Министерство науки и высшего образования РФ

Федеральное государственное автономное

образовательное учреждение высшего образования

«**СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ**»

Институт управления бизнес-процессами

Кафедра «Бизнес информатика и моделирование бизнес-процессов»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОЙ РАБОТЕ**

ПО ТЕМЕ: СОЗДАНИЕ WEB-ПРИЛОЖЕНИЯ С ПРИМЕНЕНИЕМ СОВРЕМЕННЫХ WEB-ТЕХНОЛОГИЙ ПО ТЕМАТИКЕ «КИНОТЕАТР»

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Никитин А.И.

подпись, дата

Студент УБ22-08Б, 432212194 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Александрова А.Е.

подпись, дата

Красноярск 2023

СОДЕРЖАНИЕ

[Введение 3](#_Toc156090011)

[1 Теоретическая часть 5](#_Toc156090012)

[1.1 HTML 5](#_Toc156090013)

[1.2 CSS 5](#_Toc156090014)

[1.3 JavaScript 5](#_Toc156090015)

[1.4 PHP 5](#_Toc156090016)

[1.5 MySQL 5](#_Toc156090017)

[2 Практическая часть 6](#_Toc156090018)

[2.1 Требования и описание реализации 6](#_Toc156090019)

[2.2 Общая структура 7](#_Toc156090020)

[2.3 Создание базы данных 8](#_Toc156090021)

[2.4 Регистрация и авторизация в web-приложении 11](#_Toc156090022)

[2.5 Личный кабинет пользователя в web-приложении 16](#_Toc156090023)

[2.5.1 Функционал обычного пользователя 16](#_Toc156090024)

[2.5.2 Функционал администратора 19](#_Toc156090025)

[2.5.3 Функционал кассира 25](#_Toc156090026)

[Заключение 26](#_Toc156090027)

[Список использованных источников 27](#_Toc156090028)

# ВВЕДЕНИЕ

В современном информационном обществе, где цифровые технологии становятся неотъемлемой частью нашей повседневной жизни, веб-программирование играет ключевую роль в трансформации различных отраслей. Одной из сфер, в которой веб-разработка может принести значительные улучшения, является кинематограф. Создание веб-сайта для кинотеатра с возможностью онлайн-бронирования билетов представляет собой важный шаг в развитии современного кинопространства.

Оптимизация процесса бронирования билетов в кинотеатре, позволяя пользователям легко выбирать фильмы, сеансы и места в залах, является актуальным и востребованным решением. Технологии веб-программирования открывают перед кинотеатрами новые возможности в предоставлении удобных и современных услуг для своих посетителей.

В контексте создания базы данных MySQL для веб-приложения, ориентированного на кинотеатр, важность этого шага обусловлена не только потребностью в эффективной организации данных, но и стремлением к повышению качества обслуживания пользователей. Подобно бронированию столиков в ресторане, электронная система бронирования билетов в кинотеатре обладает схожими преимуществами: от упрощения поиска информации до обеспечения безопасности и надежности хранения данных.

В данной работе рассматривается создание веб-сервиса для кинотеатра, предоставляющего возможность бронирования билетов на сеансы. Процесс интеграции веб-технологий, базы данных MySQL и современных принципов веб-дизайна направлен на создание удобного и привлекательного интерфейса для посетителей и администраторов, повышая качество и доступность услуг в мире кино.

Целью курсовой работы является разработка веб-сайта, используя технологии HTML, CSS, JavaScript, PHP, MySQL, по теме «Кинотеатр», с возможностью онлайн-бронирования билетов, а также разработка базы данных для управления пользователями, используя СУБД MySQL.

Задачи исследования включают:

1. Проектирование базы данных: Выделение основных сущностей, их взаимосвязи и структура для эффективного хранения информации о фильмах, сеансах, залах и бронированиях.

2. Верстка веб-страниц: Создание удобного и привлекательного интерфейса с применением HTML и CSS, обеспечивающего простоту взаимодействия для пользователя.

3. Программирование на PHP: Разработка логики обработки запросов, валидации данных и создание форм для взаимодействия с пользователями.

4. Работа с базой данных: Создание SQL-запросов для взаимодействия с базой данных MySQL, обеспечивающих эффективное сохранение и извлечение данных.

5. Функциональность для пользователей: Реализация возможности просмотра расписания сеансов, выбора мест и бронирования билетов.

6. Функциональность для администратора: Разработка средств добавления/удаления сеансов и пользователей через административный интерфейс.

Объектом исследования является web-приложение с применением различных web-технологий.

Предметом исследования является процесс создания web-приложения, базы данных.

Методами иcследования являются: методы анализа и синтеза информации, проектирование, практическая реализация.

# 1 Теоретическая часть

## HTML

HTML (HyperText Markup Language) — это стандартный язык разметки, используемый для создания и структурирования веб-страниц. Он представляет собой набор тегов и атрибутов, которые определяют структуру документа и содержимое на веб-сайтах.

HTML позволяет создавать структурированный документ, разделяя его на различные разделы, такие как заголовки, параграфы, списки и другие элементы. С помощью тегов <a> HTML создает гиперссылки, позволяющие пользователям переходить между страницами или ресурсами в интернете. С использованием тегов <form> и <input>, HTML создает интерактивные формы, позволяющие пользователям взаимодействовать с веб-сайтом, отправлять данные на сервер и выполнять другие действия. HTML поддерживает вставку изображений, аудио и видео файлов, обогащая визуальное и звуковое содержимое веб-страниц, а также обеспечивает создание табличных данных с помощью тегов <table>, упорядоченных (<ol>) и неупорядоченных (<ul>) списков.

Комбинирование с другими технологиями:

CSS (Cascading Style Sheets): HTML часто комбинируется с CSS для стилизации и визуального оформления веб-страниц. CSS определяет цвета, шрифты, расположение элементов и другие аспекты внешнего вида.

JavaScript: для создания интерактивности на веб-сайтах HTML используется с JavaScript. JavaScript добавляет динамическое поведение, обработку событий и многие другие функции.

PHP (Hypertext Preprocessor) и другие серверные языки: HTML работает с серверными языками программирования, такими как PHP, для создания динамических веб-страниц и взаимодействия с базами данных.

Базы данных (SQL и MySQL): HTML используется с языком SQL для взаимодействия с базами данных, хранящими информацию, которая отображается на веб-сайте.

HTML является фундаментальным инструментом для создания веб-страниц и обеспечивает основу для дальнейшего расширения функциональности с использованием других технологий.

## CSS

CSS (Cascading Style Sheets) — это язык стилей, используемый для оформления и визуального улучшения веб-страниц. CSS определяет стилизацию HTML-документов, включая цвета, шрифты, расположение элементов, размеры и другие аспекты представления.

Применение HTML и CSS в веб-разработке является критически важным. Вот несколько причин, почему:

1. Структура и семантика: HTML помогает определить структуру и логическую семантику веб-страницы. Это делает содержимое более доступным для поисковых систем и улучшает взаимодействие со скринридерами для людей с ограниченными возможностями.

2. Внешний вид и пользовательский опыт: CSS позволяет разработчикам создавать привлекательные и стильные веб-страницы. Он дает возможность настраивать внешний вид элементов HTML, что позволяет создавать индивидуальный и уникальный дизайн в соответствии с брендингом и предпочтениями пользователей. Это важно для создания положительного пользовательского опыта.

3. Адаптивность и отзывчивость: HTML и CSS позволяют создавать адаптивные и отзывчивые веб-страницы. Это означает, что веб-сайты могут автоматически адаптироваться к различным устройствам и экранам, таким как мобильные телефоны, планшеты и настольные компьютеры. Это важно для обеспечения удобства использования и достижения большей аудитории.

4. Разделение ответственности: HTML отвечает за структуру и содержание веб-страницы, а CSS за внешний вид. Это разделение позволяет разработчикам более эффективно работать в команде, улучшает обслуживание и повторное использование кода.

В целом, применение HTML и CSS в веб-разработке является неотъемлемой частью создания высококачественных и стильных веб-страниц. Они обеспечивают важные функции, которые улучшают доступность, пользовательский опыт и удобство использования веб-сайтов.

## JavaScript

JavaScript — это высокоуровневый, интерпретируемый язык программирования, который применяется веб-разработкой для создания динамических и интерактивных веб-страниц. Он поддерживает объектно-ориентированную, императивную и функциональную парадигмы программирования.

Назначение и преимущества языка:

1. JavaScript используется для придания веб-страницам интерактивности, позволяя пользователям взаимодействовать с содержимым страницы.

2. Язык позволяет динамически обновлять содержимое страницы без необходимости перезагрузки. Это особенно важно для одностраничных приложений (SPA).

3. JavaScript обрабатывает события, такие как клики, наведение, отправка форм, что позволяет создавать отзывчивые веб-приложения.

4. Язык предоставляет средства для манипуляции Document Object Model (DOM), т.е., изменения структуры и стилей веб-страницы.

5. JavaScript используется для выполнения асинхронных запросов к серверу (AJAX), обеспечивая обновление данных на странице без полной перезагрузки.

6. Интеграция с HTML и CSS: легко встраивается в веб-страницы, взаимодействует с HTML и CSS для создания полноценных веб-приложений.

JavaScript остается ключевым языком веб-разработки, обеспечивая возможность создания динамичных и интерактивных веб-приложений, а его использование не ограничивается только веб-браузерами.

## 1.4 PHP

PHP (Hypertext Preprocessor) представляет собой универсальный язык программирования, широко применяемый для создания динамических веб-страниц и веб-приложений. Этот скриптовый язык исполняется на стороне сервера, что означает, что код PHP обрабатывается на сервере, а результаты выводятся на клиентском устройстве.

Применение PHP в веб-разработке обосновывается следующими причинами:

1. Простой и понятный синтаксис PHP делает его доступным для начинающих разработчиков. Обширное сообщество PHP-разработчиков также гарантирует наличие обучающих материалов и поддержки.

2. PHP поддерживает разнообразные базы данных (MySQL, PostgreSQL, Oracle и другие) и протоколы (HTTP, FTP, SMTP и др.), обеспечивая создание различных веб-приложений.

3. Лёгкая интеграция PHP с другими технологиями, такими как HTML, CSS, JavaScript, позволяет создавать сложные веб-приложения и веб-сайты.

4. PHP является проектом с открытым исходным кодом, позволяя разработчикам свободно использовать и модифицировать его под свои нужды. Это также обеспечивает наличие обширной базы бесплатных расширений и фреймворков для PHP.

Функции PHP включают:

1. Генерация динамического контента: PHP обеспечивает создание динамических страниц, автоматически адаптирующихся к действиям пользователя.

2. Работа с формами: Легкая обработка данных, полученных из форм на веб-страницах, позволяет создавать интерактивные приложения, такие как регистрация пользователей и отправка сообщений.

3. Работа с базами данных: Мощные функции PHP обеспечивают удобную работу с базами данных, включая хранение, изменение и извлечение данных.

4. Организация и обработка файлов: PHP предоставляет средства для работы с файлами на сервере, включая чтение, запись и удаление данных.

5. Обработка ошибок и исключений: PHP дает разработчикам контроль над обработкой ошибок и исключений, что способствует созданию более надежных и безопасных приложений.

В итоге PHP остается мощным инструментом для разработки веб-приложений, привлекая внимание своей простотой использования и обширным набором возможностей.

## MySQL

MySQL - это система управления базами данных, которая позволяет организовывать и управлять большими объемами данных. Это одна из наиболее популярных реляционных СУБД, используемых для работы с веб-приложениями.

MySQL следует модели реляционных баз данных, что позволяет эффективно хранить и управлять данными в виде таблиц, связанных между собой. Как и PHP, MySQL является проектом с открытым исходным кодом. Это означает, что разработчики могут свободно использовать, изменять и распространять его без ограничений. MySQL обеспечивает быстрый доступ к данным и поддерживает многопользовательские запросы, что делает его идеальным для веб-приложений с большим количеством пользователей. Система спроектирована так, чтобы легко масштабироваться, что важно для веб-приложений, требующих расширенной базы данных при росте. MySQL обеспечивает множество средств для защиты данных, включая шифрование, аутентификацию и управление доступом, что делает его надежным выбором для хранения чувствительной информации. MySQL поддерживает транзакции, обеспечивая целостность данных даже в случае сбоев или ошибок.

В контексте разработки веб-приложения для бронирования мест в кинотеатре, MySQL становится ключевым компонентом по следующим причинам:

1. MySQL предоставляет эффективные средства для создания таблиц, содержащих данные о фильмах, сеансах, залах и других аспектах кинотеатрального процесса.

2. База данных MySQL позволяет хранить информацию о бронированиях, включая выбранные места, дату и время бронирования, а также данные о посетителях.

3. В контексте бронирования мест в кинотеатре, где одновременно может происходить множество транзакций бронирования, многопользовательская поддержка MySQL становится важным фактором для обеспечения эффективности системы.

4. Возможность определения связей между таблицами в MySQL позволяет эффективно связывать данные и обеспечивать согласованность при работе с различными аспектами системы бронирования.

Таким образом, использование MySQL для выбора фильмов и бронирования мест в кинотеатре позволяет эффективно управлять и обрабатывать данные, обеспечивает безопасность хранимых данных и способствует развитию функционала системы.

# 2 Практическая часть

## 2.1 Требования и описание реализации

Курсовой проект предполагает создание web-приложение на тему «Кинотеатр», используя технологии HTML, CSS, Javascript, PHP, MySQL.

Описание реализации:

- неавторизированные пользователи должны иметь возможность регистрироваться (форма добавления пользователя)

- авторизированные пользователи разделить на 2 группы: администраторы и авторизированные пользователи.

- администраторы имеют право просматривать, добавлять, редактировать и удалять пользователей.

- авторизированные пользователи имеют право просматривать и редактировать только свои данные.

- при регистрации пользователю присваивается статус авторизированного пользователя.

Требования к проекту:

1. Должна быть предусмотрена авторизация и регистрация пользователей.
2. Зарегистрированный пользователь имеет право редактировать свои данные и просматривать свои услуги.
3. Администраторы имеют право редактировать пользователей и добавлять услуги.
4. Неавторизированный пользователи имеют право просматривать список услуг
5. Сайт должен быть выдержан в определенном стиле и корректно открываться на любом разрешении.
6. Общий функционал может быть продуман самостоятельно.
7. Важно продемонстрировать не большой объем функционала, а минимальные требования к проекту. Проект может максимально простым, но при этом содержать все необходимые функции.
8. Отдельные бонусы предусмотрены за доп. функционал и красивый дизайн.

## 2.2 Общая структура

Разработка производилась на готовом веб-сервере на основе сборки под Windows – XAMPP. Разработка и тестирование разработанного сайта велись на PHP версии 7.4.5. Скачиваем данную программу и после скачивания приложения необходимо установить программу. При этом важно, чтобы название папки не содержало русских символов и пробелов.

После установки необходимо запустить XAMPP start и XAMPP control. Далее нажать на кнопки Start для Apache и для MySQL (Рисунок 1).

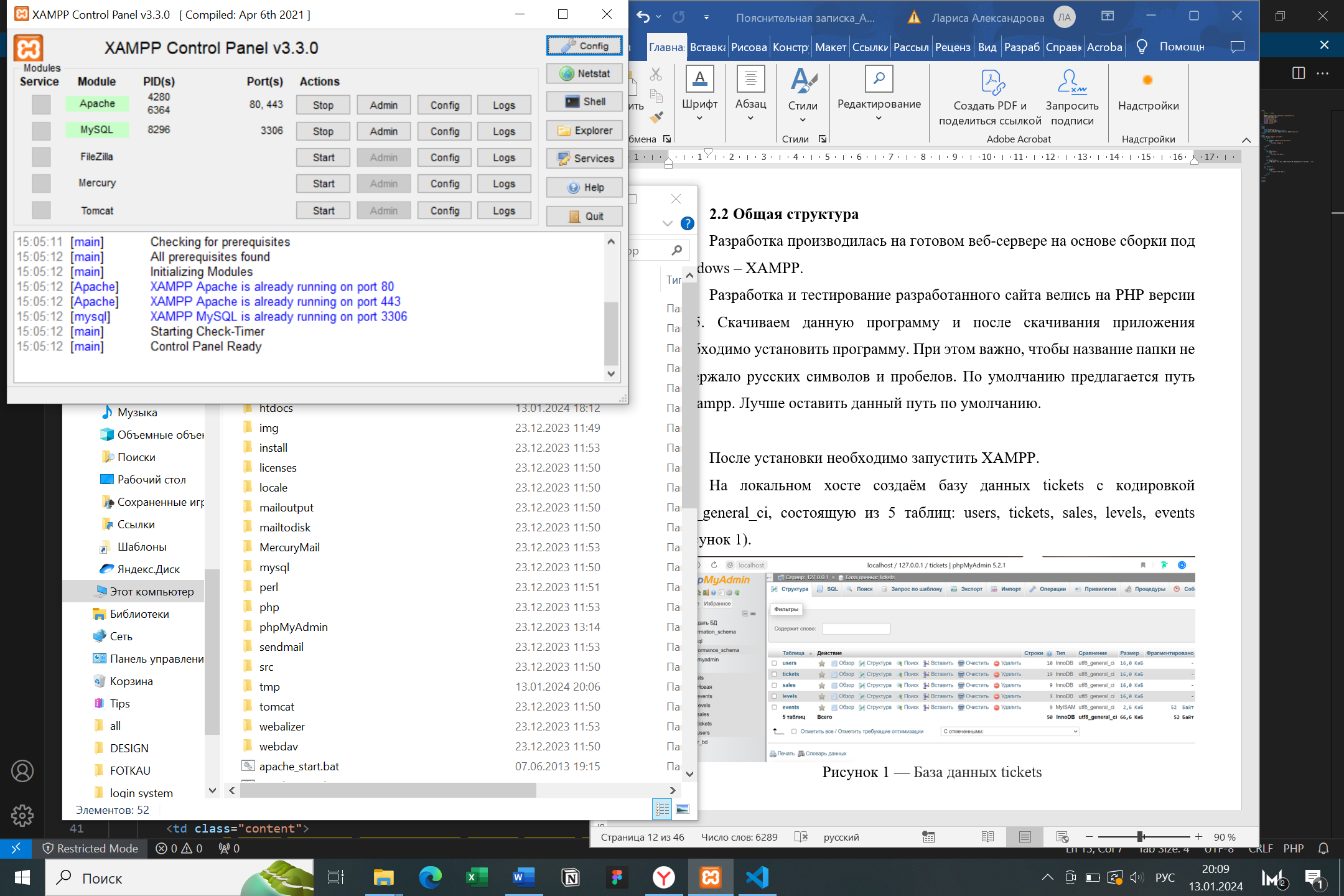


Рисунок 1 — Запуск XAMPP

На локальном хосте создаём базу данных tickets с кодировкой utf8\_general\_ci, состоящую из 5 таблиц: users, tickets, sales, levels, events (Рисунок 2).

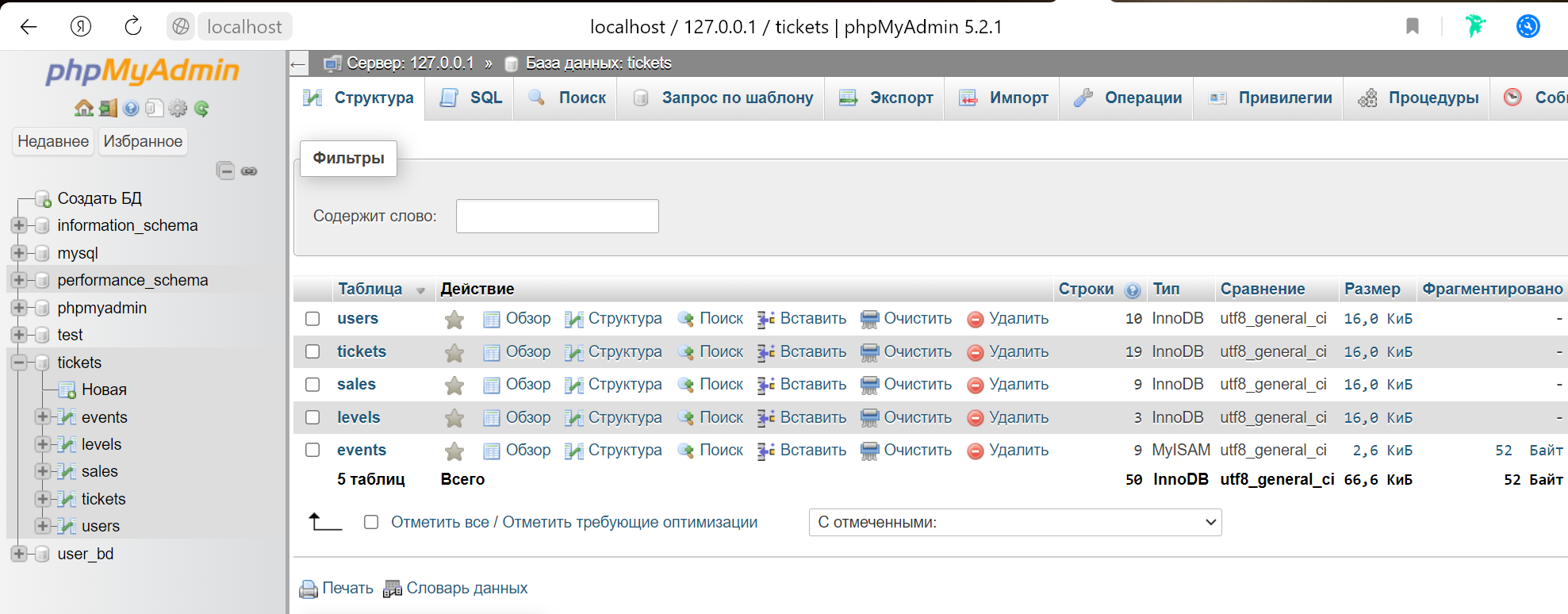


Рисунок 2 — База данных tickets

Далее была установлена система управления базами данных MySQL, которая позволяет организовывать и управлять большими объемами данных (Рисунок 3).

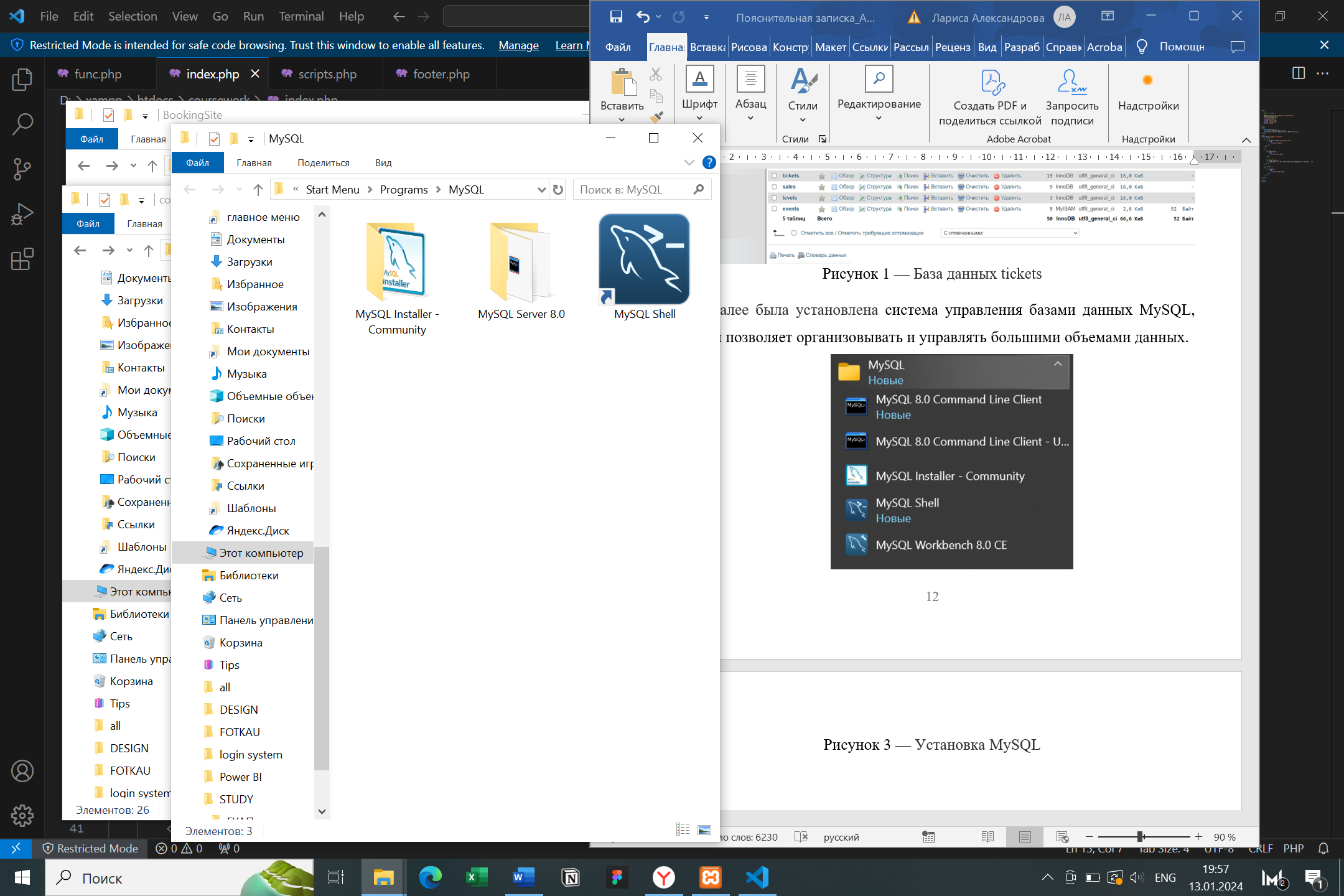


Рисунок 3 — Установка MySQL

## 2.3 Создание базы данных

Для создания базы данных необходимо на главной странице программы phpMyAdmin в поле «создать новую БД» ввести имя базы данных и нажать кнопку «Создать». Создадим базу данных tickets с кодировкой utf8\_general\_ci. Создаётся таблица пользователей users (Рисунок 4), которая содержит в себе id, fio, level (уровень доступа), password, login, email, discount\_value (индивидуальная скидка для пользователя в процентах). Примечание: логин должен быть уникальным, зарегистрировать двух пользователей с одинаковым логином невозможно. 

Рисунок 4 — Создание полей таблицы users

Далее создаётся таблица билетов tickets (Рисунок 5), которая содержит в себе user\_id (код пользователя), event\_id (код фильма), num (номер места).

Примечание: при регистрации новому пользователю по умолчанию присваивается уровень доступа 1, он становится просто пользователем. Поменять ему уровень доступа на 5 (кассир) или 10 (админ) может только администратор.

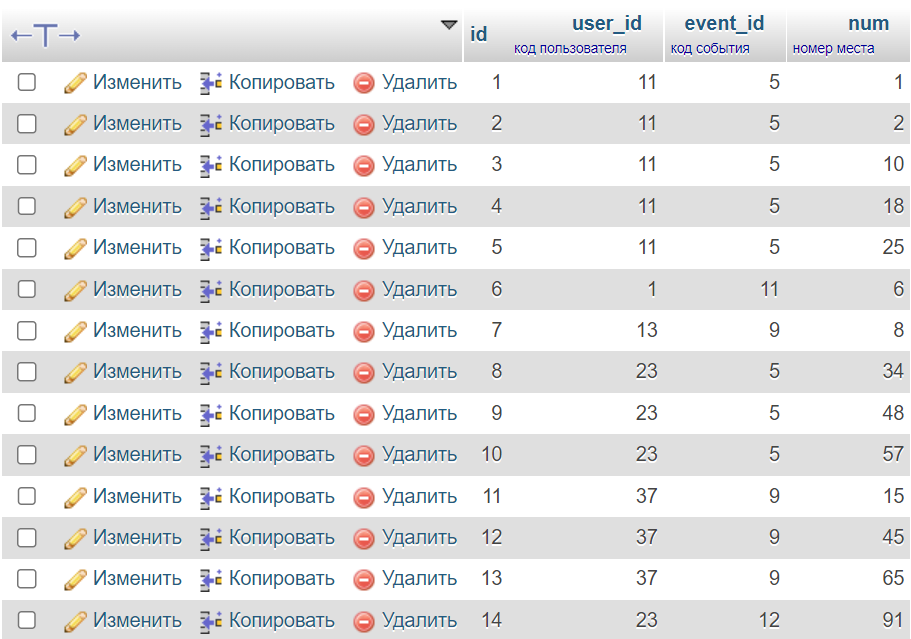


Рисунок 5 — Создание полей таблицы tickets

Примечание: если место уже занято, оно будет некликабельным и подсвечиваться красным цветом при наведении, если место свободно, будет подсвечиваться зеленым (Рисунок 6). Можно выбрать несколько мест.

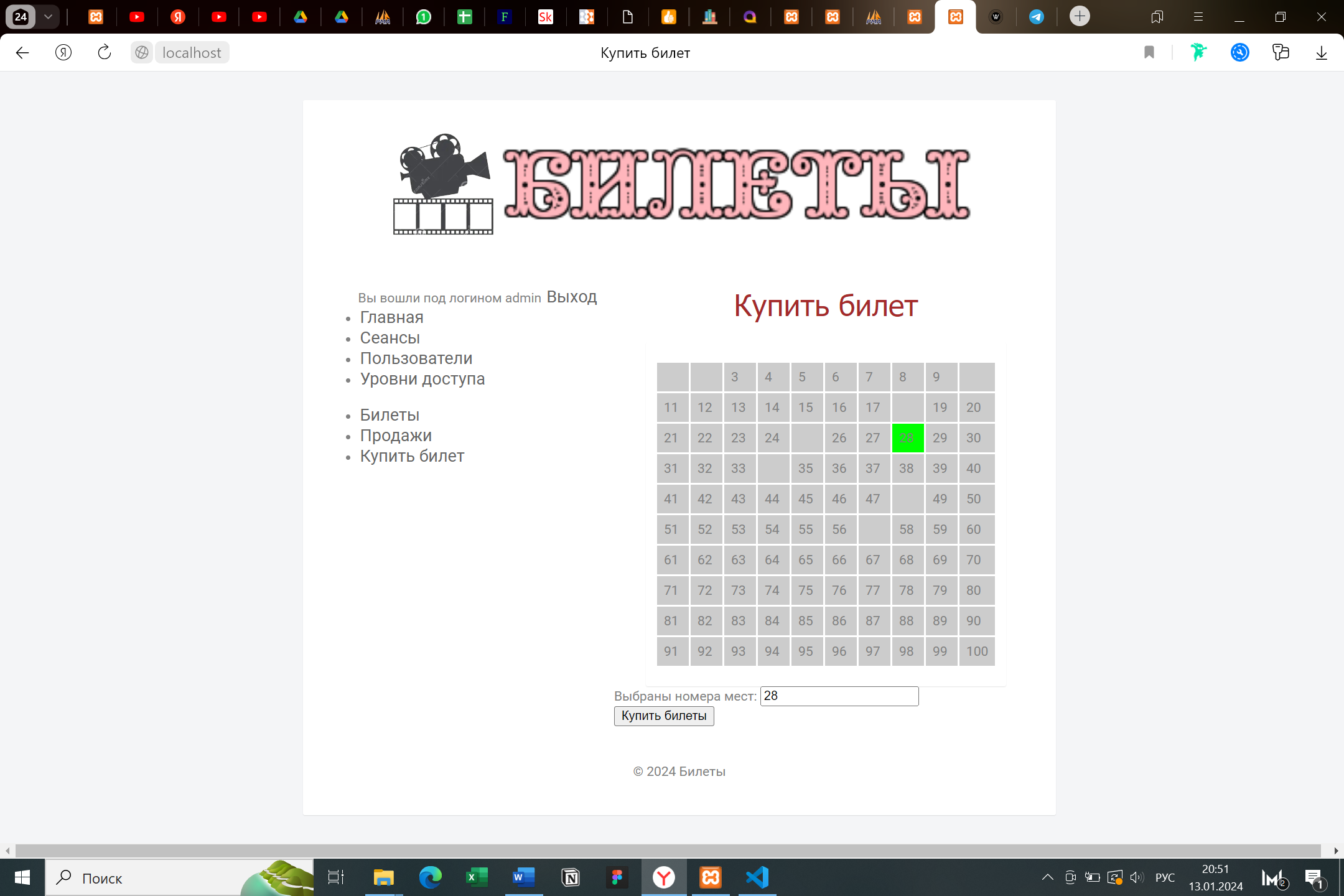


Рисунок 6 — Выделение цветом занятых и свободных мест

После создаётся таблица скидок (Рисунок 7), которая показывает сумму за билеты и учитывает скидку, если она есть (содержит в себе user\_id, event\_id, sum).

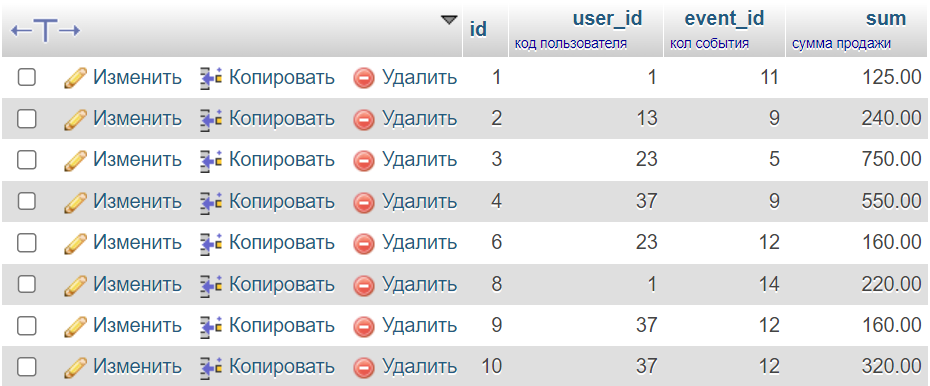


Рисунок 7 — Создание полей таблицы sales

Далее создаётся таблица уровней доступа (Рисунок 8), которая содержит в себе id (1/5/10), name (пользователь/кассир/админ). У этих групп разные полномочия: пользователь может купить и просмотреть только свои билеты, кассир может покупать билеты для любого клиента и для себя, а администратор может редактировать и удалять всё, включая пользователей, купленные билеты, сеансы, уровень доступа пользователей.

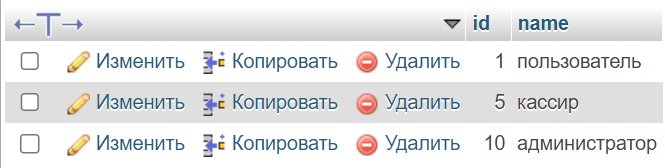


Рисунок 8 — Создание полей таблицы levels

Далее создаётся таблица сеансов кинотеатра (Рисунок 9), которая содержащая дату, название фильма, краткое описание и стоимость билета.

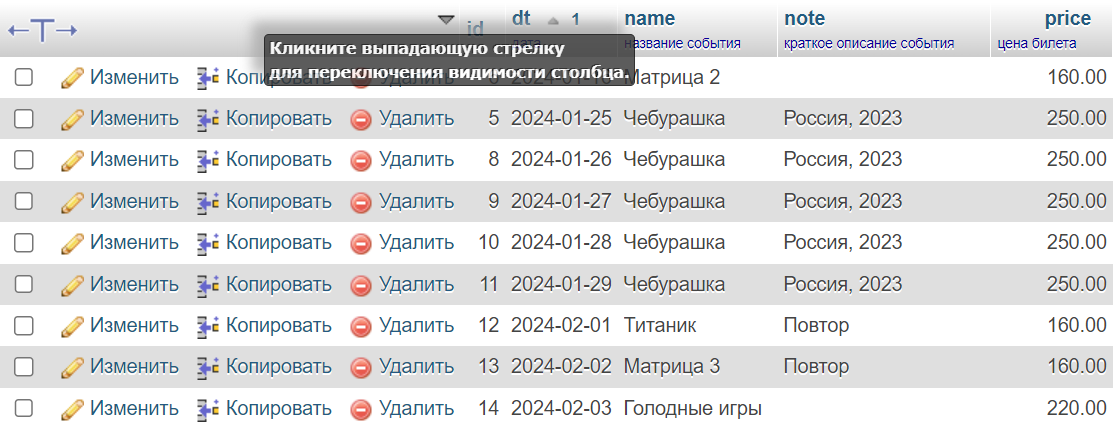


Рисунок 9 — Создание полей таблицы events

## 2.4 Регистрация и авторизация в web-приложении

Чтобы открыть сайт в адресной строке браузера ввожу путь, где находятся мои файлы: http://localhost/coursework/index.php. При входе на сайт неавторизированные пользователи имеют право просматривать сеансы, регистрироваться и авторизироваться. На рисунке 10 представлена главная страница сайта неавторизированного пользователя.

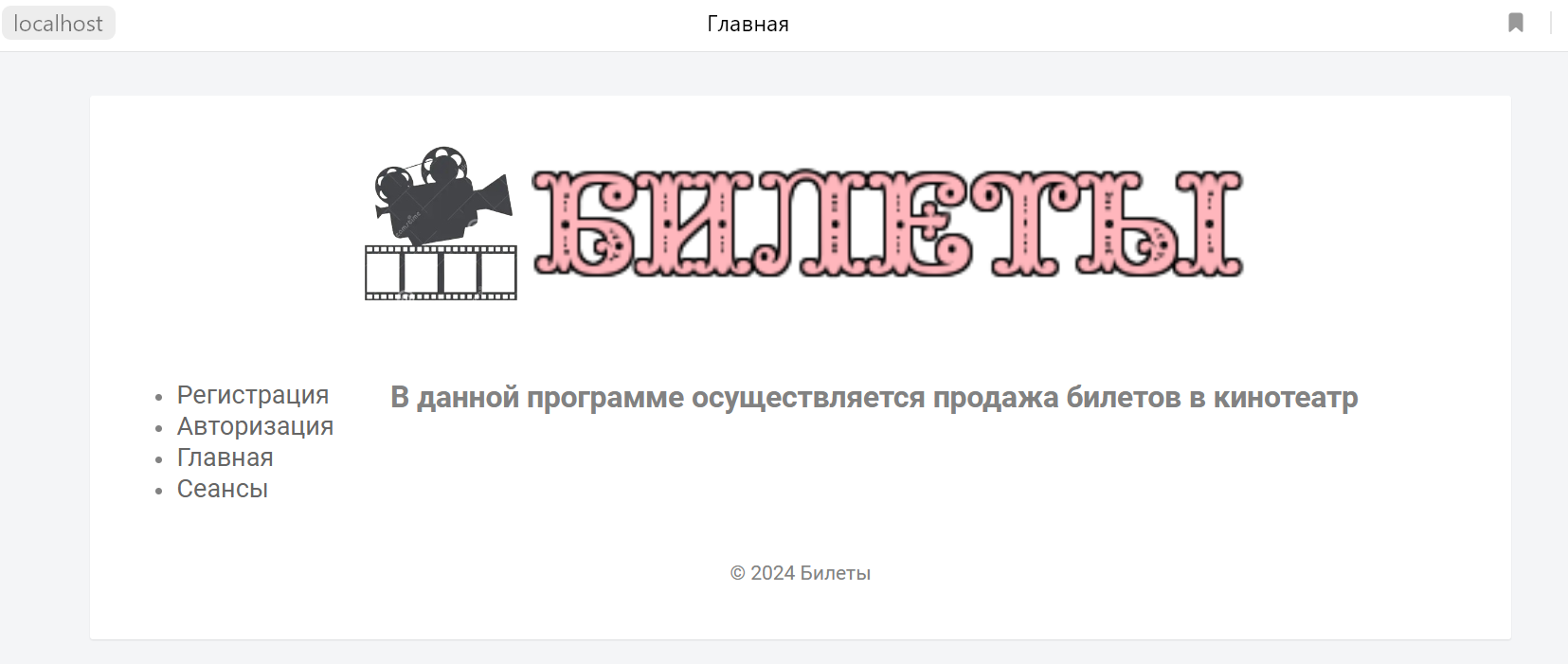


Рисунок 10 — Главная страница сайта неавторизированного пользователя

Для красивого внешнего вида сайта используются css файлы, а также в файле top.php картинка с логотипом и надпись «Билеты».

В файле menu.php, действующем на всех страницах сайта, реализованы такие функции:

Создание HTML-разметки для меню с использованием неупорядоченного списка (<ul>). Если у пользователя нет уровня ($\_SESSION['level'] не установлен), добавляются ссылки на страницу регистрации (reg.php) и страницу авторизации (login.php). Если пользователь авторизован, отображается его логин и ссылка для выхода (?do=exit).

Добавление общедоступных пунктов меню, таких как "Главная" (index.php) и "Сеансы" (events.php).

Добавление пунктов меню, доступных пользователям с различными уровнями доступа (администраторам, пользователям с уровнем 10, 5, 1), такие как "Пользователи" (users.php), "Уровни доступа" (levels.php), "Билеты" (tickets.php), "Продажи" (sales.php), и "Купить билет" (pay.php).

Для визуально красивого отображения сайта подключены файлы css для разных страниц. Пример листинга стилей для меню – файл menu\_style.css:

.rounded {

list-style: none;

font: 14px "Trebuchet MS", "Lucida Sans";

padding: 15px;

text-shadow: 0 1px 0 rgba(255,255,255,.5);

}

.rounded a {

position: relative;

display: block;

padding: .4em .4em .4em 2em;

margin: .5em 0;

background: #DAD2CA;

color: #444;

text-decoration: none;

border-radius: .3em;

transition: .3s ease-out;

}

.rounded a:hover {background: #E9E4E0;}

.rounded a:hover:before {transform: rotate(360deg);}

.rounded a:before {

position: absolute;

left: -1.3em;

top: 50%;

margin-top: -1.3em;

background: #8FD4C1;

height: 2em;

width: 2em;

line-height: 2em;

border: .3em solid white;

text-align: center;

font-weight: bold;

border-radius: 2em;

transition: all .3s ease-out;

}

body {

font-family: Arial, Verdana, sans-serif; /\* Семейство шрифтов \*/

font-size: 11pt; /\* Размер основного шрифта в пунктах \*/

background-color: #f0f0f0; /\* Цвет фона веб-страницы \*/

color: #333; /\* Цвет основного текста \*/

}

h1 {

color: #a52a2a; /\* Цвет заголовка \*/

font-size: 24pt; /\* Размер шрифта в пунктах \*/

font-family: Tahoma, Georgia, Times, serif; /\* Семейство шрифтов \*/

font-weight: normal; /\* Нормальное начертание текста \*/

text-align:center;

}

p {

text-align: justify; /\* Выравнивание по ширине \*/

padding:10px;

}

.menu2 {

width:33%;

}

.content {

width:66%;

}

Форма регистрации содержит в себе поля для ввода фамилии и имени, логина, пароля и электронной почты, а также кнопки «Очистить поля» и «Сохранить» (Рисунок 11). Все поля, кроме email обязательны для заполнения. В поле «Логин» реализована проверка его уникальности для того, чтобы код не сохранил один и тот же логин дважды. Если логин уже занят, система выдаст оповещение. В поле «Email» есть проверка на правильность ввода (наличие «@»).

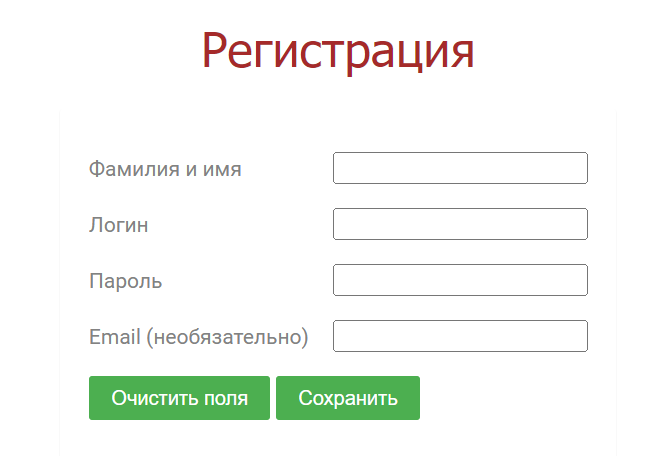


Рисунок 11 — Регистрация в web-приложении

Файл reg.php выполняет следующие функции:

Включение отображения всех ошибок с помощью error\_reporting(E\_ALL). Подключение файлов auth.php, database.php, func.php, styles.php и scripts.php для использования в коде. Заголовок страницы и внешние стили и скрипты. Основная разметка HTML с тегами <head> и <body>. Верстка таблицы с баннером, меню и контентом.

Обработка регистрации пользователя: если отправлена форма ($\_POST['login'] и $\_POST['password'] не пусты): извлечение и обработка входных данных. Проверка наличия пользователя с таким логином в базе данных. Если пользователь существует, вывод сообщения об ошибке. В противном случае, выполнение запроса на добавление нового пользователя.

Если кнопка "Сохранить" была нажата ($\_POST['btn\_submit'] не пуста), вывод предупреждения о необходимости ввода логина и пароля.

После успешной регистрации выводится соответствующее оповещение и возможность перейти на страницу авторизации (Рисунок 12).

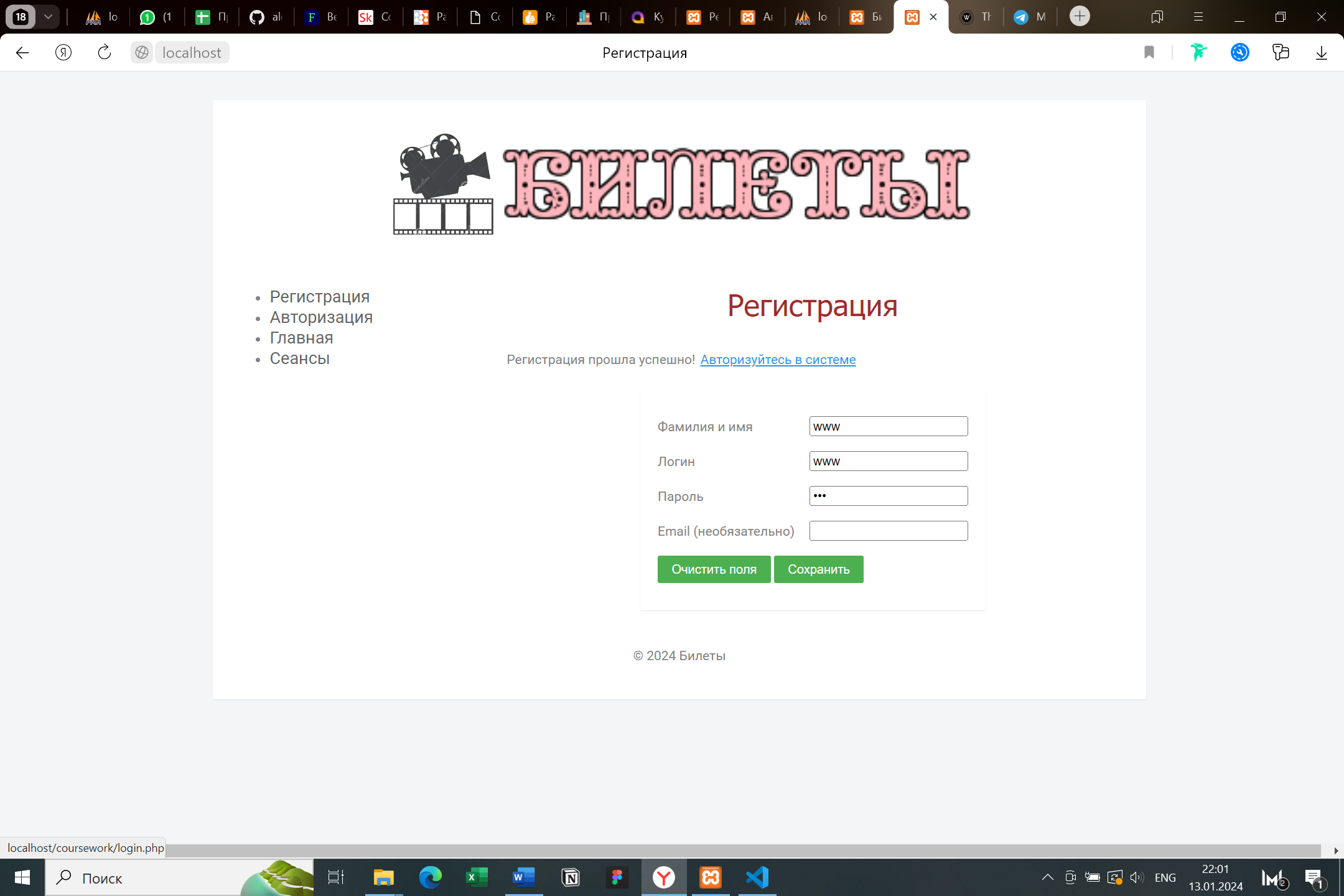


Рисунок 12 — Успешная регистрация

Форма для входа содержит в себе поля для ввода логина и пароля и кнопку «Войти» (Рисунок 13). Если пользователь ещё не зарегистрирован, то при клике ниже на «Регистрация» его перебросит на соответствующую страницу.

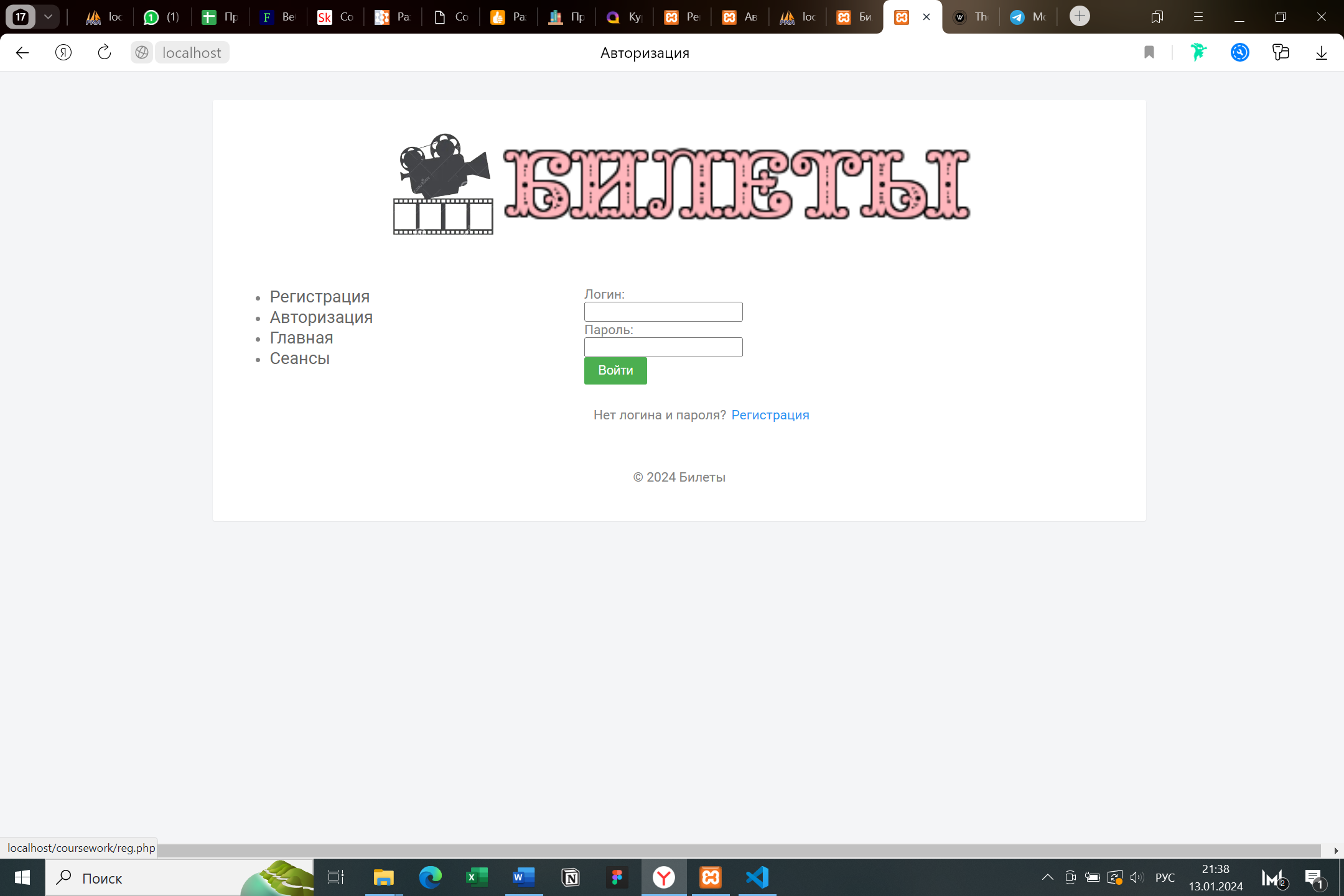


Рисунок 13 — Вход в web-приложение

При отправке формы активируется PHP код – файл database.php, устанавливается подключение к базе данных (подключение к локальному хосту). Переменные для подключения:

$hostname: Хост базы данных (в данном случае, локальный сервер);

$mysql\_login: Логин пользователя для подключения к базе данных;

$mysql\_password: Пароль пользователя для подключения к базе данных;

$database: Название базы данных.

При проверке введенных данных, если данные неверны, система оповестит о том, что логин или пароль введён неверно. Пароли в данном случае в базе данных не хешируются, чтобы было проще войти в профили разных пользователей и проверить функционал сайта для курсовой работы. В случае разработки реального сайта, пароли обязательно нужно хешировать для безопасности пользователей.

Далее реализована проверка уникальности логина для того, чтобы код не сохранил один и тот же логин дважды. Если логин уже занята, система выдаст оповещение. Листинг:

$res=mysqli\_query($con, $query) or die(mysqli\_error($con));

if (mysqli\_fetch\_array($res, MYSQLI\_BOTH)[0]) {

echo '<p>Пользователь с таким логином уже существует! </p>';

Файл auth.php выполняет следующие функции:

Проверка, существует ли текущая сессия (if (session\_id() == '') session\_start();). Если сессии нет, то она создается.

Проверка наличия параметра do в запросе и его значения exit. Если параметр и его значение обнаружены, пользователь выходит из системы (logout).

При выходе из системы удаляются данные о логине из сессии (unset($\_SESSION['login']);) и сама сессия уничтожается (session\_destroy();).

После завершения выхода из системы, пользователь перенаправляется на страницу index.php.

Файл login.php выполняет следующие функции:

Включение файлов styles.php и scripts.php. Установка заголовка страницы с указанием кодировки UTF-8. Установка уровня отображения ошибок. Включение файла database.php для подключения к базе данных. Включение файла func.php для использования вспомогательных функций. Формирование HTML-разметки страницы с использованием таблицы. Включение баннера, меню, контента и подвала страницы с помощью вспомогательных файлов (top.php, menu.php, и footer.php). Вывод формы авторизации с полями для ввода логина и пароля, а также кнопкой "Войти". Обработка данных, отправленных из формы авторизации методом POST.

Проверка наличия логина и пароля в базе данных. Если соответствие найдено, устанавливаются сессионные переменные ($\_SESSION['login'], $\_SESSION['level'], $\_SESSION['id']), и пользователь перенаправляется на главную страницу (index.php). В случае неудачи выводится сообщение об ошибке.

## 2.5 Личный кабинет пользователя в web-приложении

## 2.5.1 Функционал обычного пользователя

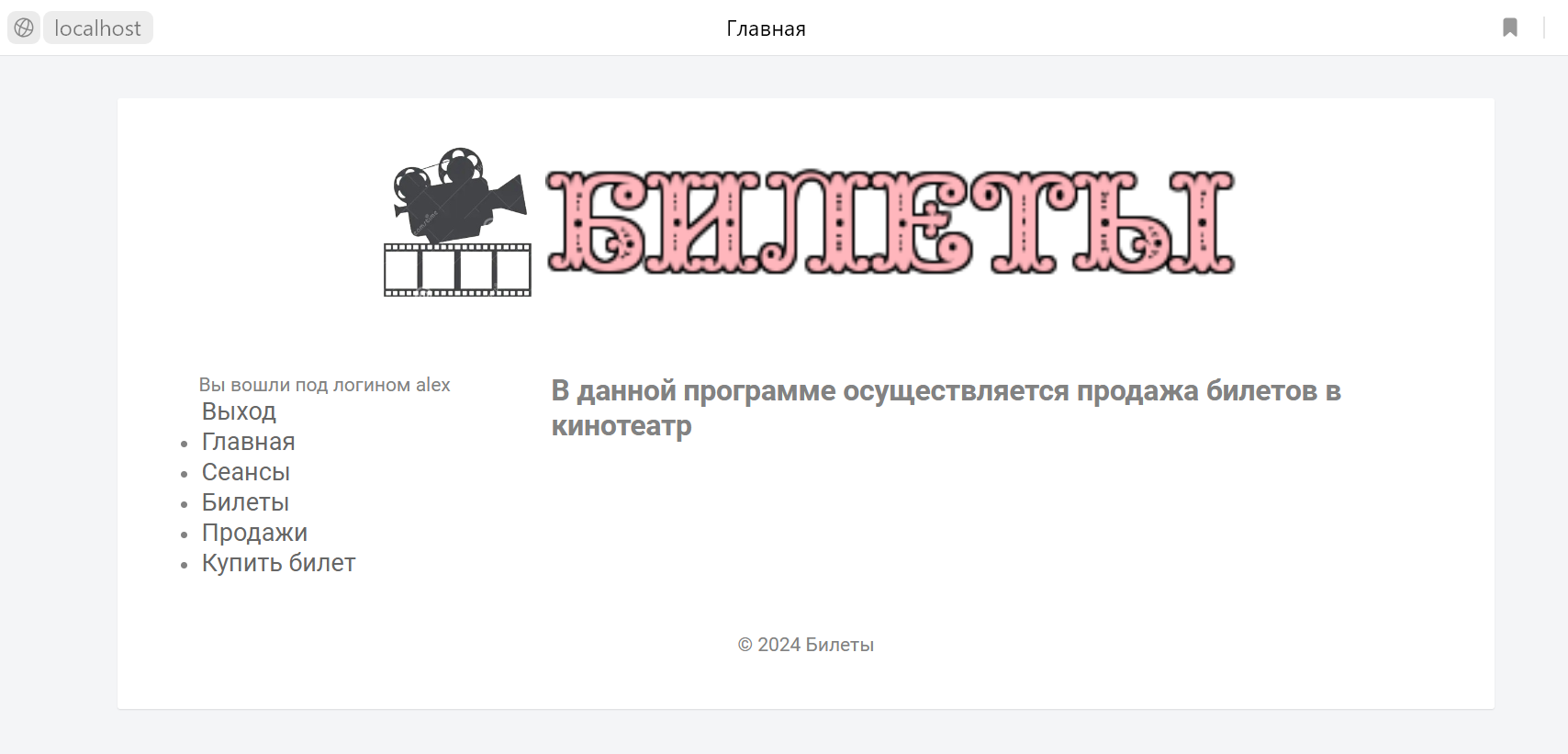


Рисунок 14 — Личный кабинет пользователя в web-приложении

При входе мы получаем информацию об авторизированном пользователе и выводим логин на панель, под которым вошел пользователь:

if ( !isset($\_SESSION['level'])) $menu.='<li> <a href="reg.php">Регистрация</a> </li>';

if ( !isset($\_SESSION['level'])) $menu.='<li> <a href="login.php">Авторизация</a> </li>';

else {

$menu.='Вы вошли под логином '.$\_SESSION['login'].' <a href="?do=exit">Выход</a>'; };

При входе для обычного пользователя доступны следующие пункты меню: выход, главная, сеансы, билеты, продажи, купить билет.

На странице «Сеансы» пользователь видит расписание фильмов (Рисунок 15). На странице «Билеты» может просмотреть билеты, которые он купил: сеанс, номер места, дату (Рисунок 16). На странице «Продажи» увидеть стоимость купленных билетов (Рисунок 17). На странице «Купить билет» будет выпадающий список из доступных сеансов и возможность выбрать нужный сеанс (Рисунок 18). Страница «Подтверждение покупки» после успешного оформления билетов одинакова для всех пользователей. С нее можно перейти на страницу с билетами и их стоимостью (Рисунок 19).



Рисунок 15 — «Сеансы» обычного пользователя



Рисунок 16 — «Билеты» обычного пользователя

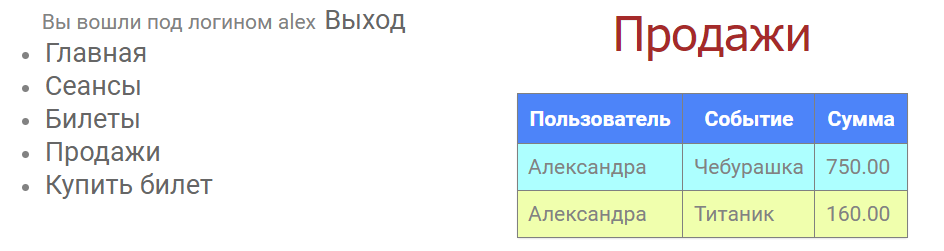


Рисунок 17 — «Продажи» обычного пользователя



Рисунок 18 — «Купить билет» обычного пользователя

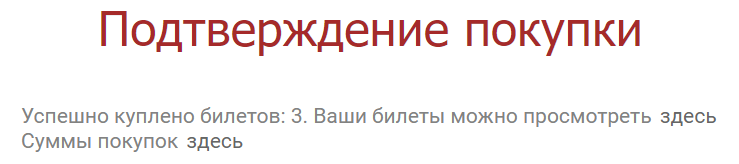


Рисунок 19 — «Подтверждение покупки»

Пример листинга стилей для таблицы:

//Чередуем цвета для строк

if($RowCt++ % 2 == 0) $Style = "background-color: #ADFFFF;";

else $Style = "background-color: #F0FFAD;";

$Table.= "<tr style=\"$Style\">";

//Пройдем по каждому полю

foreach($Row as $field => $value)

{ // делаем подсветку найденного

$value=str\_replace($mask, "<font color='red'>$mask</font>", $value);

// отображаем значение

$Table.= "<td>$value</td>";

## 2.5.2 Функционал администратора

Функционал администратора гораздо шире, помимо элементов меню обычного пользователя, добавляются пункты: пользователи и уровни доступа и доступны дополнительные функции, которые дальше рассмотрим.

Во вкладке «Сеансы» админ имеет возможность удалять записи по запросу с параметром delete\_id и редактировать текущие сеансы, если задан параметр edit\_id. А также добавлять новые записи и сохранять изменения или добавление новой записи в базу данных (Рисунок 20).

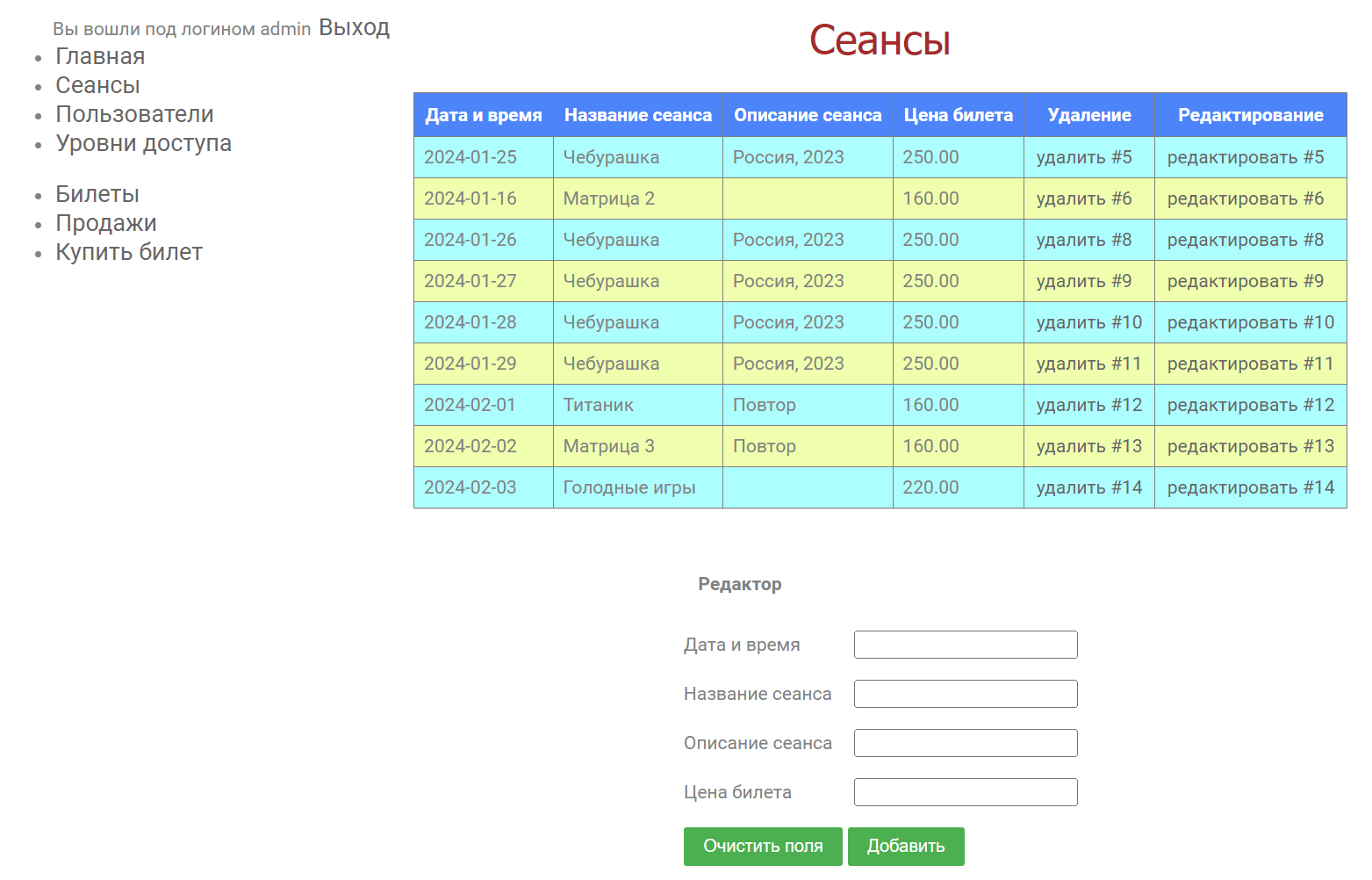


Рисунок 20 — «Сеансы» администратора

Пример листинга из events.php для удаления:

if (!empty($\_GET['delete\_id'])) {

$id=intval($\_GET['delete\_id']);

$query="

DELETE FROM `$table`

WHERE id=$id

";

mysqli\_query($con, $query) or die(mysqli\_error($con)); };

Пример листинга из events.php для редактирования:

if (!empty($\_GET['edit\_id'])) {

$id=intval($\_GET['edit\_id']);

$buf='';

foreach($param\_keys as $param\_key) {

$buf.="`$param\_key`, ";

};

$buf=trim($buf, ', ');

$query="

SELECT

$buf

FROM `$table`

WHERE id=$id

";

$res=mysqli\_query($con, $query) or die(mysqli\_error($con));

$row=mysqli\_fetch\_array($res);

foreach($param\_keys as $param\_key) {

$param\_values[$param\_key]=$row[$param\_key];};

};

Во вкладке «Пользователи» админ имеет возможность удалять и редактировать данные всех зарегистрированных пользователей и обновлять информацию в базе. А также может добавить нового пользователя, присвоить ему уровень доступа, и индивидуальную скидку (Рисунок 21).

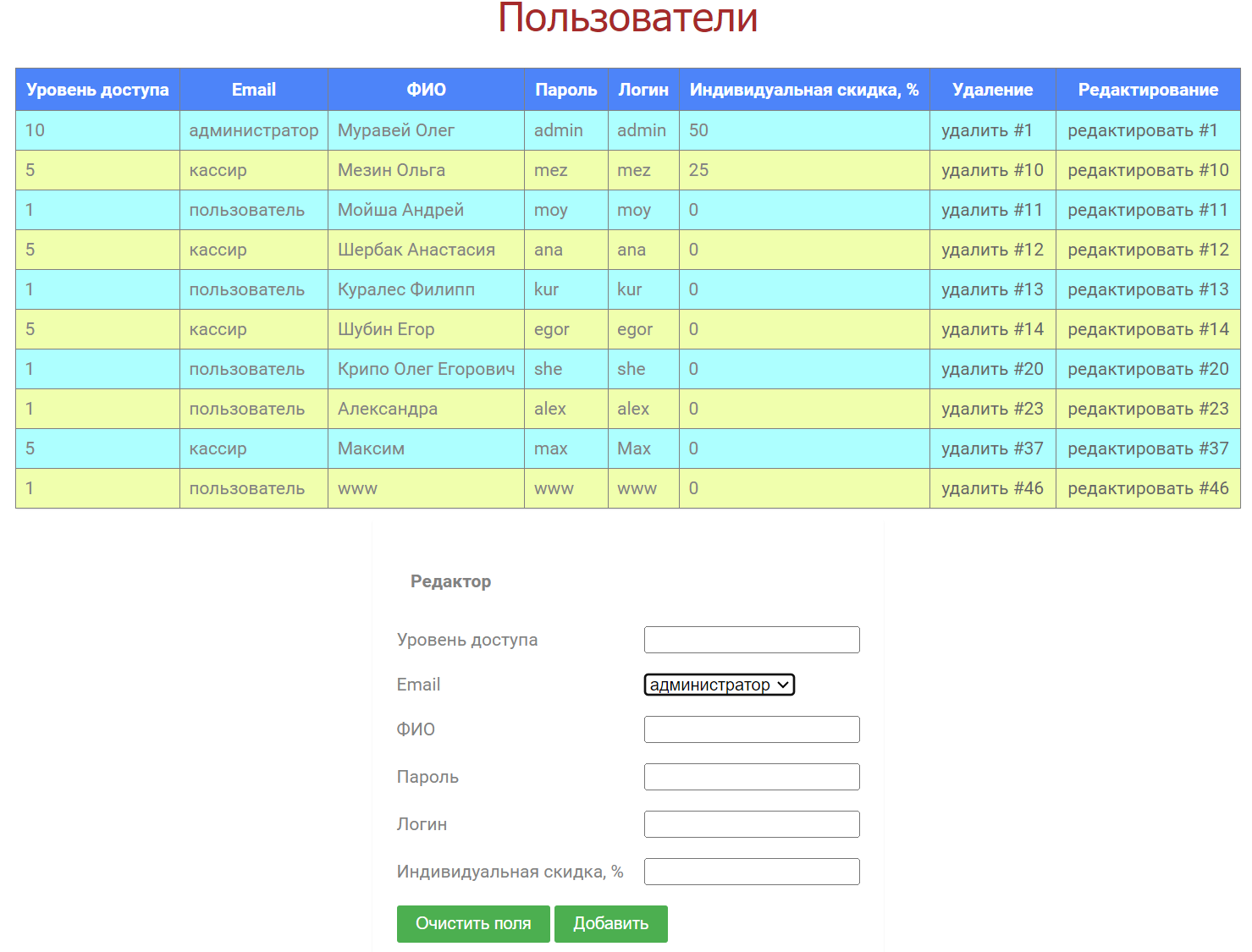


Рисунок 21 — Вкладка «Пользователи»

Пример листинга из файла users.php:

if (isset($\_POST['btn\_submit'])) // была нажата кнопка сохранить - не надо больше отображать id

$id=0;

// добавляем возможность удаления админам

$delete\_confirm="onClick=\"return window.confirm(\'Подтверждаете удаление?\');\"";

$admin\_delete=$edit ? ", CONCAT('<a href=\"$table.php?delete\_id=', `$table`.id, '\" $delete\_confirm>', 'удалить&nbsp;#', `$table`.id, '</a>') AS 'Удаление'" : '';

// добавляем возможность редактирования админам

$admin\_edit=$edit ? ", CONCAT('<a href=\"$table.php?edit\_id=', `$table`.id, '\">', 'редактировать&nbsp;#', `$table`.id, '</a>') AS 'Редактирование'" : '';

$buf='';

for($ind=0; $ind<count($param\_keys); $ind++) {

if (!empty($param\_ext[$ind]))

$buf.=$param\_ext[$ind]." AS '".$param\_str[$ind]."', ";

else

$buf.="`$table`.`".$param\_keys[$ind]."` AS '".$param\_str[$ind]."', ";

};

$buf=trim($buf, ', ');

$query="

SELECT

$buf

$admin\_delete

$admin\_edit

FROM

`$table`, `levels`

WHERE 1

AND `$table`.`level`=`levels`.`id`

ORDER BY `$table`.`id`

";

echo SQLResultTable($query, $con, '');

Во вкладке «Билеты» админ имеет возможность удалять или редактировать записи всех пользователей и обновлять информацию в базе. А также может добавить нового пользователя и купить ему билет (Рисунок 22).

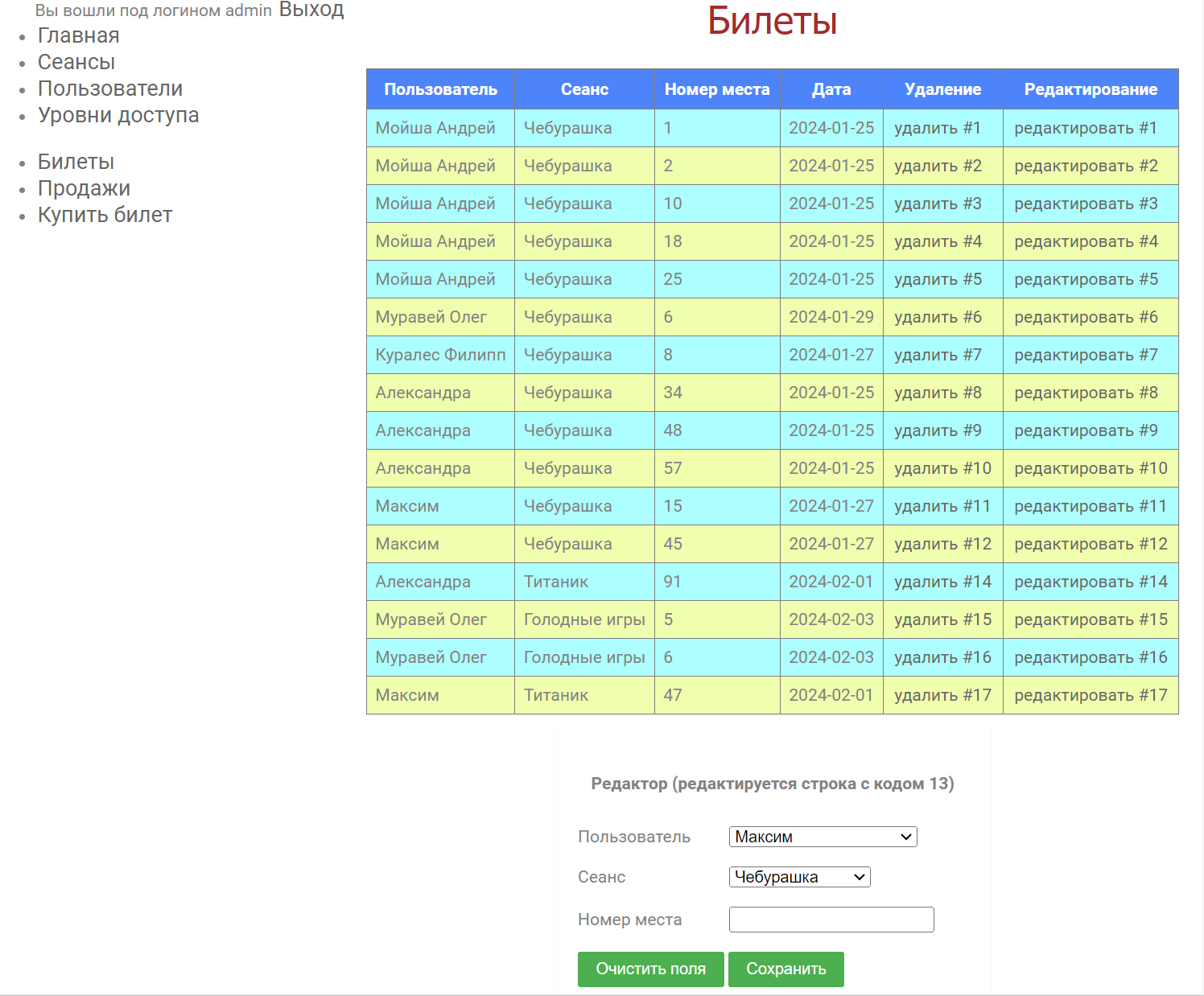


Рисунок 22 — Вкладка «Билеты» администратора

Пример листинга из файла users.php на подтверждения удаления:

$delete\_confirm="onClick=\"return window.confirm(\'Подтверждаете удаление?\');\"";

Во вкладке «Купить билет» админ может купить билет со скидкой 50%, которая будет применена при расчете стоимости (тк в базе данных я указала данный процент скидки). Закрепить скидку за пользователем может администратор для любого пользователя, по дефолту она устанавливается 0%.

Листинг из файла confirm\_pay.php:

$nums=htmlentities(trim($\_POST['nums']), ENT\_QUOTES, 'UTF-8');

$nums=array\_unique(explode(',', $nums));

if (count($nums)==0) {

// ничего не выбрано

}

else {

$cnt=0; // количество продаваемых пользователю билетов

foreach($nums as $num) {

$query="

INSERT INTO tickets

SET

user\_id=$\_SESSION[id],

event\_id=".abs(intval($\_POST['hidden\_event\_id'])).",

num=$num

";

$res=mysqli\_query($con, $query) or die(mysqli\_error($con));

if ($res) {

$cnt++; };

};

// если все прошло успешно, фиксируем факт продажи, подсчитываем сумму с учетом скидки

$event\_id=abs(intval($\_POST['hidden\_event\_id']));

$query="

INSERT INTO sales

SET

user\_id=$\_SESSION[id],

event\_id=$event\_id,

sum=$cnt

\*(SELECT price FROM events WHERE id=$event\_id LIMIT 1) # цена за 1 билет

\*ROUND(1-

(SELECT discount\_value FROM users WHERE id=$\_SESSION[id] LIMIT 1) # величина индивидуальной скидки

/100,2)

";

$res=mysqli\_query($con, $query) or die(mysqli\_error($con));

if ($res) {

echo "<p>Успешно куплено билетов: $cnt. Ваши билеты можно просмотреть <a href='tickets.php'>здесь</a><br>

Суммы покупок <a href='sales.php'>здесь</a></p>";};

};

С помощью подключенных стилей делаем нормальный визуал таблиц и страницы в целом.

## 2.5.3 Функционал кассира

Кассир имеет доступ к тем же пунктам меню, что и обычный пользователь, но более расширенный функционал для этих полей: возможность редактирования и удаления записей (как и администратор) доступных полей. В отличие от администратора у него нет права к пользователям и их личным данным и к уровням доступа. На рисунке 23 приведен пример страницы кассира «Сеансы».

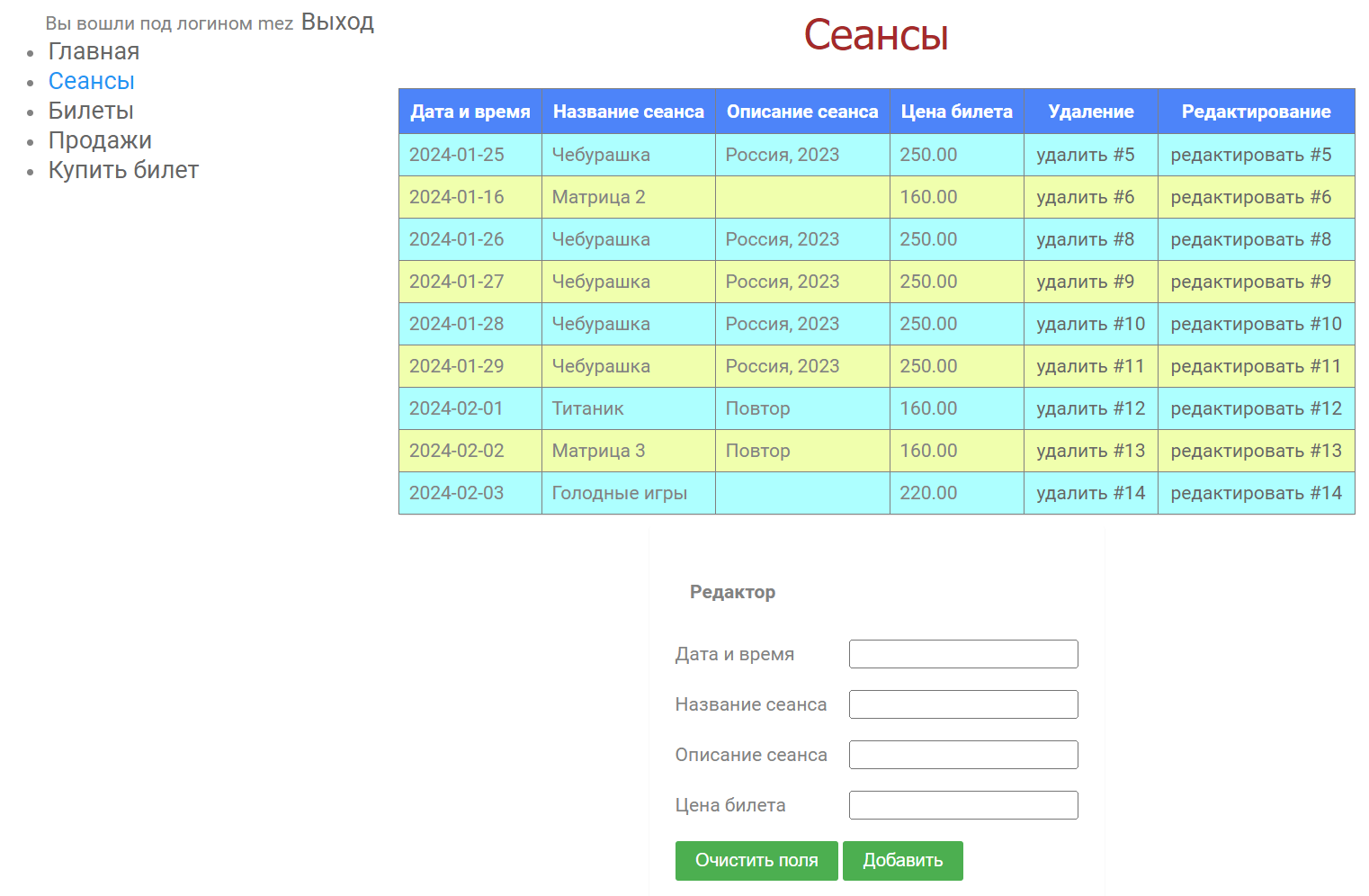


Рисунок 22 — Вкладка «Сеансы» кассира

# ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы по созданию web-сайта для кинотеатра были успешно достигнуты поставленные цели и выполнены поставленные задачи. Проект позволил значительно расширить навыки работы с PHP и MySQL, а также улучшить знания в области веб-разработки.

В работе были использованы следующие технологии: PHP для серверной части и взаимодействия с базой данных; MySQL для создания и управления базой данных; HTML, CSS, JavaScript для разработки пользовательского интерфейса веб-сайта.

Важным этапом было изучение основ работы с базами данных MySQL. Создание и управление базой данных для хранения информации о фильмах, бронированиях и данных пользователей дало понимание важности эффективного хранения и управления информацией. В процессе работы были успешно реализованы функции добавления, просмотра, редактирования, бронирования и удаления билетов, что принесет дополнительную ценность для потенциальных пользователей сайта кинотеатра.

В целом, web-сайт для кинотеатра и онлайн бронирования билетов играет роль мощного инструмента для улучшения опыта клиентов, упрощения и автоматизации процесса бронирования и управления заказами. Оно помогает улучшить эффективность и рентабельность бизнеса в этой сфере, а также улучшить коммуникацию между клиентами и владельцами заведений.

# 

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Белунцов В. Новейший самоучитель по созданию Web-страниц / В. Белунцов. - М.: NT Press, 2012. - 185 с.
2. Браун Брэдли Oracle Database. Создание Web-приложений: учебник / Брэдли Браун, Ричард Ниемик, Джозеф С. Треззо. - СПб.: Лори, 2011. - 722 с.
3. Гончаров А.Ю. Web-дизайн: HTML, JavaScript и CSS. Карманный справочник / А.Ю. Гончаров. - М.: «КУДИЦ-ПРЕСС», 2010. - 320 с.
4. Дунаев В. JavaScript: учеб. пособие / В. Дунаев. - СПб.: Питер, 2010. - 394 с.
5. Книга по html [Электронный ресурс] - Режим доступа: http://htmlbook.ru/
6. Справочник по MySQL [Электронный ресурс] – Режим доступа: https://www.spravkaweb.ru/
7. Янк, К. PHP и MySQL. От новичка к профессионалу / К. Янк. - М.: Эскимо, 2013. - 384 с.