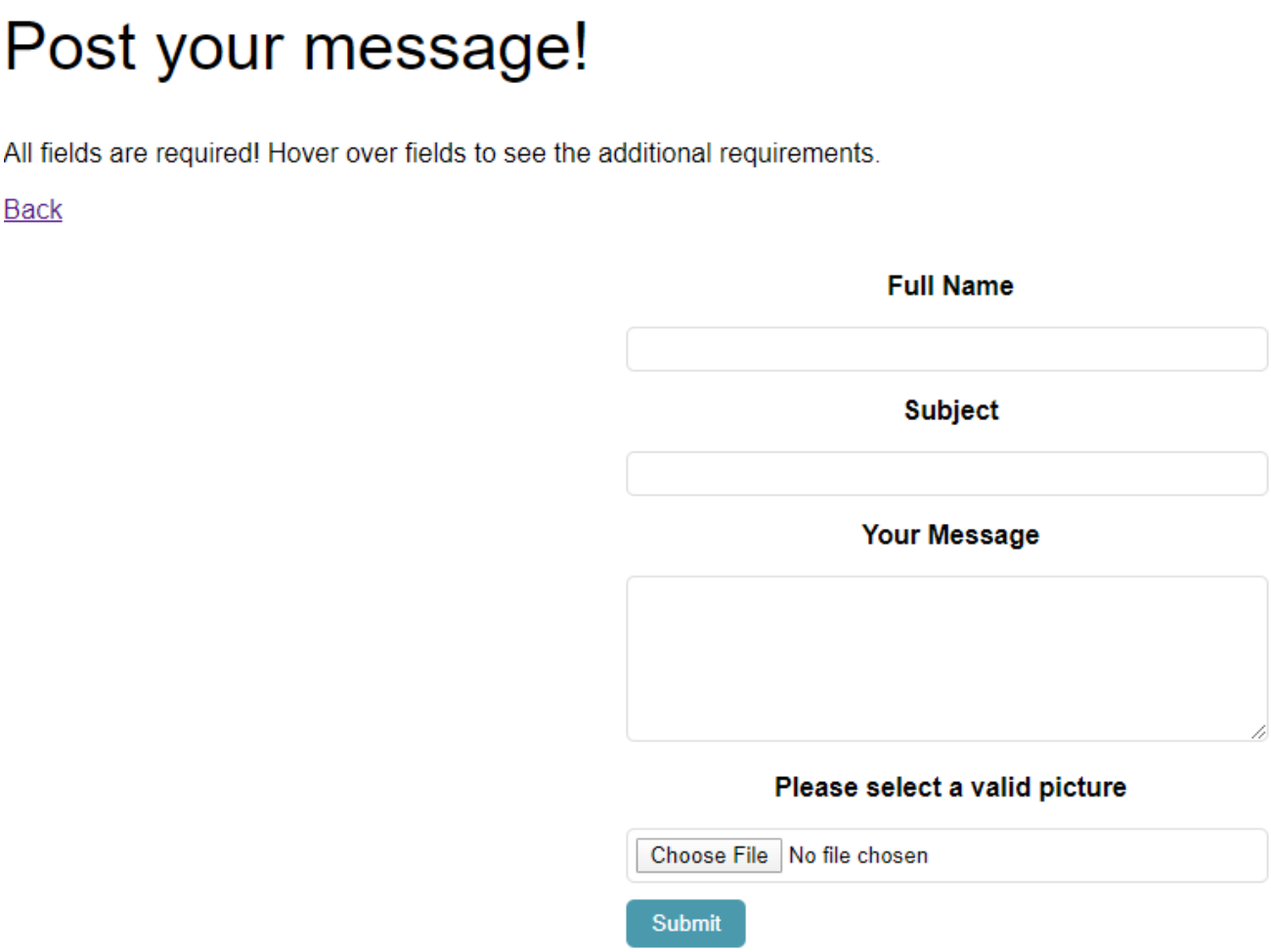

Рисунок 4 – Форма для добавления записей в блог

ВЫВОДЫ

В ходе лабораторной работы была изучена работа с файлами и базой данных в PHP. Для взаимодействия с базой данных используется интерфейс PDO. Соединение с PDO было энкапсулировано в объект ActiveRecord, который реализует все методы для работы с таблицами базы данных.