МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**(МОСКОВСКИЙ ПОЛИТЕХ)**

Факультет информационных технологий

Кафедра «Инфокогнитивные технологии»

**КУРСОВОЙ ПРОЕКТ**

на тему: *«Разработка веб-приложения для персонализированного подбора спортивных площадок в городе Санкт-Петербург».*

Направление подготовки 09.03.03 «Прикладная информатика»

Профиль «Корпоративные информационные системы»

**Выполнила:**

студентка группы 221–362

Корнеева Елизавета Сергеевна

|  |  |
| --- | --- |
|  |  |
|  | (подпись) |

16.01.2024

Москва 2024

**Введение**

В современном мире малые бизнесы часто сталкиваются с проблемой недостаточной узнаваемости. Из-за отсутствия собственного веб-сайта их потенциальные клиенты не могут легко найти информацию о предлагаемых товарах и услугах. Это приводит к упущенным возможностям и снижению продаж. В условиях высокой конкуренции на рынке, малым бизнесам особенно важно выделяться и быть доступными для своих клиентов в онлайн-пространстве.

Создание собственного веб-сайта является отличным решением для малых бизнесов. Сайт помогает не только в саморекламе, но и в организации онлайн-заказов, что значительно упрощает процесс взаимодействия с клиентами. Наличие удобного и функционального веб-сайта позволяет бизнесу представить свои товары в лучшем свете, предоставляя всю необходимую информацию и возможность легко оформить заказ.

Кроме того, веб-сайт способствует улучшению доверия клиентов. Профессионально оформленный сайт создает впечатление надежности и стабильности компании, что важно для привлечения новых клиентов и удержания существующих. В итоге создание собственного веб-сайта может стать мощным инструментом для малых бизнесов, помогая им расти и развиваться в условиях современного рынка.

**1 Цель и задачи работы**

Целью работы является разработка веб-сайта для малого бизнеса, предназначенного для повышения узнаваемости и улучшения взаимодействия с клиентами. Этот сайт будет создан для цветочного магазина «Фиалка» в городе Перми. Главной задачей веб-сайта станет предоставление клиентам удобного и доступного инструмента для просмотра ассортимента товаров, оформления онлайн-заказов и получения всей необходимой информации о магазине.

Для достижения цели были определены следующие задачи.

- Изучить предметную область, выявить ключевые сущности и определить их взаимосвязи.

- Разработать полный набор функциональных возможностей, спроектировать структуру базы данных.

- Установить локальный сервер и настроить соединение с базой данных.

- Спроектировать дизайн для desktop-версии, уделяя внимание визуальной эстетике.

- Написать код для предусмотренного функционала, сверстать страницы и добавить интерактивные элементы.

- Провести тестирование веб-приложения и при необходимости выполнить отладку.

Для разработки данного веб-приложения были выбраны следующие инструменты. Для создания интерфейса веб-страниц и обеспечения динамической и интерактивной функциональности использовались HTML, CSS и JavaScript. Обработка пользовательских запросов, взаимодействие с базой данных и реализация функционала приложения были выполнены с помощью PHP. Для хранения данных о товаре, заявках пользователей и аккаунтах администраторов использовалась база данных MySQL.

**2 Проектирование приложения**

Проектирование веб-сайта для цветочного магазина «Фиалка» предлагает широкий спектр функциональных возможностей, которые обеспечат удобство и эффективность взаимодействия пользователей с магазином. Рассмотрим основные функции и описание интерфейса, чтобы предоставить полное представление о работе сайта.

При посещении сайта пользователь попадает на главную страницу, где представлена основная информация о магазине, его контактные данные, адрес и карточки товаров. Каждая карточка содержит название товара, его описание, цену, фотографию и количество, доступное в магазине.

На сайте реализована поисковая строка для быстрого поиска товаров, а также фильтрация по различным критериям, включая указание минимальной и максимальной цены, сортировку цены по возрастанию или убыванию, а также отображение товаров только в наличии. Также была добавлена пагинация для удобного просмотра большого количества карточек товаров, по 12 карточек на страницу.

На главной странице имеется ссылка на форму оформления заявки. Пользователь может оставить свои данные, указать адрес доставки и выбрать товар. Обработка исключений включает в себя контроль количества товара, чтобы пользователь не мог заказать больше, чем имеется в наличии.

После оформления заявки администратор получает уведомление о новой заявке, после чего связывается с клиентом для уточнения деталей заказа. На странице администратора имеется возможность утвердить или удалить заявку полностью, а также изменить количество или удалить отдельные товары из заявки.

При разработке веб-сайта внимание уделялось не только его функциональности, но и эстетике. Все элементы сайта были оформлены в нежных оттенках розового и черно-белого цветового сочетания, создавая классический и стильный внешний вид.

Для добавления элегантности и удобства пользовательского интерфейса использовались SVG иконки. Все кнопки на сайте были оформлены яркими цветами, чтобы привлечь внимание пользователя. При наведении курсора на них происходит изменение цвета, что обеспечивает заметную и понятную обратную связь.

Для всех вышеперечисленных функций необходима база данных, которая будет помогать в работе. В базе данных присутствуют четыре основные сущности: «Покупатель», «Заказ», «Товар» и «Наличие товара», как отражено в инфологической модели базы данных на рисунке 1.

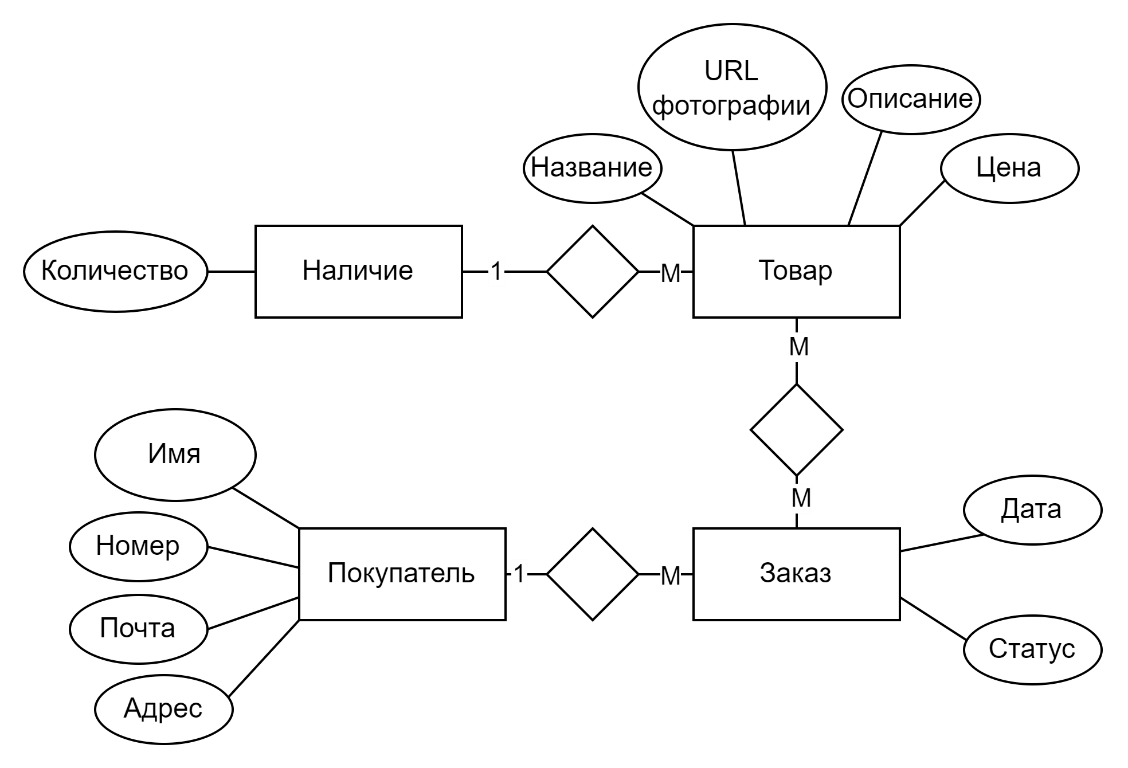


Рисунок 1 − Инфологическая модель базы данных в нотации Чена

Основной сущностью в базе данных является задача, которая определена в базе данных как «tasks». Сущность «tasks» представляет информацию о конкретной задаче, такую как ее название, описание и дата создания. Пользователь - еще одна важная сущность в базе данных. Она содержит информацию о пользователях приложения, а конкретно уникальный идентификатор, имя, логин и пароль пользователя. Пароль хранится в захэшированном виде для обеспечения безопасности.

Даталогическая модель базы данных изображена на рисунке 2.

Изображение выглядит как снимок экрана, текст, дизайн

Автоматически созданное описание

Рисунок 2 – Даталогическая реляционная модель базы данных

После создания инфологической и даталогической моделей базы данных, мы переходим к созданию физической модели базы данных. Физическая модель определяет способ хранения и организации данных на физическом уровне. На данном этапе мы уточняем типы данных, ограничения целостности, отношения таблиц и их ключи. SQL-запросы для создания базы данных:

CREATE TABLE users (

  id int NOT NULL PRIMARY KEY AUTO\_INCREMENT,

  name varchar(255) NOT NULL,

  login varchar(255) NOT NULL,

  password varchar(255) NOT NULL

);

CREATE TABLE status (

  id int NOT NULL PRIMARY KEY AUTO\_INCREMENT,

  name varchar(255) NOT NULL

);

CREATE TABLE priority (

  id int NOT NULL PRIMARY KEY AUTO\_INCREMENT,

  name varchar(255) NOT NULL

);

CREATE TABLE tasks (

  id int NOT NULL PRIMARY KEY AUTO\_INCREMENT ,

user\_id int NOT NULL,

name varchar(255) NOT NULL,

description varchar(255) NOT NULL,

status\_id int NOT NULL,

priority\_id int NOT NULL,

date\_creation datetime NOT NULL,

FOREIGN KEY (user\_id) REFERENCES users(id),

FOREIGN KEY (status\_id) REFERENCES status(id),

FOREIGN KEY (priority\_id) REFERENCES priority(id)

);

Анализ схемы сайта является неотъемлемым этапом в процессе проектирования веб-приложения. Схема сайта визуализирует структуру и навигацию сайта, что помогает лучше организовать контент и функциональность. Она позволяет определить основные разделы и страницы сайта, а также связи между ними. Анализ схемы сайта является ценным инструментом для команды разработчиков и дизайнеров, поскольку он позволяет лучше понять логику перемещения пользователя по сайту и создать удобный интерфейс.

В рассмотренной схеме сайта на рисунке 3 «Главная страница» выступает в качестве отправной точки. Следует отметить, что на сайте отсутствует строгая иерархия страниц, так как практически все страницы взаимосвязаны и имеют ссылки друг на друга. Это говорит о том, что пользователи могут свободно перемещаться по различным разделам сайта и иметь доступ к нужной информации без необходимости следовать фиксированной иерархии.

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание

Рисунок 3 – Схема сайта

Перейдя от анализа схемы сайта, рассмотрим содержимое основных страниц данного веб-приложения. На главной странице находится приветствие и выбор: войти или зарегистрироваться. После выбора будет авторизация или же создание новой учетной записи, а затем перейдем в личный кабинет. В личном кабинете находится таблица со всеми заметками, где можно удалить, отредактировать или добавить новую заметку.

**3 Реализация приложения**

Реализация веб-приложения для заметок представляет собой комплексный процесс, охватывающий различные аспекты, начиная от фронтенда и заканчивая бэкендом. В данном разделе мы подробно рассмотрим основные этапы реализации проекта, сосредотачиваясь на ключевых аспектах, таких как отображение заметок, функции поиска и фильтрации, а также добавление, редактирование и удаление заметок и смена темы.

Фронтенд-разработка отвечает за создание пользовательского интерфейса и взаимодействие с пользователем. На данном этапе основное внимание уделено созданию удобного и интуитивно понятного интерфейса для пользователя. Все вышеперечисленные действия выполняются с помощью HTML, CSS и JavaScript.

HTML используется для создания основной разметки страницы, определения таблицы с данными о заметках, форм поиска и фильтрации, а также форм авторизации/регистрации. CSS применяется для стилизации и визуального оформления веб-страницы.  С помощью CSS были заданы правила для расположения и внешнего вида элементов интерфейса.

JavaScript – это язык программирования, который обеспечивает интерактивность и динамическое поведение веб-приложения. В данном случае JavaScript использовался для обработки событий, таких как клики на элементы интерфейса [1].

Для бэкенд-разработки веб-приложения использовался язык программирования PHP [2]. Бэкенд-часть отвечает за обработку запросов от фронтенда и взаимодействия с базой данных.

База данных MySQL используется для хранения информации о заметках и пользователях. Бэкенд-сервер взаимодействует с базой данных, выполняя операции чтения, записи, обновления и удаления данных.

При создании функции регистрации была разработана HTML-форма, предназначенная для ввода данных пользователем, включая ФИО, логин и пароль. После отправки формы данные проходят обработку на сервере с использованием PHP. Данные проходят проверку на валидность, включая проверку на уникальность логина, что предотвращает дублирование аккаунтов. Пароль пользователя хэшируется для обеспечения безопасного хранения в базе данных MySQL. После этого информация о новом пользователе сохраняется в базе данных.

Функция регистрации на языке PHP:

function signup($mysqli, $name, $login, $password)

{

    $stmt = $mysqli->prepare("SELECT \* FROM users WHERE login=?");

    $stmt->bind\_param("s", $login);

    $stmt->execute();

    $result = $stmt->get\_result();

    if ($result->num\_rows == 0) {

        $stmt = $mysqli->prepare("INSERT INTO users (name, login, password) VALUES (?, ?, ?)");

        $stmt->bind\_param("sss", $name, $login, md5($password));

        $stmt->execute();

        header("Location: auth.php");

    } else {

        echo '<p>Такой пользователь уже существует</p>';

    }

}

В процессе авторизации пользователей HTML-форма позволяет ввести логин и пароль. После отправки формы данные также обрабатываются на сервере при помощи PHP. Вначале происходит проверка существования пользователя с указанным логином, затем сравниваются хэшированные пароли. В случае успешной авторизации создается сессия, которая отслеживает статус авторизации пользователя.

Функция авторизации на языке PHP:

function auth($mysqli, $login, $password)

{

    $stmt = $mysqli->prepare("SELECT \* FROM users WHERE login=? AND password=?");

    $stmt->bind\_param("ss", $login, $password);

    $stmt->execute();

    $result = $stmt->get\_result();

    if (!$result || $result->num\_rows == 0) {

        echo '<div class="help">';

        echo '<p>Такой пользователь не существует.</p>';

        echo '</div>';

    } else {

        $row = $result->fetch\_assoc();

        if ($row['login'] == $login && $row['password'] == $password) {

            session\_start();

            $\_SESSION["user"] = $row;

            header("Location: user.php");

        } else {

            echo '<div class="help">';

            echo '<p>Неверный логин или пароль</p>';

            echo '</div>';

        }

        $stmt->close();

    }

}

Для обеспечения безопасности пользовательских данных при разработке и реализации функций авторизации и регистрации были предприняты меры, включающие хэширование паролей, использование подготовленных запросов для предотвращения атак SQL-инъекций, сохранение сессий в безопасном месте и использование протокола HTTPS для защиты от перехвата данных.

Важной частью функциональности веб-приложения является возможность отображения задач пользователю, а также предоставление инструментов для их поиска и фильтрации. При отображении задач на странице личного кабинета пользователя используется HTML-таблица, где каждая строка представляет собой одну задачу. Информация о каждой задаче выводится в соответствующих столбцах таблицы.

HTML-код таблицы с задачами:

<table class="task-table">

            <thead>

                <tr>

                    <th>Название</th>

                    <th>Описание</th>

                    <th>Статус</th>

                    <th>Приоритет</th>

                    <th>Дата создания</th>

                    <th>Действия</th>

                </tr>

            </thead>

            <tbody>

                <?php

                while ($row = mysqli\_fetch\_assoc($result)) {

                    echo "<tr>";

                    echo "<td>" . $row["name"] . "</td>";

                    echo "<td>" . $row["description"] . "</td>";

                    echo "<td>" . $row["status\_name"] . "</td>";

                    echo "<td>" . $row["priority\_name"] . "</td>";

                   $beautifulDateTime = date('d.m.Y H:i', strtotime($row["date\_creation"]));

                    echo "<td>" . $beautifulDateTime . "</td>";

                    echo "<td class='icon\_table'>";

                    echo "<a href='edit\_task.php?id=" . $row["id"] . "' class='action-link'><img src='image/editL.svg' alt='Редактировать' class='themeToggleEdit'></a>";

                    echo "<a href='delete\_task.php?id=" . $row["id"] . "' class='action-link'><img src='image/deleteL.svg' alt='Удалить' class='themeToggleDelete'></a>";

                    echo "</td>";

                    echo "</tr>";

                }

                ?>

            </tbody>

        </table>

Функция показа задачи на языке PHP:

function show\_task($mysqli, $user\_id)

{

    $stmt = $mysqli->prepare("SELECT tasks.id, tasks.name, tasks.description, tasks.date\_creation, status.name AS status\_name, priority.name AS priority\_name

          FROM tasks JOIN status ON tasks.status\_id = status.id JOIN priority ON tasks.priority\_id = priority.id

          WHERE user\_id = ?");

    $stmt->bind\_param("i", $user\_id);

    $stmt->execute();

    $result = $stmt->get\_result();

    return $result;

}

Используя инструменты поиска и фильтрации, пользователи могут эффективно управлять своими задачами. На странице присутствует поисковая строка для ввода текста для поиска задач по названию или описанию. Также предусмотрены выпадающие списки для выбора фильтров по статусу задачи, приоритету и порядку сортировки по дате создания. После применения фильтров и ввода поискового запроса задачи отображаются с учетом указанных параметров.

Функция поиска по поисковой строке и фильтрам на языке PHP:

function search\_task($mysqli, $user\_id, $query, $date\_order, $status, $priority)

{

    $bindParams = array();

    $sql = "SELECT tasks.id, tasks.name, tasks.description, tasks.date\_creation, status.name AS status\_name, priority.name AS priority\_name

    FROM tasks

    JOIN status ON tasks.status\_id = status.id

    JOIN priority ON tasks.priority\_id = priority.id

    WHERE tasks.user\_id = ?";

    $bindParams[] = $user\_id;

    $bindTypes = "i";

    if (!empty($query)) {

        $query = strtolower($query);

        $sql .= " AND (LOWER(tasks.name) LIKE ? OR LOWER(tasks.description) LIKE ?)";

        $bindParams[] = "%$query%";

        $bindParams[] = "%$query%";

        $bindTypes .= "ss";

    }

    if (!empty($status)) {

        $sql .= " AND status.id = ?";

        $status = intval($status);

        $bindParams[] = $status;

        $bindTypes .= "i";

    }

    if (!empty($priority)) {

        $sql .= " AND priority.id = ?";

        $priority = intval($priority);

        $bindParams[] = $priority;

        $bindTypes .= "i";

    }

    $sql .= " ORDER BY tasks.date\_creation $date\_order";

    $stmt = $mysqli->prepare($sql);

    if (!empty($bindParams)) {

        $stmt->bind\_param($bindTypes, ...$bindParams);

    }

    $stmt->execute();

    $result = $stmt->get\_result();

    return $result;

}

Для удобства пользователей предусмотрена возможность редактирования и удаления задач. При редактировании задачи пользователь видит форму, заполненную данными о выбранной задаче, и может вносить изменения. Данные о задаче отправляются на сервер, где происходит их обновление в базе данных. После удаления задачи она удаляется из базы данных, а пользователь перенаправляется на страницу личного кабинета для обновления списка задач.

Функция редактирования задачи на языке PHP:

function edit\_task($mysqli, $name, $description, $status\_id, $priority\_id, $task\_id)

{

    $stmt = $mysqli->prepare("UPDATE tasks SET name=?, description=?, status\_id=?, priority\_id=? WHERE id=?");

    $stmt->bind\_param("ssiii", $name, $description, $status\_id, $priority\_id, $task\_id);

    $stmt->execute();

    header("Location: user.php");

}

Функция удаления задачи на языке PHP:

function delete\_task($mysqli, $task\_id)

{

    $stmt = $mysqli->prepare("DELETE FROM tasks WHERE id=?");

    $stmt->bind\_param("i", $task\_id);

    $stmt->execute();

    header("Location: user.php");

}

Функция добавления задачи на языке PHP:

function add\_tasks($mysqli, $user\_id, $name, $description, $status\_id, $priority\_id)

{

    date\_default\_timezone\_set('Europe/Moscow');

    $currentDateTime = date('Y-m-d H:i:s');

    $stmt = $mysqli->prepare("INSERT INTO tasks (user\_id, name, description, status\_id, priority\_id, date\_creation) VALUES (?, ?, ?, ?, ?, ?)");

    $stmt->bind\_param("isssis", $user\_id, $name, $description, $status\_id, $priority\_id, $currentDateTime);

    if ($stmt->execute()) {

        header("Location: user.php");

        exit;

    } else {

        echo "Ошибка: " . $mysqli->error;

    }

}

Смена темы сайта позволяет пользователям выбирать предпочтительный внешний вид сайта. JavaScript-код обеспечивает переключение темы при нажатии на кнопку, изменяя CSS-переменные и изображения для соответствующих элементов страницы. Сохранение выбранной темы при перезагрузке страницы осуществляется с использованием куки.

Файл tema.js:

let root = document.querySelector(":root");

let themeToggle = document.querySelector("#themeToggle");

let themeToggleDelete = document.querySelectorAll(".themeToggleDelete");

let themeToggleEdit = document.querySelectorAll(".themeToggleEdit");

let iconLight = "image/month.svg";

let iconDark = "image/sun.svg";

let iconLightDelete = "image/deleteL.svg";

let iconLightEdit = "image/editL.svg";

let iconDarkDelete = "image/deleteD.svg";

let iconDarkEdit = "image/editD.svg";

function setTheme(theme) {

    root.classList.toggle('dark', theme === 'dark');

    themeToggle.src = theme === 'dark' ? iconDark : iconLight;

    themeToggleDelete.forEach((element) => {

        element.src = theme === 'dark' ? iconDarkDelete : iconLightDelete;

    });

    themeToggleEdit.forEach((element) => {

        element.src = theme === 'dark' ? iconDarkEdit : iconLightEdit;

    });

    document.cookie = "theme=" + theme + ";path=/";

}

themeToggle.addEventListener('click', () => {

    event.preventDefault();

    const newTheme = root.classList.contains('dark') ? 'light' : 'dark';

    setTheme(newTheme);

});

function getCookie(name) {

    const decodedCookie = decodeURIComponent(document.cookie);

    const cookies = decodedCookie.split(';');

    for (let i = 0; i < cookies.length; i++) {

        let cookie = cookies[i];

        while (cookie.charAt(0) === ' ') {

            cookie = cookie.substring(1);

        }

        if (cookie.indexOf(name + '=') === 0) {

            return cookie.substring(name.length + 1, cookie.length);

        }

    }

    return '';

}

setTheme(getCookie("theme"));

**4 Основные сценарии использования приложения**

В данном разделе мы продемонстрируем некоторые ключевые возможности приложения, а первым делом покажем начальное окно веб-приложения в двух темах (рисунки 4–5).

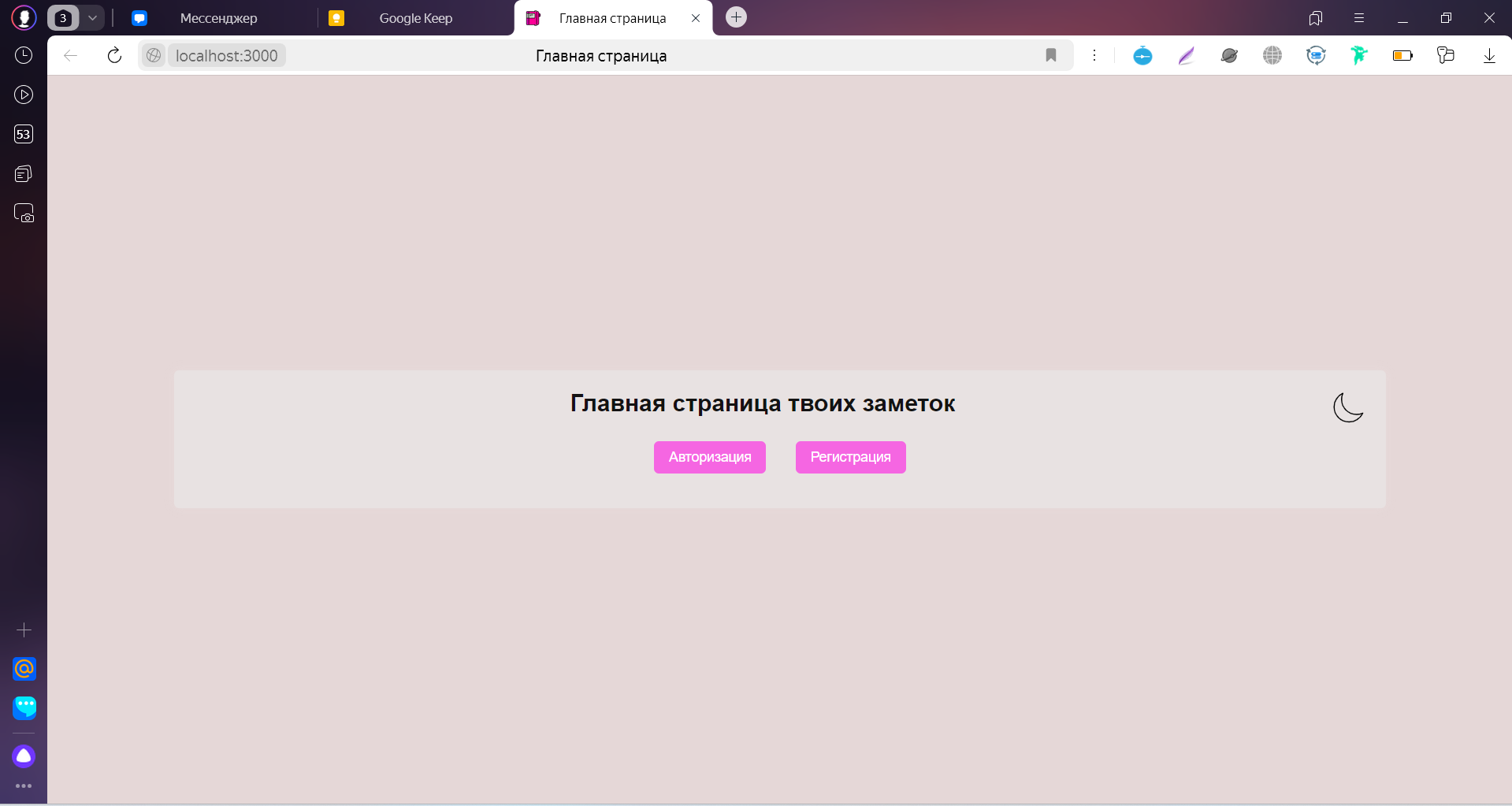


Рисунок 4 – Начальный экран светлой темы

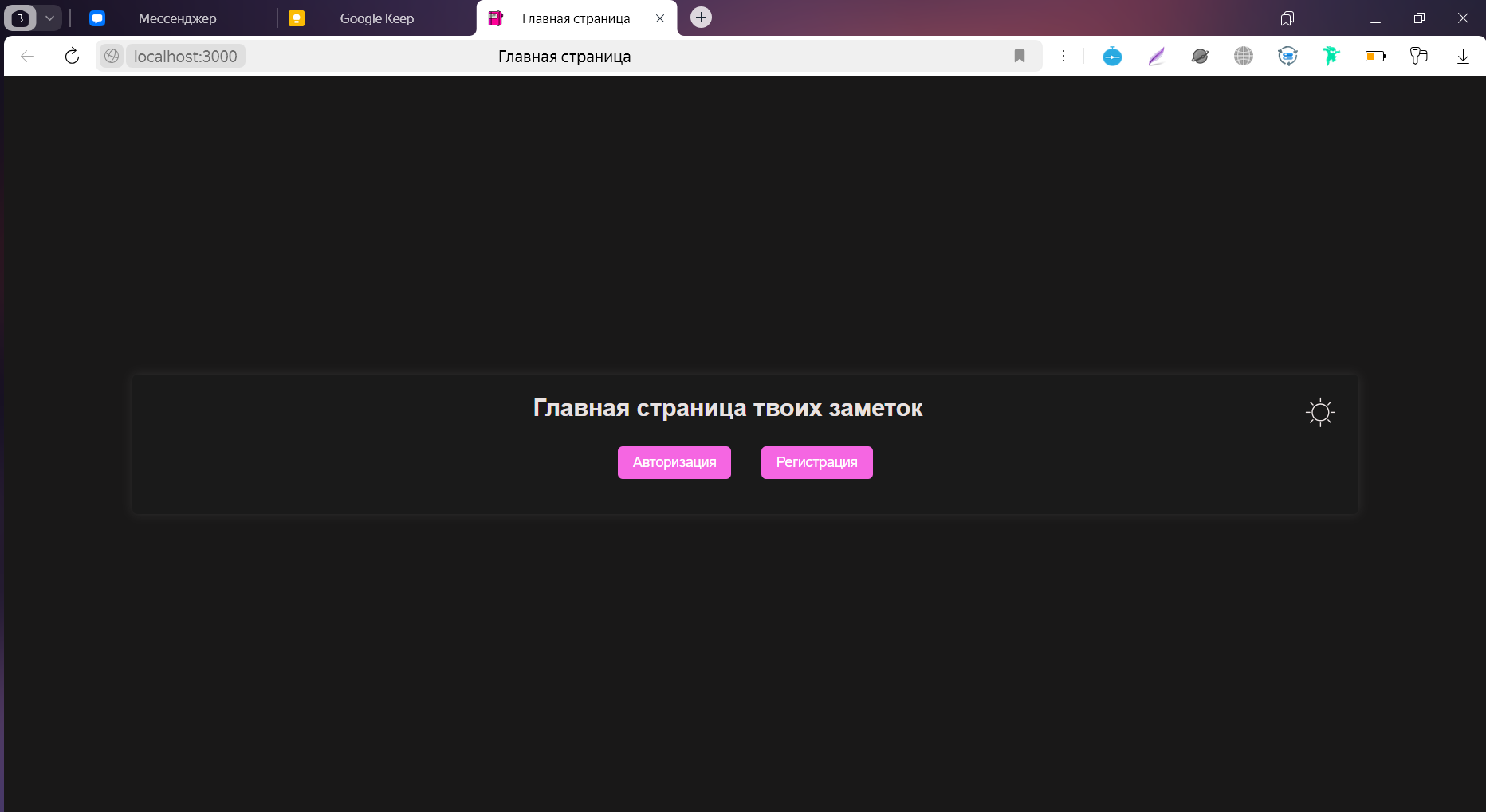


Рисунок 5 – Начальный экран темной темы

Личный кабинет пользователя в двух темах изображен на рисунках 6–7.

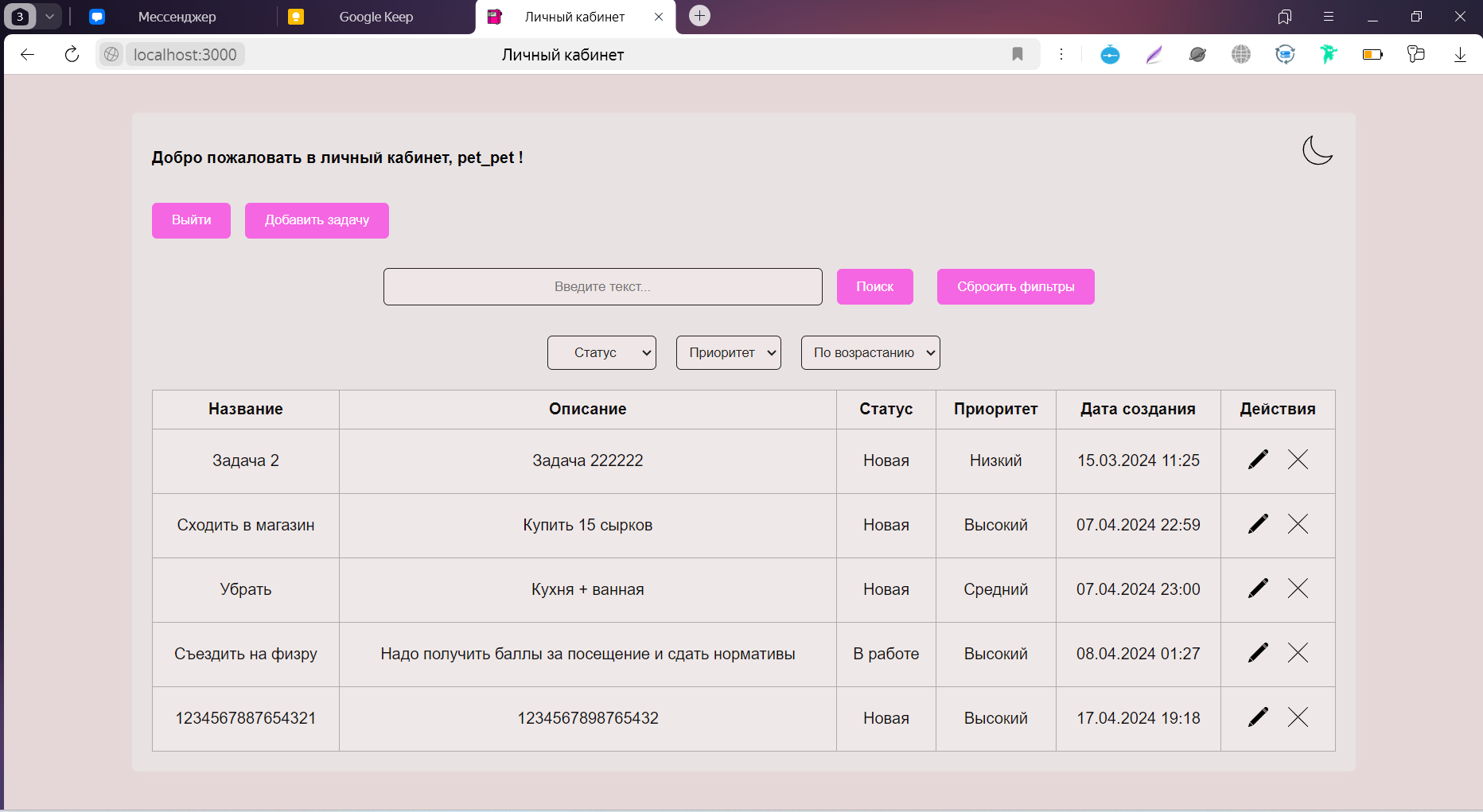


Рисунок 6 – Личный кабинет светлой темы

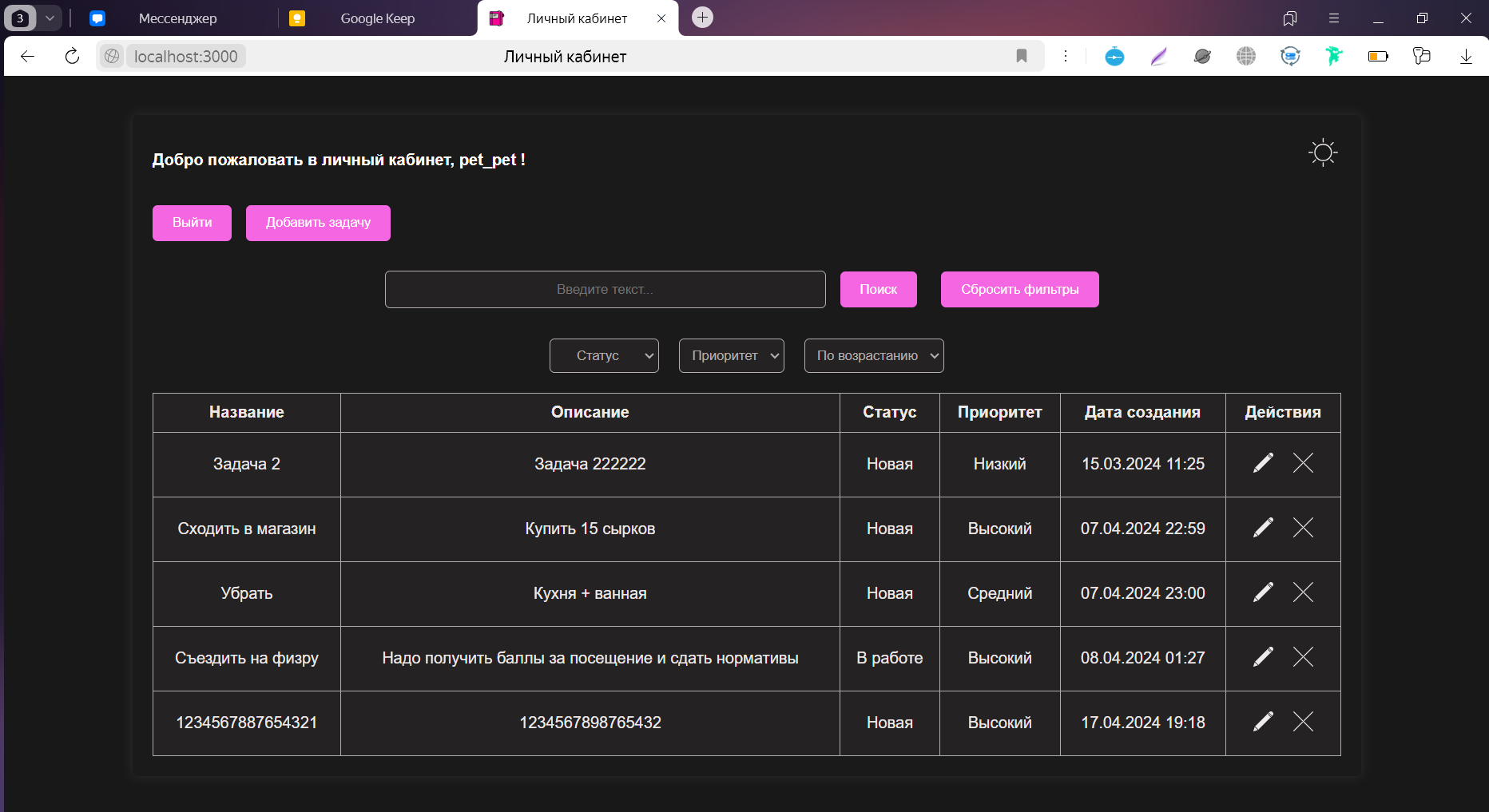


Рисунок 7 – Личный кабинет темной темы

Продемонстрируем работу функции поиска и фильтрации. Мы совершим поиск, используя фильтр по статусу задачи, выбрав статус «Новая», и приоритету, выбрав приоритет «Высокий». Результаты поиска будут показаны на рисунке 8.

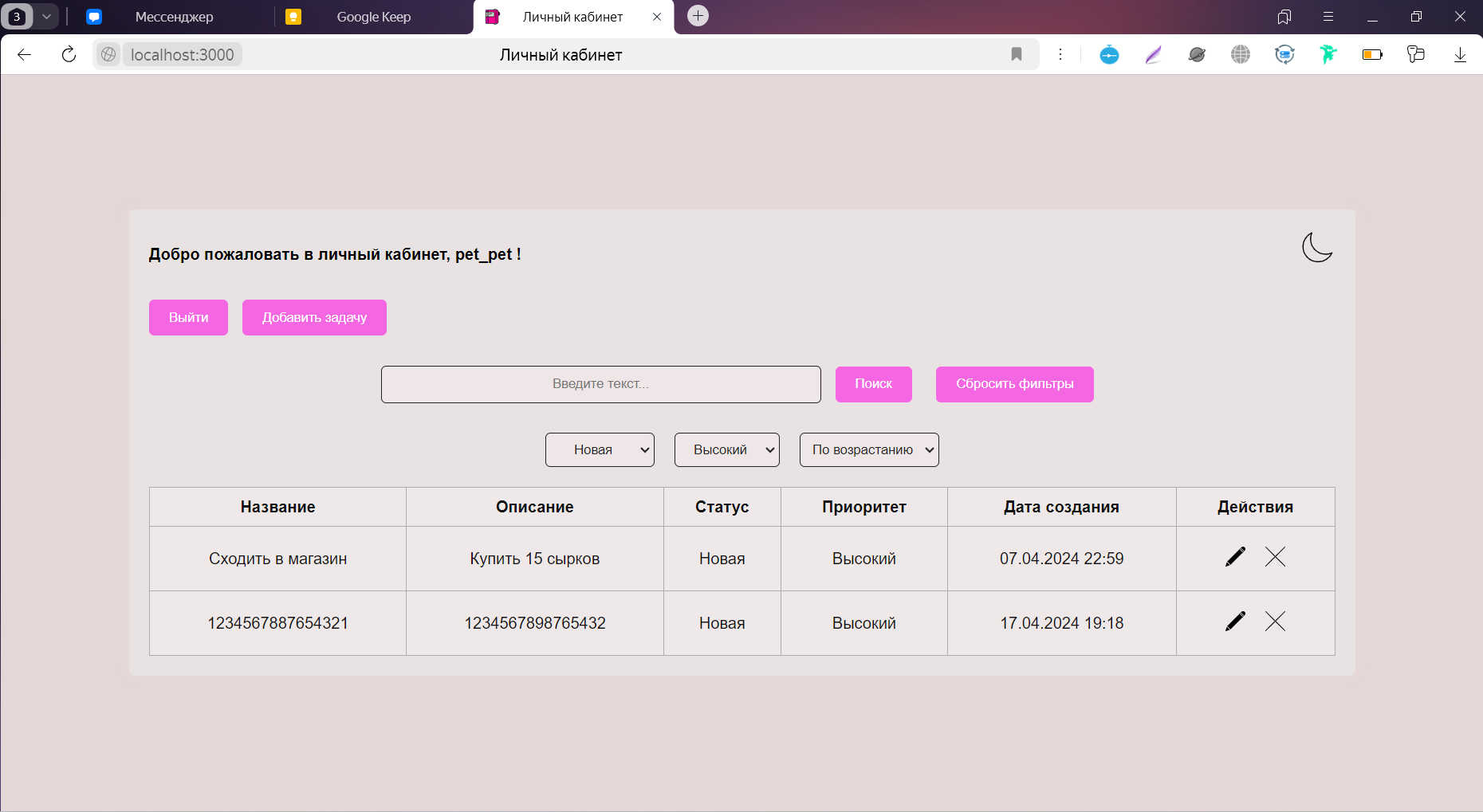


Рисунок 8 – Демонстрация функции поиска и фильтрации

После выполнения фильтрации по статусу и приоритету мы можем перейти к добавлению задачи, как показано на рисунке 9. Заполняем форму добавления задачи и нажимаем на кнопку «Добавить задачу». Задача отображается в личном кабинете (рисунок 10).

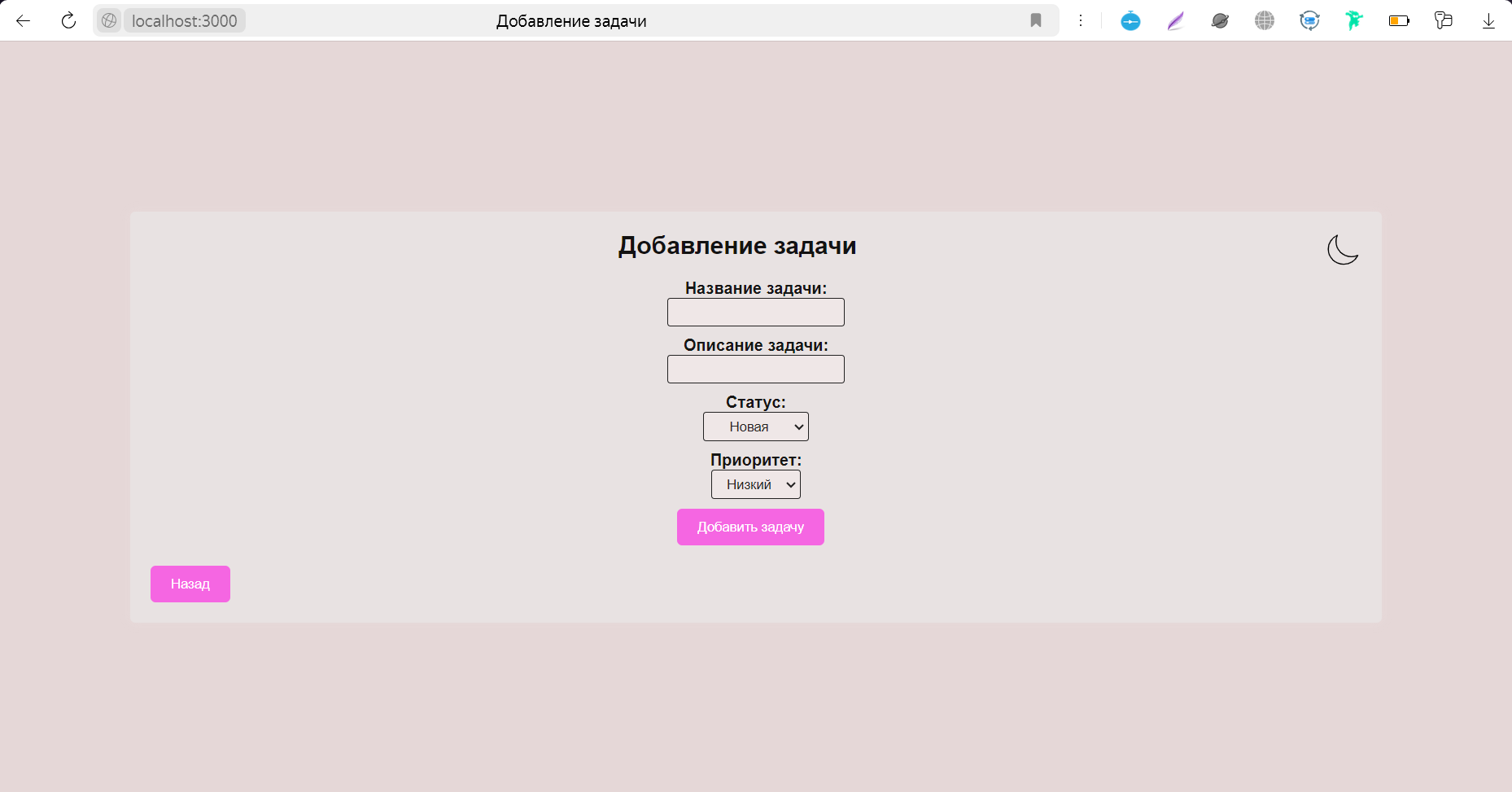


Рисунок 9 – Форму добавления новой задачи

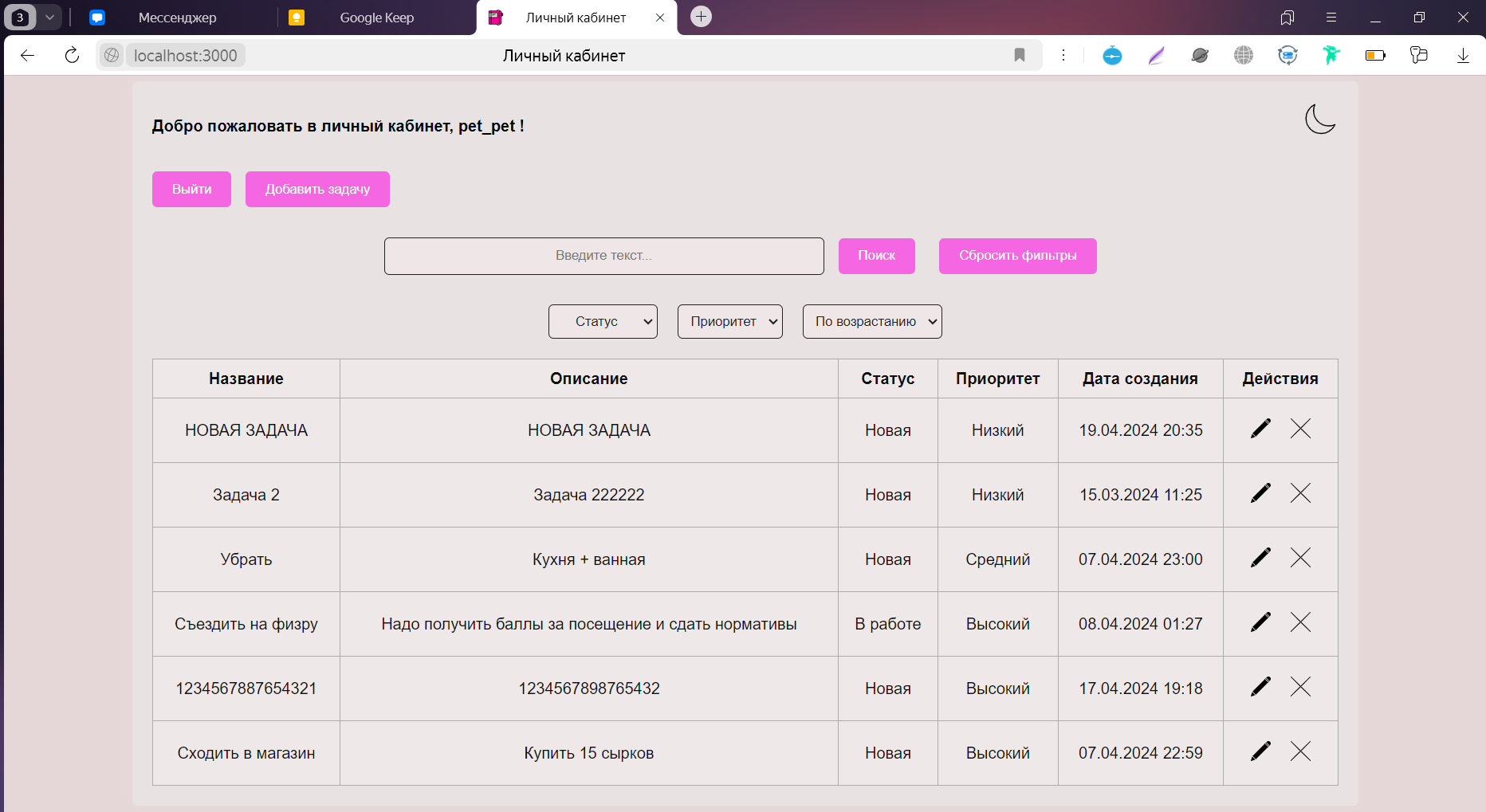


Рисунок 10 – Личный кабинет с новой добавленной задачей

Кроме того, задачи можно еще редактировать и удалять. Проверим функцию редакции задачи, нажав на иконку ручки. Форма редактирования задачи показана на рисунке 11. Изменим описание задачи и нажмем на кнопку «Сохранить изменения». Теперь задача в личном кабинете изображена с изменениями (рисунок 12).

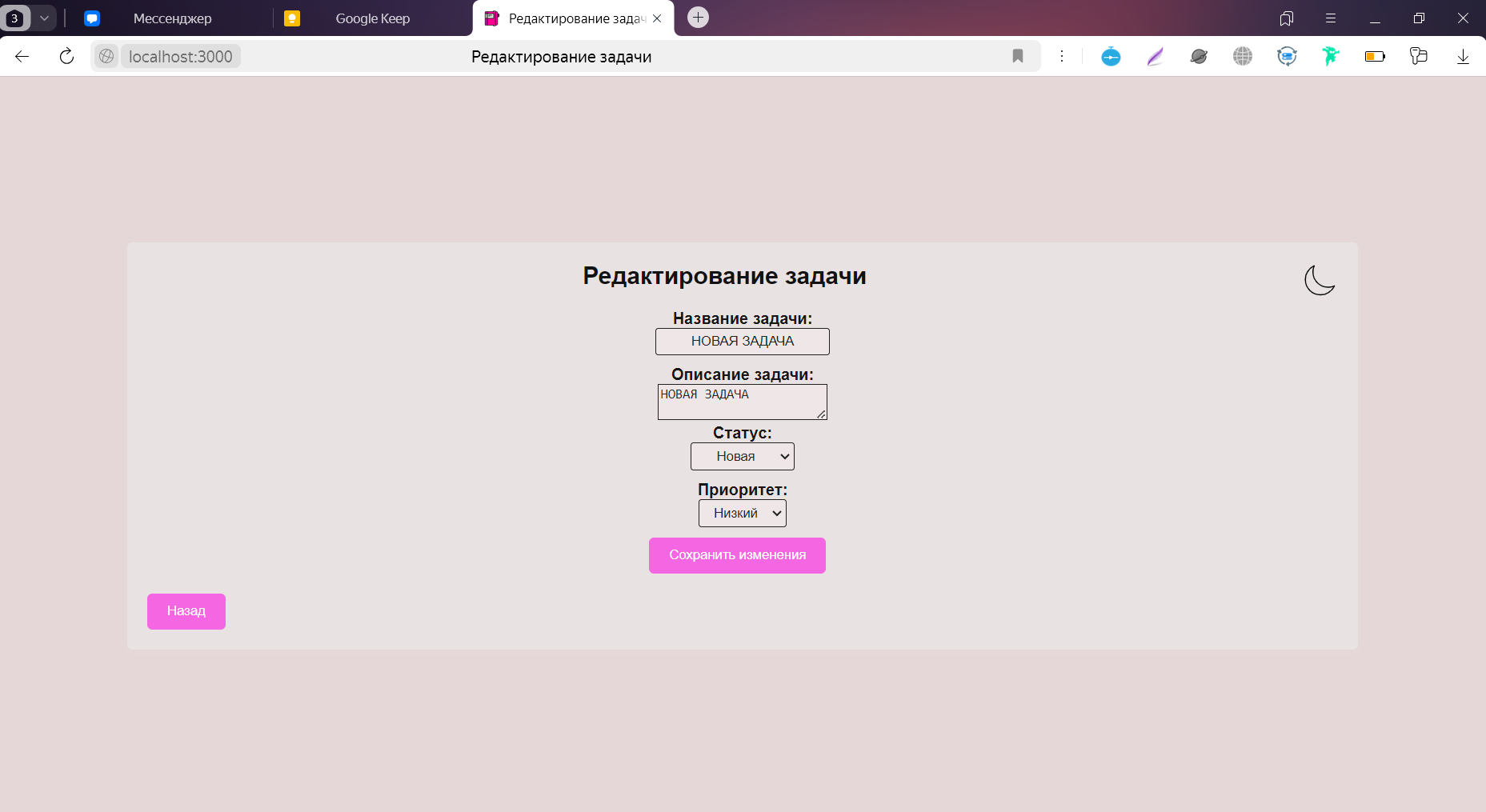


Рисунок 11 – Форма редактирования задачи

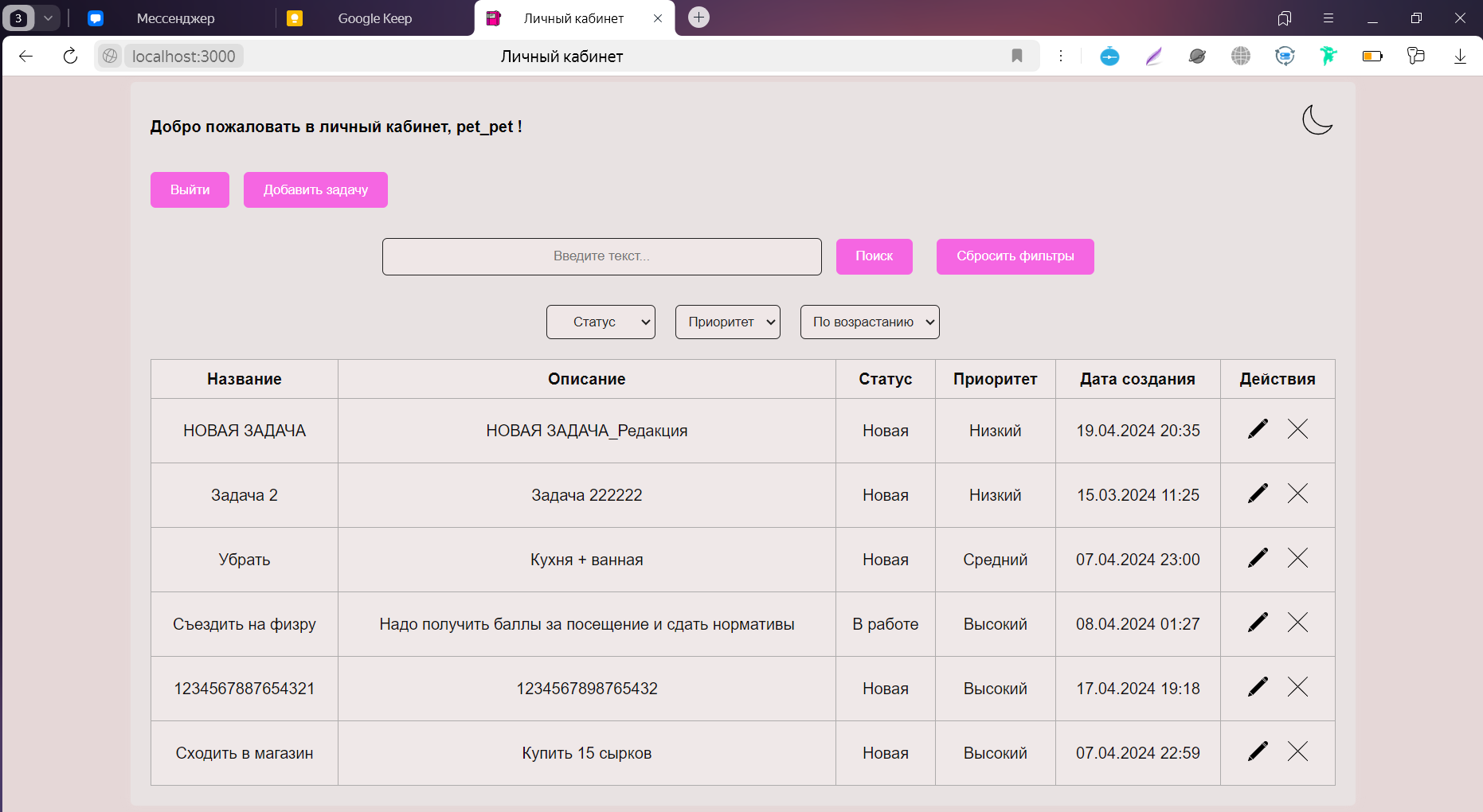


Рисунок 12 – Личный кабинет с измененной задачей

Удалим только что отредактированную задачу, нажав на крестик (рисунок 13).

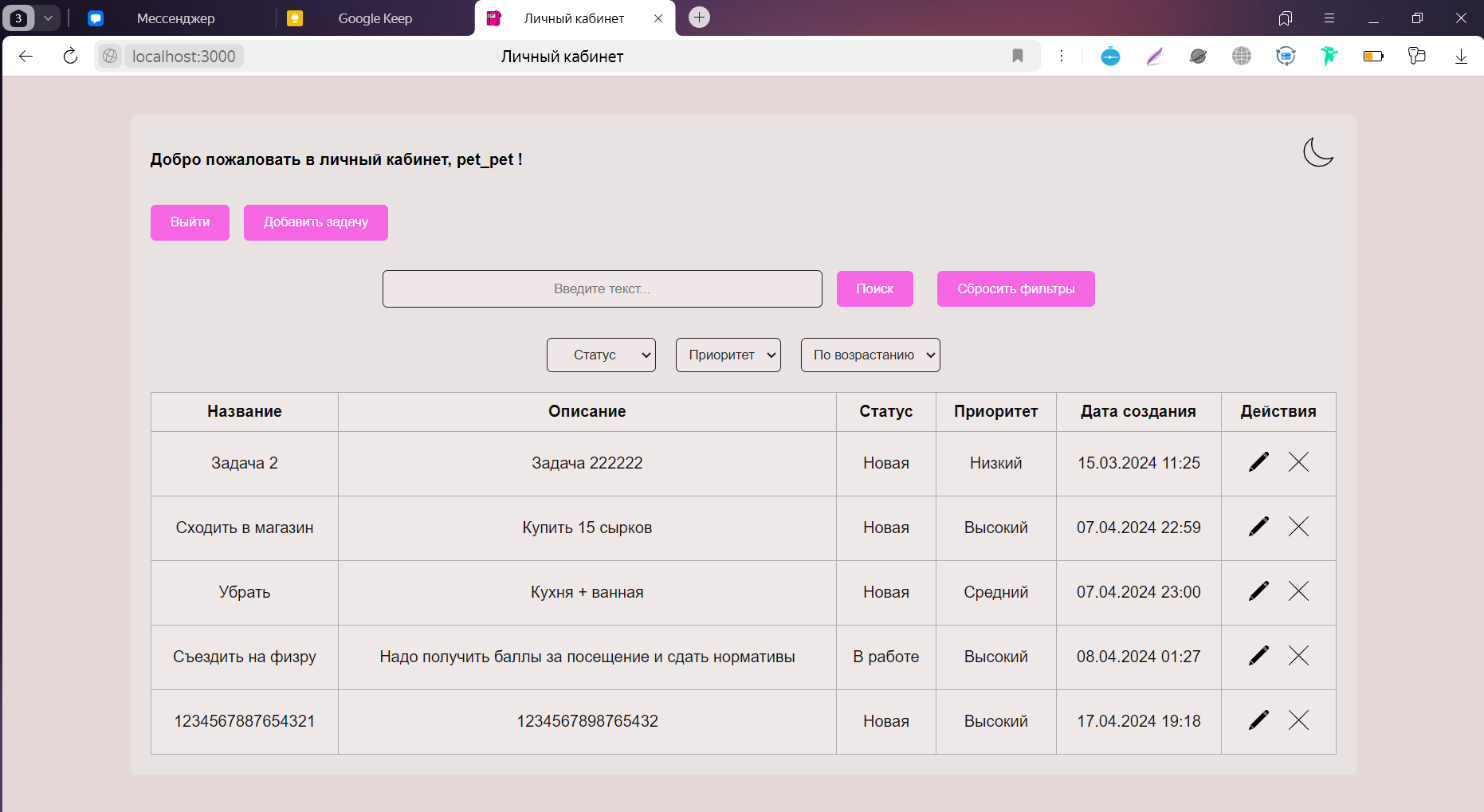


Рисунок 13 – Личный кабинет после удаления задачи

**Заключение**

В рамках курсового проекта по дисциплине «Базы данных» было разработано веб-приложение для управления делами и задачами пользователей с учетом их индивидуальных потребностей.

Целью работы было создание инструмента, который помог бы эффективно организовывать время и управлять повседневными делами, что в итоге привело бы к повышению продуктивности и эффективности пользователей.

В процессе разработки была проведена аналитическая работа по выделению сущностей и их взаимосвязей в предметной области, спроектирована база данных, учтены все необходимые атрибуты и связи между сущностями, создан локальный сервер с настроенным соединением к базе данных, разработан дизайн интерфейса для desktop-версии приложения с учетом визуальной эстетики и удобства использования, написан и отлажен код для реализации всего функционала приложения, включая обработку запросов пользователей и взаимодействие с базой данных, проведено тестирование веб-приложения, в результате которого были выявлены и устранены возможные ошибки и недочеты.

Дальнейшее развитие проекта может включать в себя добавление возможности напоминания о задачах и выставление дедлайнов. В целом, разработанное веб-приложение может стать полезным инструментом для пользователей, стремящихся к более организованному и продуктивному образу жизни. Результаты данного проекта могут быть применены для создания полноценного сервиса по управлению задачами и делами с учетом индивидуальных потребностей пользователей.

**Список литературы и интернет-ресурсов**

1. Современный учебник JavaScript [Электронный ресурс]. URL: https://learn.javascript.ru/ (дата обращения: 09.04.2024).

2. Руководство по PHP [Электронный ресурс]. URL: https://www.php.net/manual/ru/index.php (дата обращения: 11.04.2024).