**Аннотация**

Этот проект описывает разработку дизайна веб-приложения и его тестирование для осуществления управления спортивным клубом «Черные медведи». Веб-приложение позволяет спортсменам записываться на тренировки, а тренеры могут формировать записи для тренировок на основе этих записей. На сайте отображается расписание матчей клуба. Администратор может управлять информацией о спортсменах и тренерах, включая изменение расписания тренировок и удаление записей.

## 

**Содержание**

[Введение 4](#_Toc184167390)

[1. Назначение и область применения 5](#_Toc184167391)

[2. Постановка задачи 6](#_Toc184167392)

[3. Описание программы 7](#_Toc184167393)

[3.1 Общие сведения 7](#_Toc184167394)

[3.2 Функциональное назначение 7](#_Toc184167395)

[3.3 Описание логической структуры 8](#_Toc184167396)

[3.4 Используемые технические средства 9](#_Toc184167397)

[3.5 Вызов и загрузка 10](#_Toc184167398)

[3.6 Входные и выходные данные 10](#_Toc184167399)

[4. Программа и методика испытаний 11](#_Toc184167400)

[4.1 Объекты испытаний 11](#_Toc184167401)

[4.2 Цель испытаний 11](#_Toc184167402)

[4.3 Требования к программе 11](#_Toc184167403)

[4.4 Методы испытаний 12](#_Toc184167404)

[4.5 Тестовый пример 14](#_Toc184167405)

[5. Руководство оператора 15](#_Toc184167406)

[5.1 Выполнение программы 15](#_Toc184167407)

[5.2 Сообщение оператору 20](#_Toc184167408)

[6. Мероприятия по информационной безопасности 21](#_Toc184167409)

[Заключение 23](#_Toc184167410)

[Приложения. Текст программы 24](#_Toc184167411)

[Источники, использованные при разработке 28](#_Toc184167412)

**Введение**

В данном документе представлены требования к дизайну, безопасности и функциональности веб-приложения, которое будет служить инструментом для автоматизации процессов управления в спортивном клубе «Спортсила».

Цель данного приложения — улучшить взаимодействие между сотрудниками клуба, оптимизировать планирование тренировок, учёт спортсменов и тренеров, а также формирование отчётности. Для достижения этой цели необходимо разработать интуитивно понятный и простой дизайн, а также провести тщательное тестирование приложения, чтобы исключить возможные ошибки.

Разрабатываемое приложение будет использоваться исключительно внутри спортивного клуба «Спортсила».

**1. Назначение и область применения**

Разработка для веб-приложения дизайна и тестирования предназначено для понятного, красивого, безопасного, функционально-верного веб-приложения. Верно и безопасно должна отрабатывать такая функция - автоматизации управления спортивным клубом. Она позволяет эффективно планировать тренировки, вести учет спортсменов и тренеров, организовывать матчи. Область применения — спортивные клубы, секции, организации, проводящие спортивные мероприятия. Аналогами данного приложения являются: Google Calendar, MS Outlook Calendar, Notion, но у этих приложений есть недостатки, поэтому было принятое решение создать собственное приложение.

**2. Постановка задачи**

В данном разделе необходимо четко сформулировать задачи, которые решаются с помощью веб-приложения:

1. Описание задач для спортсмена (понятная и безопасная запись на тренировки, просмотр расписания, личной информации).
2. Описание задач для тренера (понятный и безопасный просмотр свободных спортсменов, назначение тренировок, просмотр расписания).
3. Описание задач для администратора (понятное и безопасное добавление/удаление спортсменов и тренеров, редактирование данных, организация матчей).

## 

**3. Описание программы**

**3.1 Общие сведения**

Веб-приложение «BALANCE PRO» для управления спортивным клубом разработано с использованием следующих технологий:

1. **Языки программирования**: PHP (версия 8.1), HTML5, CSS3, JavaScript (с использованием библиотеки jQuery).
2. **Фреймворки**: Bootstrap (для CSS).
3. **Библиотеки**: не использовались.
4. **СУБД**: MySQL (версия 8.0).

Веб-приложение «BALANCE PRO» для управления спортивным клубом тестировалось с помощью:

1. рпорл
2. фыв

**3.2 Функциональное назначение**

Приложение разделено на следующие функциональные блоки:

1. **Модуль авторизации**: Предоставляет возможность входа в систему для спортсменов, тренеров и администратора. Включает регистрацию новых пользователей.
2. **Модуль управления профилем**: Позволяет пользователям просматривать и редактировать свою личную информацию.
3. **Модуль управления тренировками**: Для спортсменов – составление списка свободных дат и свободного времени для занятия спортом. Для тренеров – просмотр записей на тренировки и назначение тренировок спортсменам. Для администраторов – глобальное управление расписанием, редактирование/удаление слотов и назначений.
4. **Модуль управления данными спортсменов**: Администратор может добавлять, редактировать и удалять информацию о спортсменах.
5. **Модуль управления данными тренеров**: Администратор может редактировать и удалять информацию о тренерах.
6. **Модуль управления матчами**: Ввод и отображение информации о предстоящих матчах спортивного клуба.
7. **Административный модуль**: Предоставляет администратору полный доступ ко всем функциям системы.

**3.3 Описание логической структуры**



Рисунок 1 - Функциональная схема веб-приложения

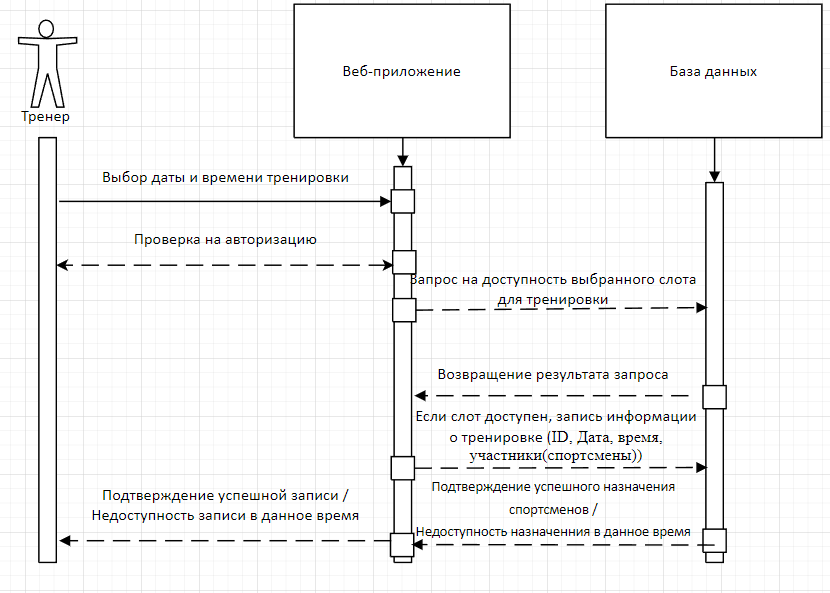


Рисунок 2 - Диаграмма последовательности. Роль - тренер

**3.4 Используемые технические средства**

1. **Серверная часть**: PHP 8.1.
2. **База данных:** MySQL 8.0, хранящая информацию о пользователях, расписании тренировок, матчах и т.д.
3. **Библиотеки PHP**: PDO (PHP Data Objects) для взаимодействия с базой данных.
4. **Фреймворки/Библиотеки JavaScript**: jQuery для упрощения работы с DOM.
5. **Фреймворки CSS**: Bootstrap для создания адаптивного дизайна.
6. **Хостинг**: Hestia.

**Дополнения к описанию входа в PhpMyAdmin**:

При необходимости доступа к PhpMyAdmin для администрирования базы данных, вход осуществляется через веб-интерфейс управления хостингом Hestia (или указание конкретного места входа, предоставляемого Hestia). Входные данные для PhpMyAdmin (логин и пароль) хранятся в учетной записи хостинга Hestia.

Веб-приложение «BALANCE PRO» для управления спортивным клубом тестировалось с помощью:

1. Рпорл

**3.5 Вызов и загрузка**

Пользователь, войдя на сайт спортивного клуба, выбирает свою роль: спортсмен, тренер или администратор. Система отображает соответствующую форму входа или регистрации. После регистрации спортсмена предлагается авторизоваться (используются надежные инструменты сохранности и шифрования данных). Ввод логина и пароля проверяется с данными в базе данных (пароли сверяются в зашифрованном виде с обоих сторон (и со стороны сервера, и со стороны пользователя) проверки). Успешная авторизация перенаправляет спортсмена на персональную страницу с личными данными и расписанием тренировок. Для поддержания авторизации используется механизм сессий и cookies.

* 1. **Входные и выходные данные**

**Входные данные:**

1. **Регистрация**: Логин, пароль, имя, фамилия, отчество, контактная информация (телефон), дата рождения.
2. **Запись на тренировку**: Дата и время тренировки.
3. **Администратор**: Данные для управления пользователями и расписанием.

**Выходные данные:**

1. **Главная страница**: Информация о клубе, таблица с расписанием предстоящих матчей.
2. **Личный кабинет спортсмена**: Личная информация, расписание тренировок, возможность записи на тренировки.
3. **Личный кабинет тренера**: Личная информация, список спортсменов, расписание тренировок, возможность назначения тренировок.
4. **Административная панель**: Интерфейс для управления пользователями, расписанием и матчами (таблицы, формы редактирования).
5. **Панель ошибки:** Информация о произошедшем сбое веб-приложения с вариантами решения проблемы, если причина на стороне пользователя.

**4. Программа и методика испытаний**

**4.1 Объекты испытаний**

Веб-приложение спортивного клуба, модуль «Черные медведи», включающий в себя функционал регистрации, авторизации и личного кабинета. Объектами тестирования являются:

1. Форма регистрации пользователя-спортсмена.
2. Форма авторизации пользователя-спортсмена.
3. Личный кабинет пользователя-спортсмена (отображение личных данных, расписание тренировок).
4. База данных пользователей.
5. Серверная часть приложения, обрабатывающая запросы от клиента.

**4.2 Цель испытаний**

Проверить соответствие функциональности модуля «BALANCE PRO» заявленным требованиям, выявляя ошибки и дефекты в работе системы. Цель – убедиться в стабильности, безопасности и удобстве использования модуля для пользователей-спортсменов.

* 1. **Требования к программе**

1. Регистрация должна быть возможна только с уникальным логином и телефоном.
2. Пароль должен соответствовать определенным критериям сложности (минимальная длина, наличие цифр и символов).
3. Авторизация должна быть быстрой и безопасной, используя механизм сессий.
4. Личный кабинет должен отображать актуальные данные пользователя и расписание тренировок.
5. Система должна обрабатывать ошибки ввода данных (невалидный телефон, неверный формат данных) и выдавать понятные сообщения об ошибках.
6. Система должна быть защищена от SQL-инъекций и XSS (межсайтовое исполнение сценариев).
7. Дизайн страниц единообразен, прост и понятен пользователю.
8. Сайт должен загружаться за максимально быстрое время на устройствах пользователя.
9. Наличие безопасного и функционального лимита времени активной сессии пользователя и cookies.
10. Выполнение логирования сайта.

**4.4 Методы испытаний**

Используемые методы: unit-тесты, функциональное тестирование, интеграционное тестирование, тестирование безопасности, тестирование производительности (опционально), нефункциональное тестирование, тестирование пользовательского интерфейса (UI).

**Примеры тестовых сценариев:**

1. **Регистрация:**

* SCENARIO\_1: Регистрация с корректными данными. Ожидаемый результат: успешная регистрация, перенаправление на страницу авторизации.
* SCENARIO\_2: Регистрация с некорректными данными (например, не валидный номер телефона). Ожидаемый результат: сообщение об ошибке, блокировка регистрации.
* SCENARIO\_3: Попытка регистрации с уже существующим логином. Ожидаемый результат: сообщение об ошибке.

1. **Авторизация:**

* SCENARIO\_4: Авторизация с корректным логином и паролем. Ожидаемый результат: успешная авторизация, перенаправление на личный кабинет.
* SCENARIO\_5: Авторизация с некорректным логином/паролем. Ожидаемый результат: сообщение об ошибке.
* SCENARIO\_6: Проверка работы сессий (авторизация с одного устройства, выход, попытка доступа без повторной авторизации). Ожидаемый результат: доступ запрещен.

1. **Личный кабинет:**

* SCENARIO\_7: Проверка отображения личных данных. Ожидаемый результат: корректное отображение всех данных пользователя.
* SCENARIO\_8: Проверка отображения расписания тренировок. Ожидаемый результат: корректное отображение актуального расписания.
* SCENARIO\_9: Редактирование личных данных. Ожидаемый результат: успешное сохранение изменений.
* SCENARIO\_10: Проверка обработки ошибок (например, попытка редактирования с некорректными данными). Ожидаемый результат: сообщение об ошибке.

**4.5 Тестовый пример**

| **ID** | **Тестовый сценарий** | **Шаги** | **Ожидаемый результат** | **Результат** | **Статус** | **Приоритет** |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | Регистрация с корректными данными | 1. Ввести корректные данные в форму регистрации. 2. Нажать кнопку “Зарегистрироваться”. | Успешная регистрация, переход на страницу авторизации | Успешная регистрация, переход на страницу авторизации | Пройден | Критический |
| 2 | Регистрация с невалидным паролём | 1. Ввести некорректный пароль. 2. Нажать кнопку “Зарегистрироваться”. | Сообщение об ошибке «Пароль слишком короткий(надо больше 4 символов)!». | Сообщение об ошибке «Пароль слишком короткий(надо больше 4 символов)!». | Пройден | Критический |
| 3 | Добавление свободного слота спортсменом | 1. Ввести корректный промежуток времени. 2. Нажать кнопку “Сохранить”. | Внизу тестовых полей появиться новый слот свободного времени спортсмена для тренировки. | Внизу тестовых полей появиться новый слот свободного времени спортсмена для тренировки. | Пройден | Высокий |
| 4 | Просмотр доступных спортсменов тренером | 1. Открыть тренерскую страницу сайта.  2. Просмотр слотов спортсменов, если такие уже добавлены и назначение необходимых. | Тренер назначил одного спортсмена на тренировку. | Тренер не назначил ни одного спортсмена на тренировку, так как нет свободных спортсменов. | Пройден | Высокий |

**5. Руководство оператора**

**5.1 Выполнение программы**

**5.1.1 Пользователь-Спортсмен:**

**Регистрация:**

1. Откройте веб-сайт спортивного клуба.

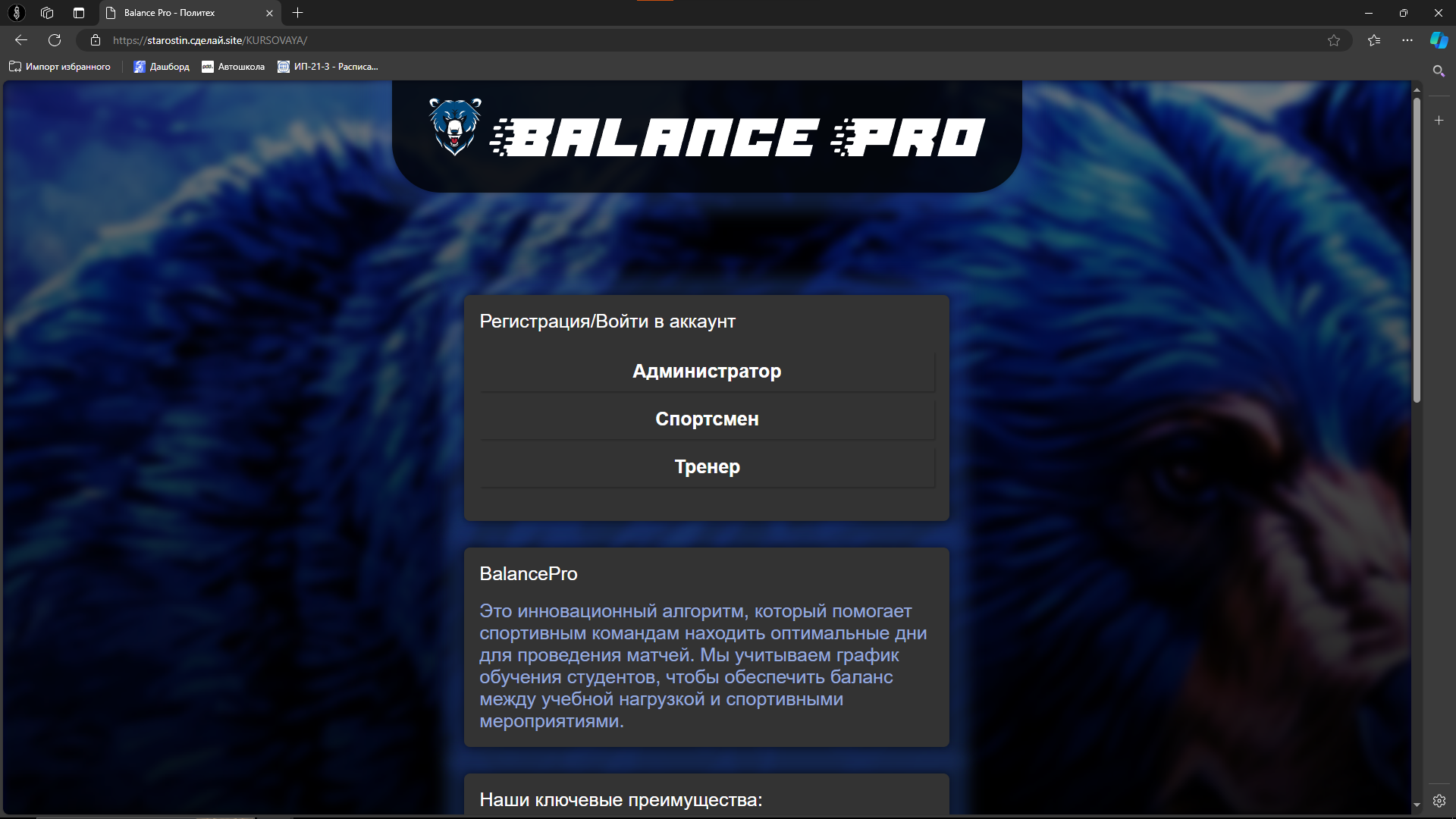


Рисунок 3 - Главная страница веб-сайта

1. Нажмите на кнопку «Спортсмен» в блоке «Регистрация/Войти в аккаунт»
2. Заполните все необходимые поля формы. Убедитесь, что введенные данные соответствуют требованиям.

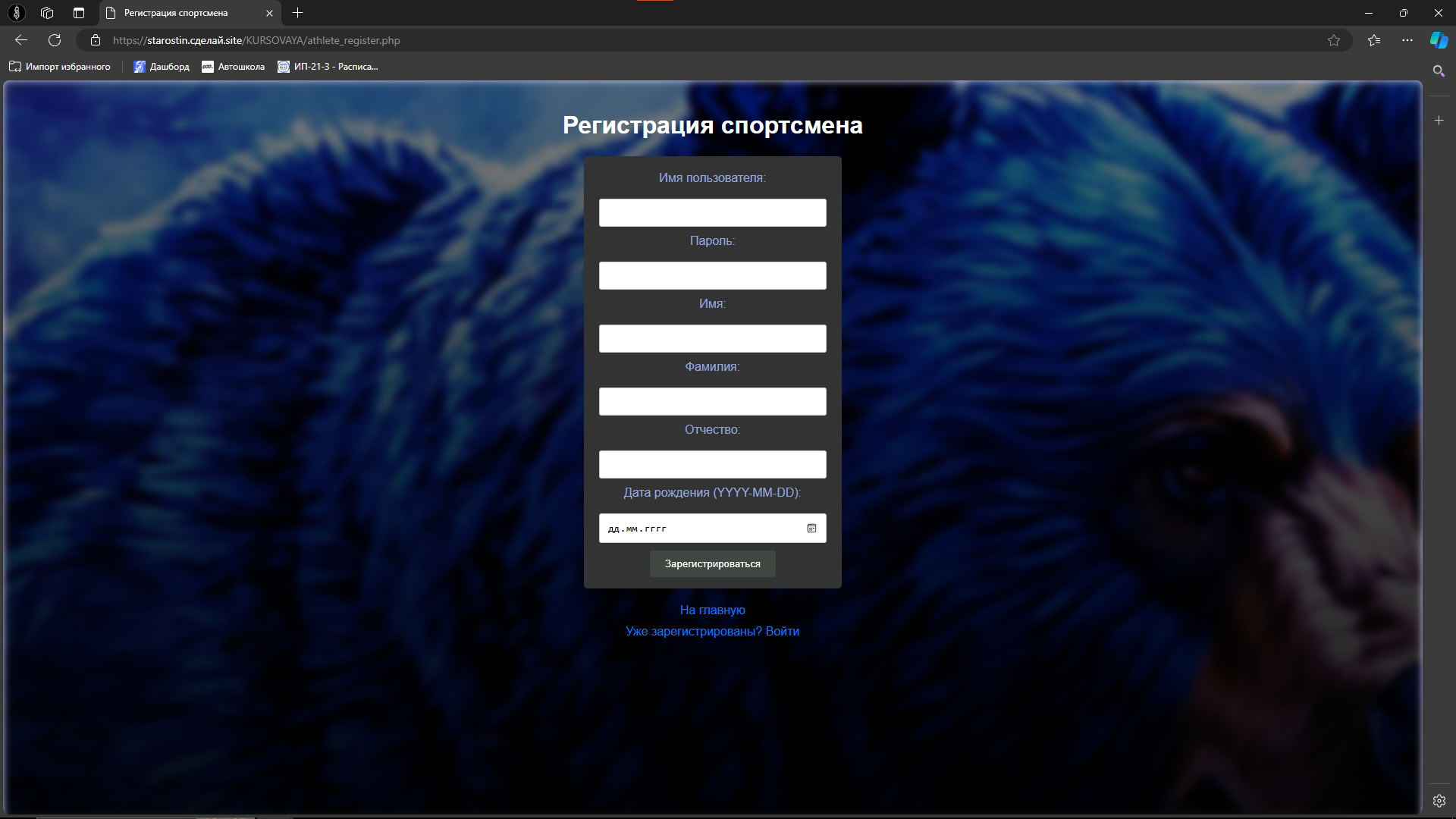


Рисунок 4 - Форма регистрации для спортсмена

1. Проверьте правильность заполнения данных.
2. Нажмите кнопку “Зарегистрироваться”.
3. Система проверит корректность данных и уникальность логина. Если все в порядке, вы получите сообщение об успешной регистрации и сможете перейти к авторизации. В случае ошибок, система отобразит сообщение об ошибке, указывая на неверный формат или недопустимое значение введенного поля.

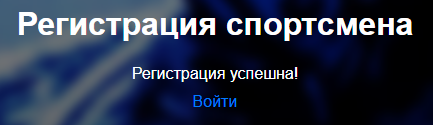


Рисунок 5 - Успешное сообщение о завершении регистрации

**Авторизация:**

1. После регистрации система предлагает авторизоваться.
2. Нажмите на кнопку “Войти”.
3. Введите свой логин и пароль.

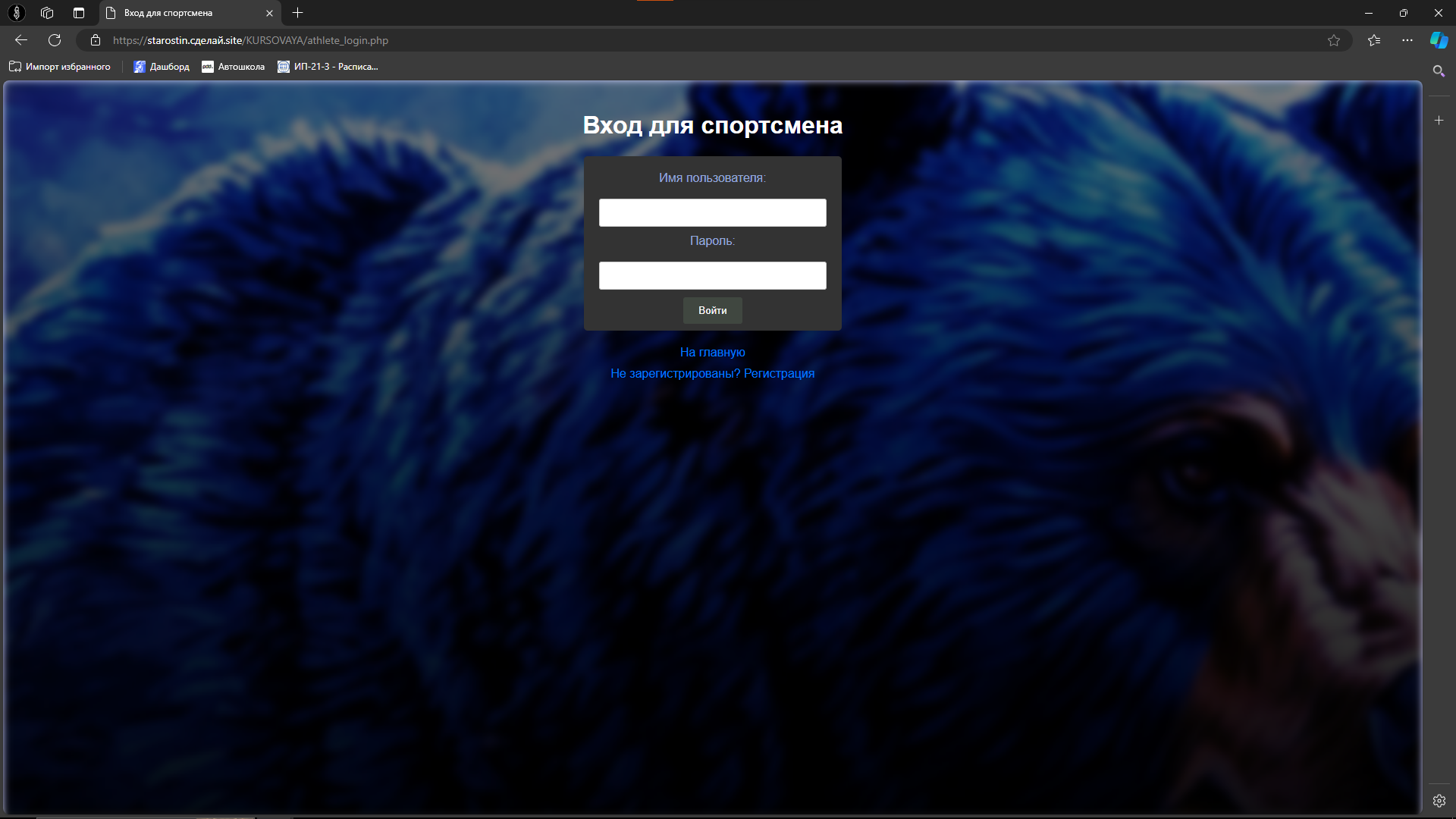


Рисунок 6 - Форма входа для спортсмена

1. Нажмите кнопку “Войти”.
2. Если данные верны, система перенаправит вас на вашу личную страницу (личный кабинет). Если данные неверны, система отобразит сообщение об ошибке. Пожалуйста, проверьте правильность написания логина и пароля.

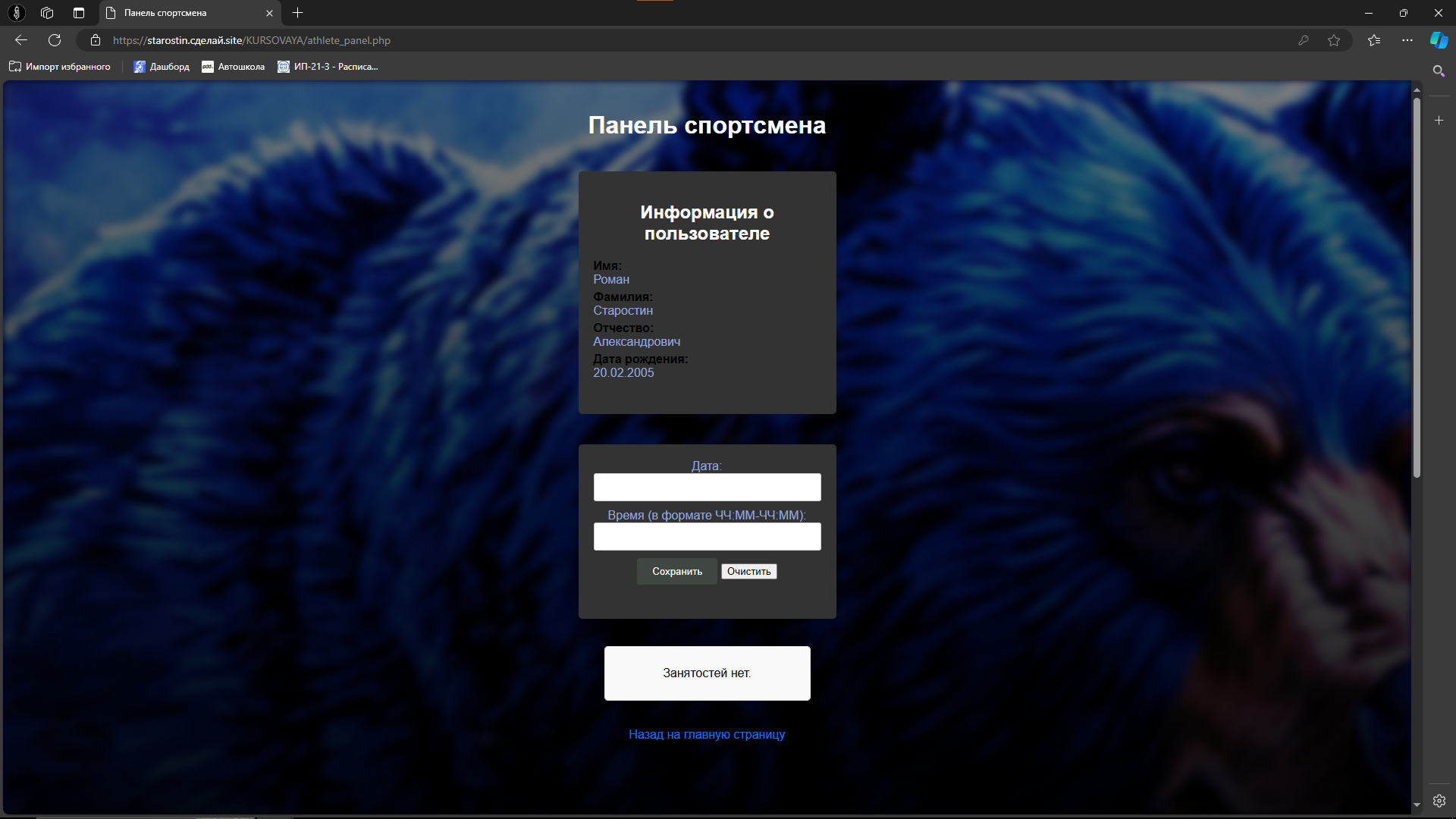


Рисунок 7 - Личный кабинет спортсмена

**Работа с личным кабинетом:**

1. В личном кабинете вы можете просматривать и редактировать свои личные данные.
2. Планировать свои тренировки.

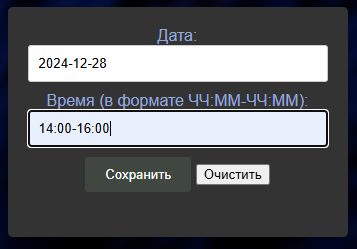


Рисунок 8 - Выставление даты и время для тренировки

1. Просматривать расписание тренировок.

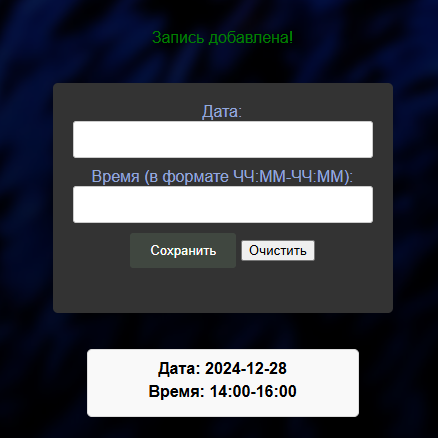


Рисунок 9 - Отображение добавленной записи

**5.1.2 Пользователь-Тренер:**

**Регистрация/Авторизация:** Аналогично пользователю-спортсмену. Вместо личных данных могут быть данные о квалификации и опыте.

**Работа с личным кабинетом:**

1. Просматривать расписание тренировок.
2. Брать спортсмена на тренировки.

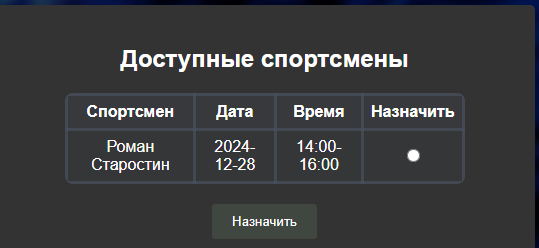


Рисунок 10 - Добавление спортсмена для проведения тренировок

1. Создавать расписание тренировок для своих учеников.

**5.1.3 Администратор:**

**Авторизация**: Отдельная форма входа для администратора.

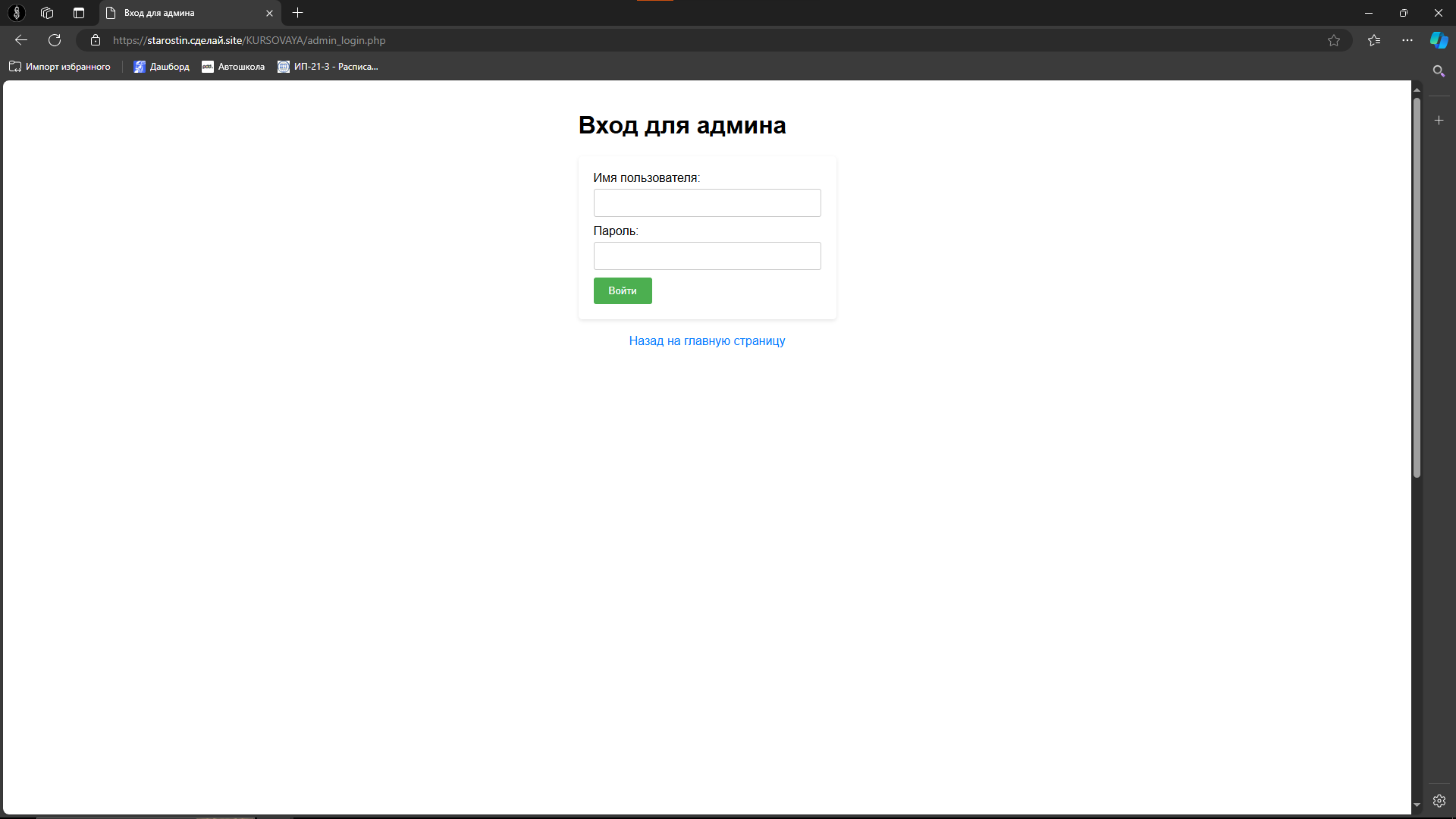


Рисунок 11 - Форма входа для администратора

**Работа с личным кабинетом**:

1. Управление пользователями (добавление, удаление, изменение данных спортсменов и тренеров).
2. Управление расписанием тренировок.
3. Управление данными тренировок.

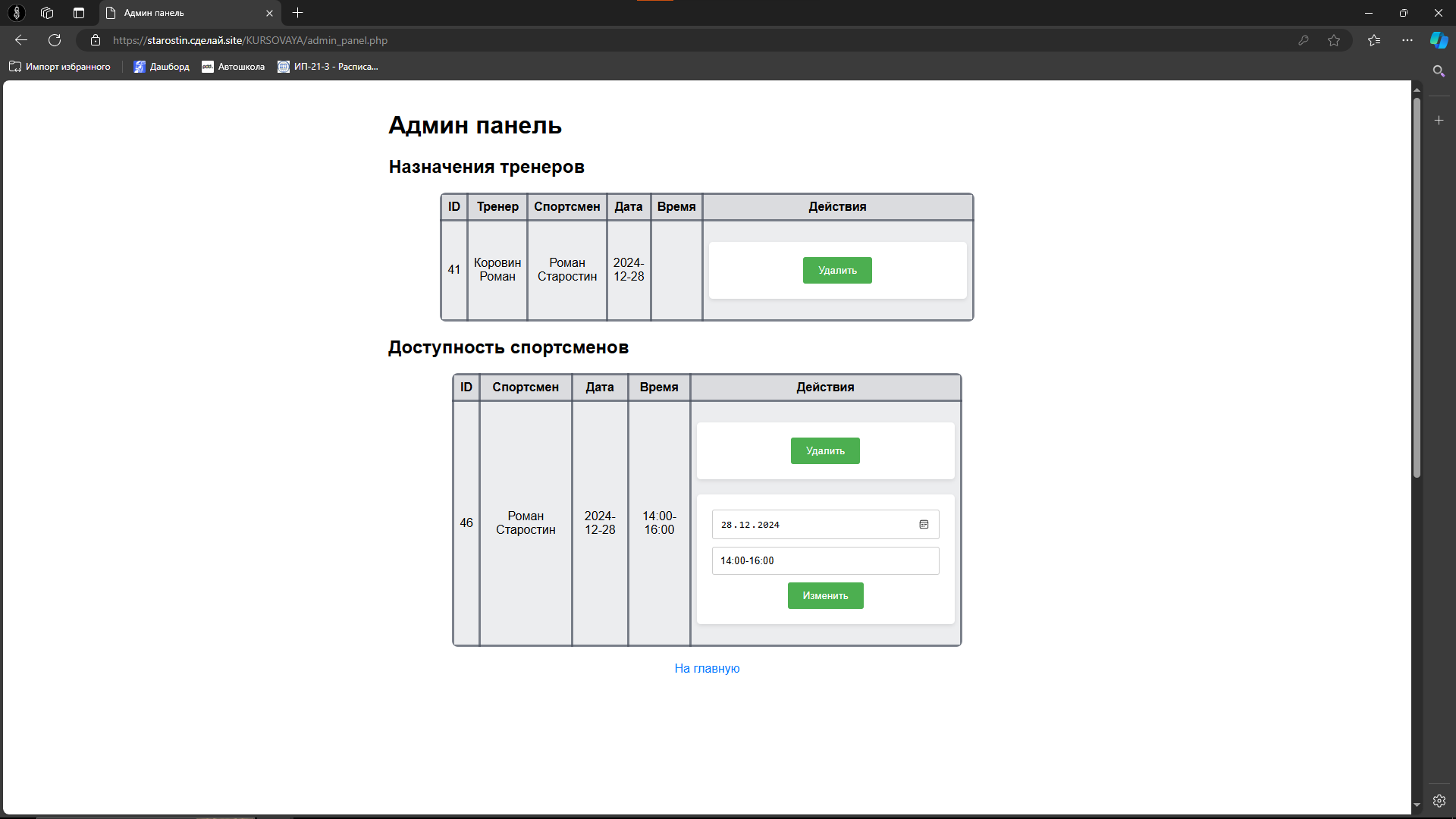


Рисунок 12 - Форма управления пользователями

**Unit-тест:**

Тест для страницы регистрации спортсмена для проверки возможности регистрации. Для выполнения теста надо:

1. В папке “Test” лежат все тесты для данного сайта.
2. Открываем в любом редакторе кода код сайта и там открываем консоль и пишем команду phpunit ...php – вместо «..» напиши название файла с тестом (например, AthleteRegistrationUnitTest.php) и после чего будет вывод сообщения в консоли об статусе теста.

**Текст кода теста (фрагмент):**

<?php

namespace Tests;

use PHPUnit\Framework\TestCase;

use PDO;

use stdClass;

/\*\*

 \* Класс для тестирования функций регистрации спортсменов.

 \*/

class AthleteRegistrationTest extends TestCase

{

    /\*\*

     \* Мок объекта PDO для имитации работы с базой данных.

     \* @var \PHPUnit\Framework\MockObject\MockObject

     \*/

    private $pdoMock;

    /\*\*

     \* Метод, выполняемый перед каждым тестом.

     \* Здесь настраивается мок PDO.

     \*/

    protected function setUp(): void

    {

        parent::setUp();

        $this->pdoMock = $this->createMock(PDO::class);

        $this->pdoMock->method('prepare')->willReturn($this->createMock(\PDOStatement::class));

    }

/\*\*

     \* Тест для случая, когда пользователь уже существует.

     \*/

    public function testUsernameAlreadyExists()

    {

        $username = 'existinguser';

        $password = 'password123';

        $firstName = 'John';

        $lastName = 'Doe';

        $patronymic = 'Patronovich';

        $birthdate = '1990-01-01';

        // Подготовка мока PDO для проверки существования пользователя

        $this->pdoMock->expects($this->once())

            ->method('prepare')

            ->with('SELECT id FROM athletes WHERE username = ?')

            ->will($this->returnValue($this->createMock(\PDOStatement::class)))

            ->shouldBeCalledTimes(1);

        // Вызов приватного метода register с параметрами регистрации существующего пользователя

        $result = $this->callPrivateMethod('register', [$username, $password, $firstName, $lastName, $patronymic, $birthdate]);

        // Проверка результата

        $this->assertEquals('Имя пользователя уже занято.', $result['error']);

        $this->assertFalse(isset($result['success']));

    }

**Вывод результата:**

Произошел сбой из-за того, что имя пользователя уже занято:

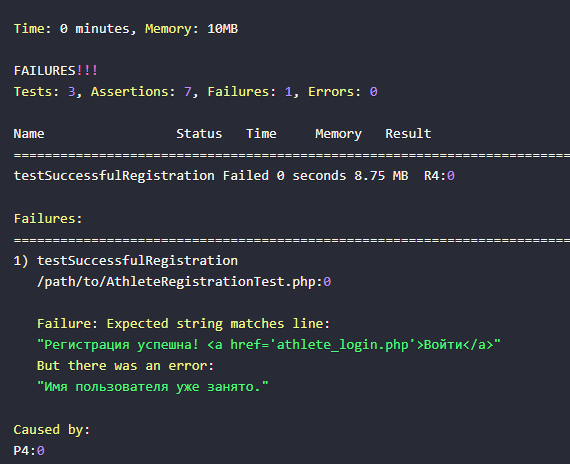
****

Рисунок 13 – не удачный unit тест

**Интеграционное тестирование:**

Тест для страницы регистрации спортсмена для проверки возможности регистрации. Тут проверка взаимодействия между компонентами/модулями системы.

**Текст кода теста (фрагмент):**

<?php

namespace App\Tests;

use App\Entity\Athlete;

use App\Repository\AthleteRepository;

use Symfony\Bundle\FrameworkBundle\Test\KernelTestCase;

/\*\*

 \* Класс для интеграционного тестирования функций регистрации спортсменов.

 \*/

class AthleteRegistrationIntegrationTest extends KernelTestCase

{

    /\*\*

     \* Объект EntityManager для работы с базой данных.

     \* @var \Doctrine\ORM\EntityManagerInterface

     \*/

    private $entityManager;

    /\*\*

     \* Репозиторий для работы с объектами Athlete.

     \* @var \App\Repository\AthleteRepository

     \*/

    private $athleteRepository;

    /\*\*

     \* Метод, выполняемый перед каждым тестом.

     \* Здесь инициализируется EntityManager и AthleteRepository.

     \*/

    protected function setUp(): void

    {

        parent::setUp();

        // Получаем EntityManager из контейнера Symfony

        $this->entityManager = static::$kernel->getContainer()->get('doctrine')->getManager();

        // Получаем репозиторий Athlete из EntityManager

        $this->athleteRepository = $this->entityManager->getRepository(Athlete::class);

    }

    /\*\*

     \* Тест для проверки успешной регистрации нового пользователя.

     \*/

    public function testSuccessfulRegistration()

    {

        // Создаем клиент для отправки запроса

        $client = static::createClient();

        // Отправляем POST-запрос на эндпоинт регистрации

        $crawler = $client->request('POST', '/register', [

            'username' => 'newuser',

            'password' => 'password123',

            'firstName' => 'John',

            'lastName' => 'Doe',

            'patronymic' => 'Patronovich',

            'birthdate' => '1990-01-01'

        ]);

        // Проверяем, что после успешной регистрации происходит перенаправление на страницу входа

        $this->assertResponseRedirects('/login', 'Expected redirect after successful registration');

        // Проверяем HTTP-статус ответа

        $this->assertResponseStatusCodeSame(302, 'Expected HTTP status code 302');

        // Проверяем, что пользователь зарегистрирован в базе данных

        $athlete = $this->athleteRepository->findOneBy(['username' => 'newuser']);

        $this->assertNotNull($athlete, 'User should be registered in the database');

    }

**Вывод результата:**

Все тесты прошли успешно, что говорит о том, что система регистрации работает корректно в различных сценариях:

* Тест testSuccessfulRegistration подтвердил, что новый пользователь может быть успешно зарегистрирован.
* Тест testUsernameAlreadyExists показал, что система правильно обрабатывает попытку регистрации существующего пользователя.
* Тест testInvalidDate подтвердил корректную обработку некорректного формата даты рождения.

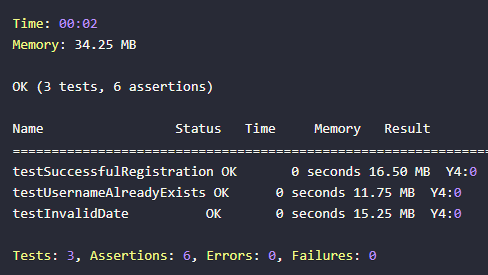
****

Рисунок 14 – удачный интеграционныйt тест

**Тестирование безопасности:**

**Защита через завершение сессий и cookies файлов:**

При авторизации идет установка cookies на 60 секунд (такое время в рамках теста):

# авторизация

$sql = 'SELECT id FROM users WHERE login = ? AND password = ?';

$query = $pdo->prepare($sql);

$query->execute([$login, $password]);

if ($query->rowCount() == 0)

{echo "Такого пользователя нет!";}

else

{

    setcookie('auth\_user', $login, time() + 60, '/');

    header('Location: /user.php');

}

**Проверка от sql-инъекций:**

Для проверки возьмем такой вредоносный ввод: «admin' ORDER BY 1--» и “password123”.

Результат: «Пользователь не найден»:

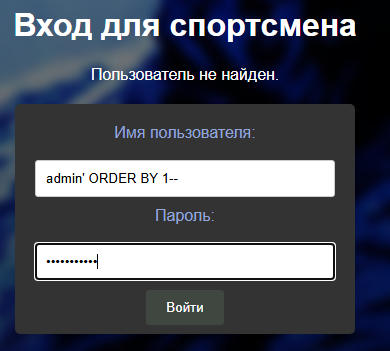


Рисунок 15 – инъекция не сработала

**Функциональное тестирование:**

Это тестирование можно проверить с помощью расширения Selenium IDE для браузеров.

**Тест:**

Тестирование страницы администратора на возможность изменения данных слота спортсмена и на возможность удаления пользователя:

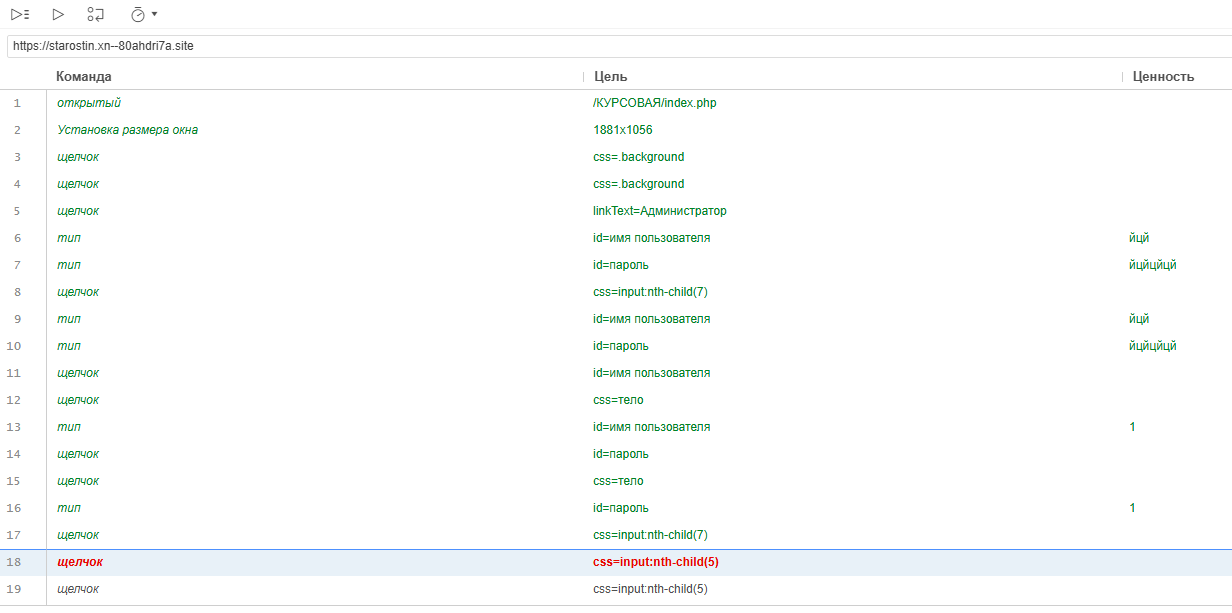


Рисунок 16 – список действий

**Результат:**

Тест провален, так как пользователя у которого надо поменять слот свободного времени был удален:

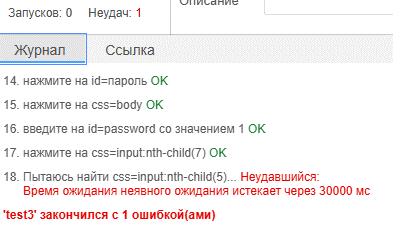


Рисунок 17 – тест провален

**Тестирование пользовательского интерфейса (UI):**

Проверка адаптивности сайта к различным размерам экранов устройств:

Проверка на устройстве с размерами 1800 на 1076px:

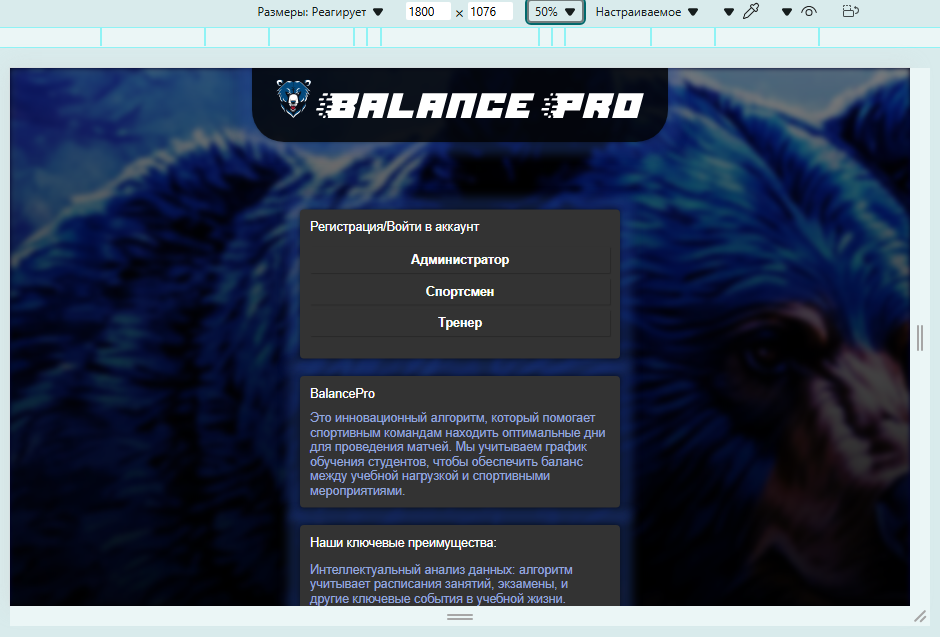


Рисунок 18 – на ноутбуки

И на устройстве с размерами 375 на 667px:

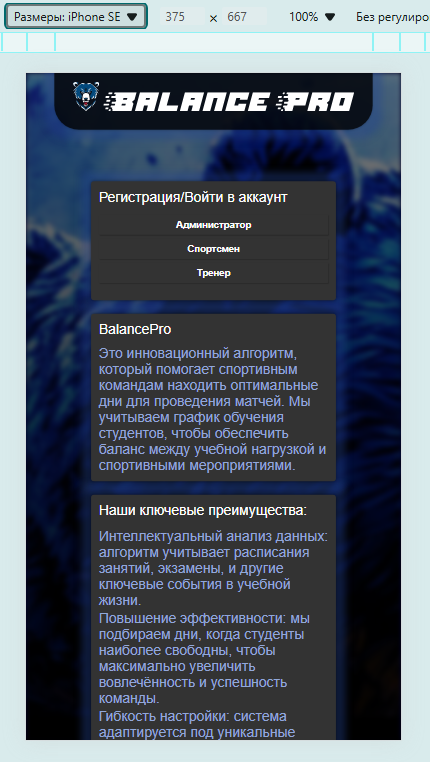


Рисунок 19 – на IPhone SE

**5.2 Сообщение оператору**

1. **Типы сообщений:**

* **Успешные операции**: «Регистрация прошла успешно!», «Авторизация успешна!», «Изменения сохранены!», «Тренировка добавлена/удалена!»

1. **Ошибки:**

* **Общие** **ошибки**: «Ошибка входа», «Ошибка авторизации», «Ошибка сохранения данных», «Ошибка соединения с сервером», «Неизвестная ошибка».
* **Ошибки** **ввода** **данных**: «Некорректный логин», «Некорректный пароль», «Некорректный номер телефона», «Поле обязательно для заполнения».
* **Ошибки** **доступа**: «У вас нет доступа к этой функции», «Нет прав доступа».
* **Технические** **ошибки**: «Ошибка базы данных», «Ошибка обработки запроса».

**6. Мероприятия по информационной безопасности**

**1. Защита от несанкционированного доступа:**

* **Аутентификация и авторизация:** Использование надежных методов аутентификации (логин/пароль) с проверкой на стороне сервера. Реализация механизма авторизации, ограничивающего доступ к функциям в зависимости от роли пользователя (спортсмен, тренер, администратор). Регулярное обновление паролей администраторами. Принудительная смена пароля через определённый промежуток времени.
* **Защита от SQL-инъекций:** Использование параметризованных запросов или хранимых процедур для предотвращения SQL-инъекций. Валидация и санитация пользовательского ввода.
* **Защита от XSS-атак:** Экранирование пользовательского ввода на стороне сервера и клиента, использование специализированных библиотек для предотвращения XSS-атак.
* **Защита от CSRF-атак:** Использование токенов CSRF для защиты от атак подделки межсайтовых запросов.
* **HTTPS:** Использование протокола HTTPS для шифрования данных, передаваемых между клиентом и сервером.
* **Брандмауэр:** Использование брандмауэра для защиты сервера от несанкционированного доступа из внешней сети.
* **Ограничение доступа к файловой системе:** Серверный код должен иметь ограниченные права доступа к файловой системе.

**2. Защита данных:**

* **Шифрование данных:** Шифрование конфиденциальных данных (пароли, личные данные) в базе данных.
* **Безопасное хранение паролей:** Хранение паролей в хешированном виде с использованием односторонних функций хеширования (например, bcrypt, Argon2). Хеширование с использованием соли.
* **Регулярное резервное копирование:** Регулярное создание резервных копий базы данных и файлов приложения. Хранение резервных копий в безопасном месте.
* **Контроль доступа к данным:** Реализация системы управления доступом к данным, позволяющей ограничивать доступ к данным в зависимости от роли пользователя.

**3. Мониторинг и аудит:**

* **Логирование:** Ведение подробных логов событий, включая попытки входа, ошибки и изменения данных.
* **Мониторинг системы:** Регулярный мониторинг системы на наличие уязвимостей и атак.
* **Аудит безопасности:** Регулярное проведение аудита безопасности системы для выявления и устранения уязвимостей.

**4. Управление уязвимостями:**

* **Регулярное обновление ПО:** Своевременное обновление веб-сервера, базы данных и используемого программного обеспечения для устранения известных уязвимостей.
* **Управление уязвимостями:** Использование системы управления уязвимостями для отслеживания известных уязвимостей и их своевременного исправления.

**5. Политика безопасности:**

* **Политика паролей:** Разработка и внедрение политики паролей, устанавливающей требования к сложности и периодичности смены паролей.
* **Политика доступа:** Разработка и внедрение политики доступа к системе, определяющей права доступа различных пользователей.
* **Политика реагирования на инциденты:** Разработка и внедрение плана реагирования на инциденты безопасности.

**Заключение**

В документе описаны программа и методика создания дизайна веб-приложения для спортивного клуба, включая руководство для оператора, инструкции по тестированию и информационной безопасности. Анализ выявил необходимость всестороннего тестирования функциональности сайта и обеспечения удобства использования для всех типов пользователей: спортсменов, тренеров и администраторов. Также требуется дальнейшее внедрение новых надёжных мер, технологий безопасности для защиты данных пользователей.

Успешное тестирование и внедрение мер безопасности гарантируют стабильную работу веб-приложения и высокий уровень обслуживания. Дальнейшая работа будет направлена на улучшение функциональности, усиление мер безопасности и регулярный мониторинг возможных сбоев через систему логирования событий на сайте.

**Приложения. Текст программы**

Ниже представлена схема базы данных для спортивного клуба «Черные медведи»:

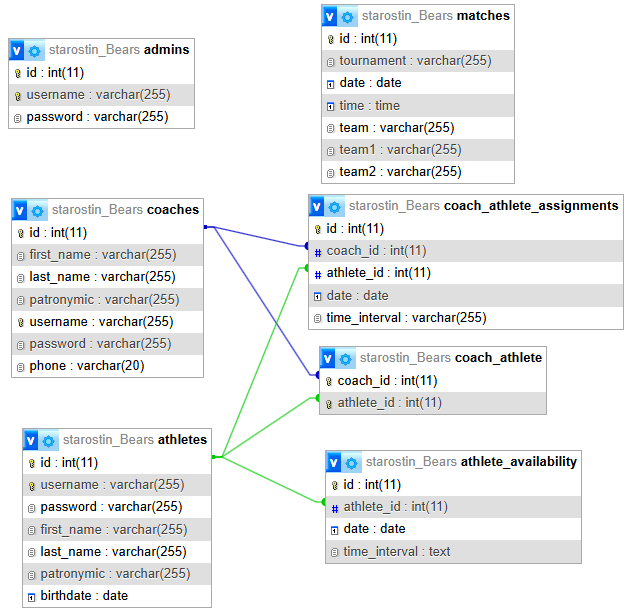


Рисунок 13 – Схема БД для спортивного клуба «Черные медведи».

Один из примеров кода, который я предоставляю ниже, отвечает за:

1. Body: основной стиль для тела документа.
2. Background: фоновое изображение с эффектами.
3. Background::before: прозрачный фон поверх фона.
4. Background::after: тень в виде линейного градиента.

Блок кода из стиля главной страницы:

body {

    font-family: 'Arial', sans-serif;

    margin: 0;

    background-color: black;

    padding: 20px;

    display: flex;

    flex-direction: column;

    align-items: center;

    min-height: 180vh;

    overflow: auto;

}

.background {

    position: absolute;

    top: 0;

    left: 0;

    width: 100%;

    height: 100%;

    background-image: url('uploads/image.png');

    background-size: cover;

    background-position: center;

    z-index: 1;

    filter: blur(5px);

}

.background::before {

    content: "";

    position: absolute;

    top: 0;

    left: 0;

    width: 100%;

    height: 100%;

    background: rgba(0, 0, 255, 0.5);

    filter: blur(10px);

    mix-blend-mode: overlay;

    z-index: 2;

}

.background::after {

    content: "";

    position: absolute;

    top: 0;

    left: 0;

    width: 100%;

    height: 120%;

    background: linear-gradient(to bottom, rgba(0, 0, 0, 0.5), rgba(0, 0, 0, 1));

    z-index: 3;

}

Скрипт php страницы спортсмена, который вызывается в блоке кода, который отвечает за отправку и сохранение данных об удобном времени для тренировок. Они хорошо защищены от SQL-инъекций за счет использования подготовленных запросов. Блок кода проверяет был ли отправлен POST-запрос, если нет, то вывод сообщения об ошибки, иначе запись в переменные данных пользователя и проверка полученных данных верности заполнения, если не верны, то вывод ошибки об не верном формате, иначе идет проверка на верность отработки внесения данных в бд и в случае ошибки будет выдаваться сообщением об этом, иначе успешно будут сохранены и выведены пользователю ниже окна заполнения текстовых полей.

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

    $date = $\_POST["date"];

    $time\_interval = $\_POST["time\_interval"];

    if (empty($date) || empty($time\_interval)) {

        $error = "Заполните все поля!";

    } else {

        if (!preg\_match('/^\d{4}-\d{2}-\d{2}$/', $date)) {

            $error = "Неверный формат даты. Используйте YYYY-MM-DD.";

        } elseif (!preg\_match('/^\d{2}:\d{2}-\d{2}:\d{2}$/', $time\_interval)) {

            $error = "Неверный формат времени. Используйте ЧЧ:MM-ЧЧ:MM.";

        } else {

            $sqlInsert = "INSERT INTO athlete\_availability (athlete\_id, date, time\_interval) VALUES (?, ?, ?)";

            $stmtInsert = $conn->prepare($sqlInsert);

            if ($stmtInsert === false){

                $error = "Ошибка при подготовке запроса: " . $conn->error;

            } else {

                $stmtInsert->bind\_param("iss", $athleteId, $date, $time\_interval);

                if ($stmtInsert->execute()) {

                    $success = "Запись добавлена!";

                    $stmtAvailability->execute();

                    $resultAvailability = $stmtAvailability->get\_result();

                } else {

                    $error = "Ошибка добавления: " . $stmtInsert->error;

                }

                $stmtInsert->close();

            }

        }

    }

**Источники, использованные при разработке**

**Нормативно-техническая документация:**

1. ГОСТ Р ИСО/МЭК 25051-2017 Требования к качеству готового к использованию программного продукта (RUSP) и инструкции по тестированию.
2. ЕСПД Единая система программной документации.

**Научно-техническая литература:**

1. Робин, Н. Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript, CSS и HTML5. — 2019.

**Интернет-ресурсы:**

1. PHP: Hypertext preprocessor [Электронный ресурс]. URL: [http://php.net](http://php.net/).
2. JavaScript Tutorial [Электронный ресурс]. URL: <http://w3schools.com/js>.
3. MySQL [Электронный ресурс]. URL: <https://www.mysql.com/>.
4. htmlbook.ru | Для тех кто делает сайты [Электронный ресурс]. URL: <http://htmlbook.ru/>.
5. CSSTricks - На ресурсе представлены обучающие материалы, советы и трюки по CSS. [Электронный ресурс]. URL: https://css-tricks.com/.
6. Codecademy - интерактивные уроки, опросники и упражнения по написанию кода для освоения CSS [Электронный ресурс]. URL: https://www.codecademy.com/.
7. Статья на сайте htmlacademy.ru «Тестирование сайтов: на что обратить внимание» [Электронный ресурс]. URL: https://www.codecademy.com/.
8. Статья на сайте sky.pro «Процесс проведения тестирования веб-сайтов». [Электронный ресурс]. URL: https://sky.pro.
9. «Обеспечение безопасности Web-сайтов» [Электронный ресурс]. URL: https://cyberleninka.ru/
10. «Рекомендации по обеспечению безопасности веб-сайта» [Электронный ресурс]. URL: https://www.geeksforgeeks.org/