Правительство Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»
(НИУ ВШЭ)

Московский институт электроники и математики им. А.Н. Тихонова

ОТЧЕТ
О ЗАДАНИИ 9.1
по дисциплине «Проектный семинар»
Приложение на классах

Студент гр. БИБ203
Е. Д. Чевкин
«19» июня 2022 г.

Руководитель
Приглашенный преподаватель кафедры информационной безопасности киберфизических систем
ведущий программист
_____В.В. Башун
«___» июня 2022 г.

Москва 2022

Содержание

1 Структура приложения

На рисунке 1 изображена структура файловой системы приложения.

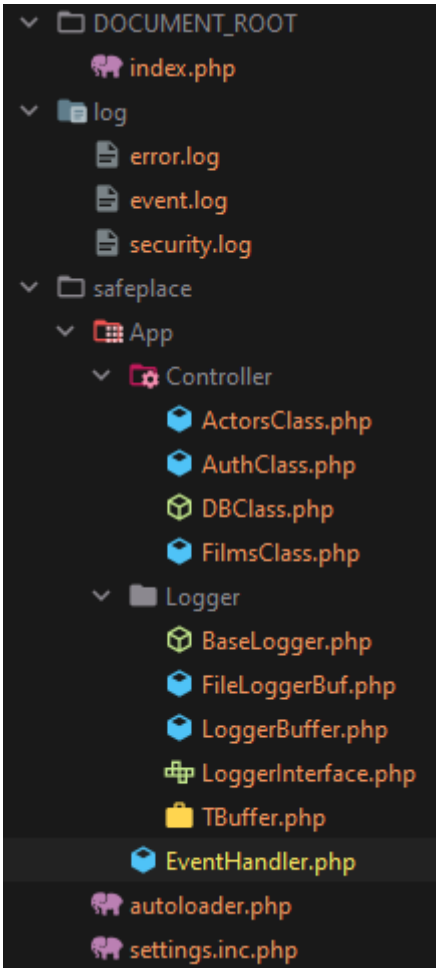

Рисунок 1 – файловая структура приложения

Структура пространства имен изображена на рисунке 2.

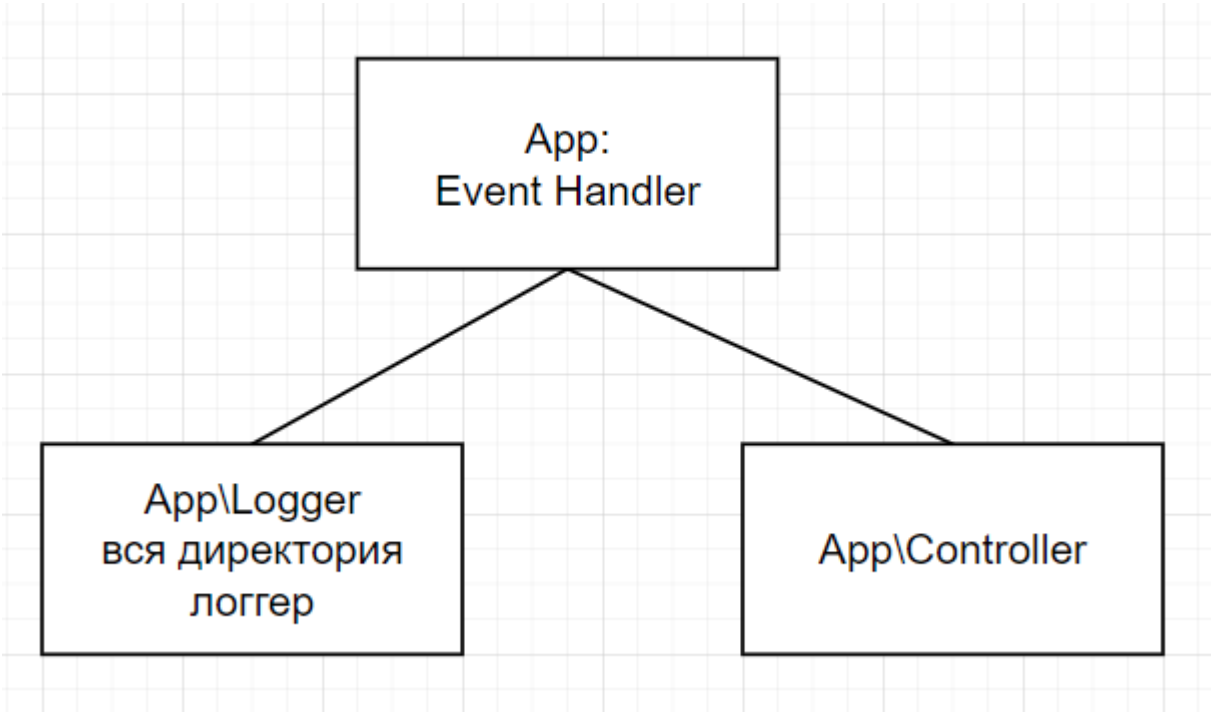

Рисунок 2 – структура пространства имен

2. Переписывание на классы

2.1 Классы сущностей

Для каждой основной сущности были написаны классы, реализующие по 5 методов: list, get, add, edit и delete. Классы сущностей наследуют родительский класс DBClass. На рисунках 3–5 изображены написанные классы.

```php
abstract class DBClass
{
    protected PDO $dbh;
    protected LoggerInterface $errorLogger;
    protected LoggerInterface $securityLogger;
    protected LoggerInterface $eventLogger;

    public function __construct(PDO $dbh, LoggerInterface $errorLogger,  LoggerInterface $securityLogger, LoggerInterface $eventLogger){...}

    protected function queryFetchAll($query, $queryParams){...}
}
```

Рисунок 3 – родительский класс DBClass

```php
class ActorsClass extends DBClass
{
    public function list(){...}

    public function get(){...}

    public function add(){...}

    public function edit(){...}

    public function delete(){...}
}
```

Рисунок 4 – класс ActorsClass для сущности actors

```php
class FilmsClass extends DBClass
{
    public function list(){...}

    public function get(){...}

    public function add(){...}

    public function edit(){...}

    public function delete(){...}
}
```

Рисунок 5 – класс FilmsClass

2.2 Класс авторизации

Для запросов авторизации и выхода написан отдельный класс AuthClass, который тоже наследует от DBClass. Этот класс и его методы изображены на рисунке 6.

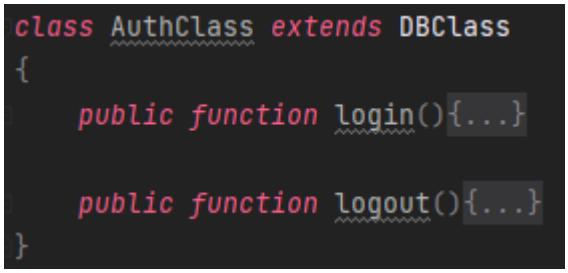

Рисунок 6 – класс AuthClass для авторизации

2.3 Event Handler

Вызовом api управляет класс EventHandler, приведенный на рисунке 7.

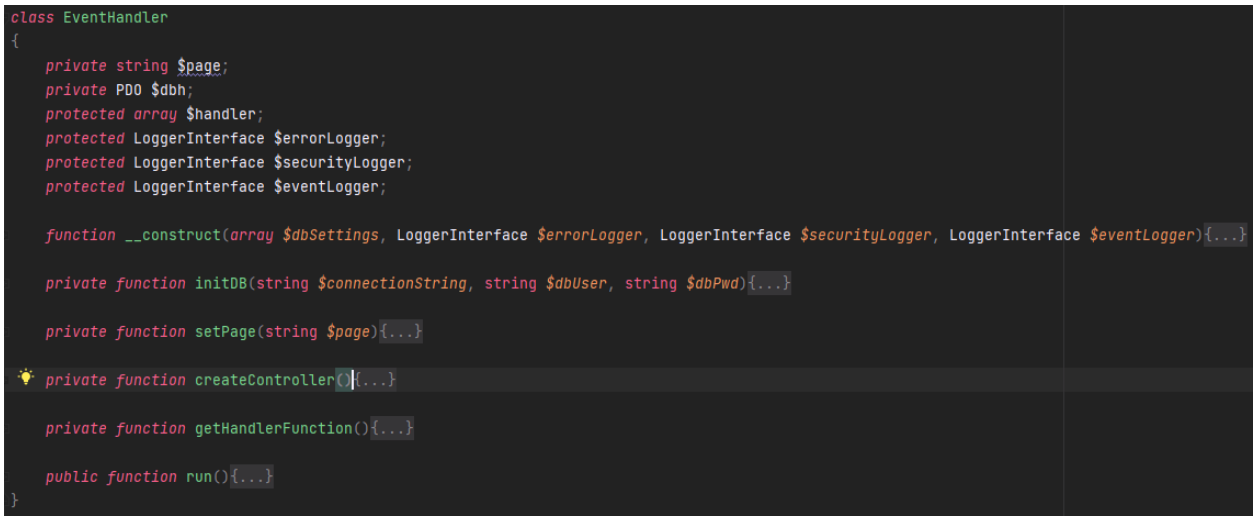

Рисунок 7 – класс EventHanler

У класса есть конструктор, инициализирующий соединение с БД, инициализирующий логгеры. Также внутри конструктора вызывается метод setPage, который определяет, к какому api надо будет делать запрос, а также определяющий, имеет ли пользователь достаточные права для обращения к выбранному api.Метод run() отвечает непосредственно за обращение к api, и вывод результата запроса на экран.

3. Index и settings

В данном пункте рассмотрим файлы индекса и настроек приложения.

3.1 Index.php

На рисунке 8 представлен index.php.

```php
<?php
require_once '../safeplace/settings.inc.php';
require_once MODULES_DIR.'autoloader.php';

use App\EventHandler;
use App\Logger\FileLoggerBuf;

$errorLogger = new FileLoggerBuf( fileName: errorLogPath,  bufferSize: 1);
$securityLogger = new FileLoggerBuf( fileName: securityLogPath,  bufferSize: 1);
$eventLogger = new FileLoggerBuf( fileName: eventLogPath,  bufferSize: 1);

try{
    session_start();
    $app = new EventHandler($dbSettings, $errorLogger, $securityLogger, $eventLogger);
    $app->run();
}
catch (Exception $e)
{
    $errorLogger->logEvent($e->getMessage(), $e->getFile(), $e->getLine(), $e->getTraceAsString());
    echo json_encode([]);
}
```

Рисунок 8 – содержание index.php

В данном файле создаются логгеры, создается сессия, создается объект класса EventHandler, и происходит вызов метода run().

3.2 settings.inc.php

На рисунке 9 изображен файл настроек settings.inc.php

```php
const MODULES_DIR = '../safeplace/';
const DEBUG = true;

const errorLogPath = '../log/error.log';
const securityLogPath = '../log/security.log';
const eventLogPath = '../log/event.log';

const sessPath = '../sessions/';
ini_set( option: 'session.save_path',  value: sessPath);

$dbSettings = [
    'connectionString' => 'mysql:host=localhost;dbname=db2',
    'dbUser' => 'egor',
    'dbPwd' => '1234'
];
```

Рисунок 9 – содержание settings.inc.php

Здесь указываются константы, которые в большинстве своем являются путями до файлов логирования, директории для сессий. Также задается connectionString для подключения к БД.

4. Autoloader

В приложении используется автозагрузчик классов, основанный на предложенном во время обучения. Содержание файла представлено на рисунке 10.

```php
if (!defined( constant_name: 'MODULES_DIR')) {
    throw new RuntimeException( message: 'config not set');
}

spl_autoload_register(
    function ($class_name)
    {
        if (file_exists( filename: MODULES_DIR.$class_name.'.php'))
        {
            require MODULES_DIR.$class_name.'.php';
        }
    }
);
```

Рисунок 10 – файл autoloader.php

5. Логирование

В приложении присутствуют три файла логирования.

- error.log

  error.log используется для записи всех ошибок, не относящихся к безопасности. Ошибки подключения к базе данных, неправильные методы и параметры запросов, все это пишется сюда.

- security.log

  Этот лог-файл предназначен строго для ошибок, вызванных несанкционированными действиями пользователя. Неавторизованный доступ к приватным api, неудачные попытки авторизации пишутся в этот файл.

- event.log

  event.log используется для логирования выполненных запросов. Каждый выполненный запрос будет отражен в данном файле.

Для логирования были предложены классы Logger, которые были рассмотрены в ходу обучения.

6. Вызовы из Insomnia

Для каждого метода классов FilmsClass, ActorsClass, AuthClass были написаны запросы в REST-клиенте. Все нестандартные ситуации были отражены в запросах. Ниже будут приведены несколько скриншотов ответов Insomnia на запросы. Приводить все нецелесообразно, они будут доступны для просмотра и взаимодействия.

Рисунок 11 – запрос на выборку записей из сущности actors.

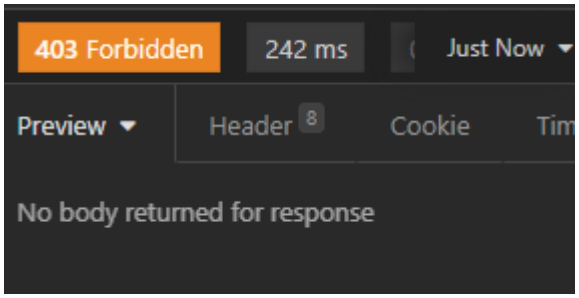

Рисунок 12 – запрос к сущности actors методом get. Так как мы неавторизованы, то и ответ соответствующий.

Рисунок 13 – запрос тем же методом к той же сущности, но после авторизации

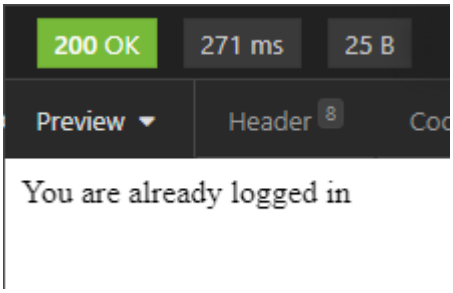

Рисунок 14 – ответ на повторную авторизацию

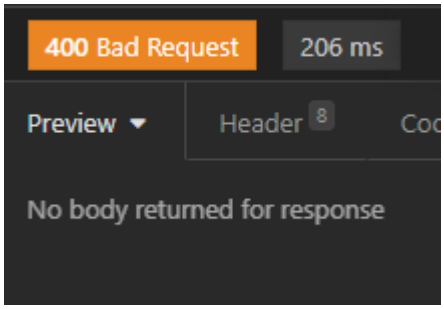

Рисунок 15 – ответ на любой запрос, выполненный неправильным методом.

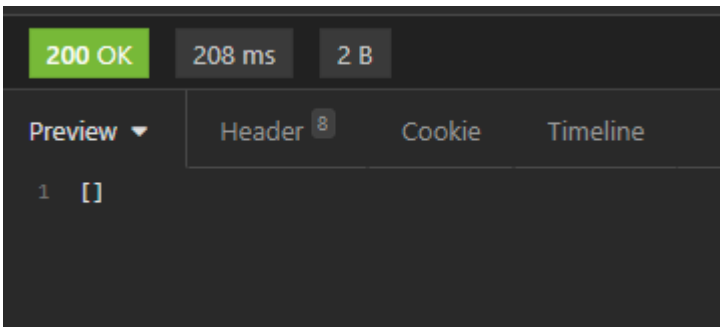

Рисунок 16 – ответ на любой запрос с некорректно заданными параметрами.

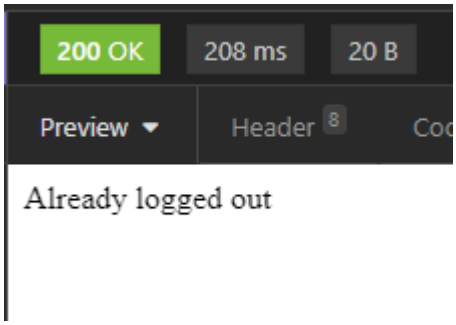

Рисунок 17 – ответ на повторный выход из системы

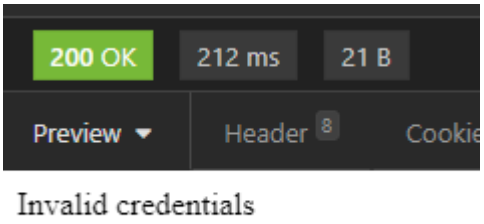

Рисунок 18 – ответ на любую неверную авторизацию

7 Выводы о проделанной работе

В ходе проделанной работы, были получены знания по разработке web-приложения. Был получен опыт работы с классами, наследованием и полиморфизмом. Был освоен материал, связанный с пространствами имен. Были углублены знания и навыки по работе с языком программирования PHP, а также с Insomnia.

8 Список использованной литературы

1. Задание 9.1 (приложение на классах) – методические указания https://drive.google.com/file/d/1y5fKGZdap7RvMewHz2T_WO3HBaJB1aRc/view
2. 9.1 Структура приложения – лекция в формате презентации https://classroom.google.com/u/0/w/MzIwMjkyNjk3MTMx/t/all
3. 8.2 Автозагрузка – лекция в формате презентации https://classroom.google.com/u/0/c/MzIwMjkyNjk3MTMx
4. 8.1 Пространство имен - лекция в формате презентации https://classroom.google.com/u/0/w/MzIwMjkyNjk3MTMx/t/all