МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное

учреждение высшего образования

«Казанский национальный исследовательский технический

университет им. А.Н. Туполева-КАИ»

(КНИТУ-КАИ)

Институт компьютерных технологий и защиты информации

(наименование института (факультета), филиала)

\_\_Кафедра Прикладной Математики и Информатики \_\_

(наименование кафедры)

\_\_\_\_\_\_\_\_09.03.04 «Программная инженерия» \_\_\_

(шифр и наименование направления подготовки (специальности))

пояснительная записка

к Курсовой работе

по дисциплине:

«Технологии Веб-программирования»

на тему: «Клиент-серверное приложение “Бюро Находок”»

Обучающийся 4310 Белов В.А.

(номер группы) (подпись, дата) (Ф.И.О.)

Руководитель доцент каф. ПМИ Тутубалин П.И.

(должность) (Ф.И.О.)

Курсовая работа зачтена с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись, дата)

Казань 2024

ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 3](#_Toc168484198)

[1. ПОСТАНОВКА ЗАДАЧИ 4](#_Toc168484199)

[1.1 Основные требования к задаче 4](#_Toc168484200)

[1.2. Разработка терминологического словаря предметной области 6](#_Toc168484201)

[1.3. Общие требования к проектируемой системе 11](#_Toc168484202)

[1.4. Категории пользователей проектируемой системы 11](#_Toc168484203)

[2. МОДЕЛЬ ДАННЫХ СИСТЕМЫ 12](#_Toc168484204)

[2.1 Выбранная модель данных системы 12](#_Toc168484205)

[2.2 Преимущества выбранной системы 13](#_Toc168484206)

[3. МАКЕТЫ HTML-СТРАНИЦ СИСТЕМЫ 14](#_Toc168484207)

[3.1 Макет главной страницы 14](#_Toc168484208)

[3.2 Макет страницы личного кабинета 14](#_Toc168484209)

[4. СТРУКТУРНАЯ СХЕМА МОДУЛЕЙ ПО, СОСТАВЛЯЮЩИХ СИСТЕМУ 16](#_Toc168484210)

[4.1 Структурная модель приложения. Модель MVC 16](#_Toc168484211)

[4.2 Структурная схема HTML страниц на стороне клиента 17](#_Toc168484212)

[5. ДИАГРАММА КЛАССОВ РАЗРАБОТАННОЙ СИСТЕМЫ 19](#_Toc168484213)

[6. ОБЗОР ПО, ВЫБРАННОГО ДЛЯ РЕАЛИЗАЦИИ СИСТЕМЫ 20](#_Toc168484214)

[6.1 Среда разработки – Visual Studio Code 20](#_Toc168484215)

[6.2 Веб-браузер Yandex Browser и инструменты разработчика 20](#_Toc168484216)

[6.3 Программная платформа Node.js 21](#_Toc168484217)

[6.4 СУБД MongoDB 21](#_Toc168484218)

[7. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 22](#_Toc168484219)

[7.1 Руководство пользователя 22](#_Toc168484220)

[7.2 Руководство для работников 27](#_Toc168484221)

[ИСПОЛЬЗОВАННЫЕ ИСТОЧНИКИ 31](#_Toc168484222)

[ПРИЛОЖЕНИЕ. Ресурсы с исходным кодом 32](#_Toc168484223)

[ПРИЛОЖЕНИЕ 1.Файл server.js 32](#_Toc168484224)

[ПРИЛОЖЕНИЕ 2.Файл routes.js 32](#_Toc168484225)

[ПРИЛОЖЕНИЕ 3.Файл announcementModel.js 35](#_Toc168484226)

[ПРИЛОЖЕНИЕ 4.Файл userModel.js 36](#_Toc168484227)

[ПРИЛОЖЕНИЕ 5. Файлы main.html, main.js, main.css 37](#_Toc168484228)

[ПРИЛОЖЕНИЕ 6. Файл announcementController.js 44](#_Toc168484229)

[ПРИЛОЖЕНИЕ 7. Файл userController.js 48](#_Toc168484230)

# ВВЕДЕНИЕ

К сожалению, никто не застрахован от потери личных вещей. Случайно оставленный ноутбук, или выпавшая карта из кармана становится досадной утратой.

В эпоху цифровых технологий, объявление на бумажных издательствах или досках объявлений о потери или находке почти не имеет смысла, т.к. это зачастую долго и неэффективно. Также отдача потерянных вещей в какую-либо организация бюро находок, затрачивает ваше время и силы.

Я предлагаю решение данной проблемы, путем создания электронной площадки Бюро находок, которая позволит быстро и эффективно сообщить о находке или пропаже.

В данном приложении Вы сможете добавить объявление о находке или пропаже. Также сможете посмотреть уже выставленные объявления, при необходимости отсортировав их по дате. Быстро и легкость, вот, что позволит вам быстро найти свою вещь.

Тема моей курсовой работы – разработка и реализация клиент-серверного приложения «Бюро находок». В этой работе предложена один из возможных реализаций подобного сервиса.

# ПОСТАНОВКА ЗАДАЧИ

## 1.1 Основные требования к задаче

Целью курсовой работы является разработка и программная реализация клиент-серверного приложения, а также закрепления знаний, полученных при изучении дисциплины, и получение практических навыков программирования информационных систем с использованием современных технологий и инструментальных средств, и оформления сопровождающей документации программного обеспечения.

Задача курсовой работы состоит в разработке клиент-серверного приложения «Бюро находок» (web-клиента и web-сервера). Для выполнения этой задачи, были определены следующие требования:

* произвольная среда разработки – удобный современный текстовый редактор;
* применение системы контроля версий – git;
* опора на представление «Model-View-Controller»;
* разработка web-клиента с применением html, css, JavaScript, jQuery;
* разработка web-сервера с применением JSON, AJAX, Node.js, Express и NPM;
* использование базы данных (БД) на стороне web-сервера с применением СУБД – MongoDB;

Большая часть сети Интернет построена на технологии клиент-серверного взаимодействия. Выбор данного типа взаимодействия, позволит удаленно передавать файлы между устройством клиента и сервера, что подчеркнет актуальность клиент-серверного приложения

Объектом моей курсовой работы является создание клиент-серверного приложения «Бюро находок».

Предметом работы являются языки программирования для разработки приложений web-сервера и web-клиента.

Пользователи и бизнес-процессы предметной области проекта

Таблица 1 Пользователи и бизнес-процессы предметной области проекта

|  |  |
| --- | --- |
| **Пользователи** | **Основные бизнес-процессы** |
| Администратор  Модератор  Пользователь | Администрирование системы:   * Обновление и поддержка актуальности находок. * Управление функционалом пользователей * Управление персоналом   Добавление находок и их Модерирование:   * Модерация входящих заявок * Возможность добавления находок   Редактирование находок:   * Редактирование информации о находке * Изменение статуса находки   Отображение найденных и потерянных вещей:   * Просмотр всех найденных и потерянных вещей. * Возможность просмотра найденных и потерянных вещей на определенный день/неделю. |

## 1.2. Разработка терминологического словаря предметной области

Таблица 2 Терминологический словарь

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Термин** | **Применение** | **Комментарии** |
| 1 | Бюро находок | Организация, занимающаяся каталогизацией, хранением и возвратом утерянных вещей | В нашем случае за хранение и передачу вещей будет отвечать сам пользователь |
| 2 | Находка | Вещь, которую случайно нашли. В нашем случае, будет использоваться, как обобщенный термин найденных и утерянных вещей. | Используется в качестве определения найденной/утерянной вещи |
| 3 | Найденная вещь | Вещь, обнаруженная случайно или при активном поиске владельца. | Поиск потерянных вещей является одной из главных функций онлайн бюро находок, помогая вернуть утерянные предметы и привлекая владельцев. |
| 4 | Потерянная вещь | Вещь, которая была случайно утрачена или оставлена. | Размещение объявлений о найденных вещах на онлайн бюро помогает ускорить процесс их возврата и связать находившего с владельцем. |
| 5 | Нашедший (человек) | Человек, который нашел вещь | Откликается на объявление пропажи, либо сам создает заявку о находке |
| 6 | Потерявший (человек) | Человек, который потерял вещь | Откликается на объявление о находке, либо сам создает заявку о пропаже |
| 7 | Время находки | Предположительное время нахождение/пропажи вещи | Указывает на временной диапазон для облегчения навигации по данным |
| 8 | Место находки | Место нахождения вещи, либо предположительное место утраты вещи | Указывает местоположение находки. |
| 9 | Онлайн бюро находок | Онлайн сервис, позволяющий пользователям размещать объявления о потерянных и найденных вещах, чтобы помочь им вернуть утерянное имущество или найти потерянное. | Онлайн бюро находок удобно использовать для поиска утерянных вещей, так как позволяет достигнуть большей аудитории, чем обычные доски объявлений. |
| 10 | Объявление о потерянной вещи | Описание утерянной вещи, размещенное на онлайн бюро находок для поиска владельца. | Чем более подробное объявление, тем выше шанс найти утерянное имущество. |
| 11 | Объявление о найденной вещи | Описание найденной вещи, размещенное на онлайн бюро находок для поиска владельца. | Это помогает лицу, нашедшему вещь, связаться с ее владельцем и вернуть ее. |
| 12 | Категоризация | Разделение объявлений на онлайн бюро находок по категориям (например, ключи, документы, телефоны). | Это упрощает поиск нужной информации и ускоряет процесс нахождения утерянной вещи. |
| 13 | Контактная информация | Данные, оставляемые в объявлении для связи с автором или владельцем найденной вещи. | Важно указать достоверные контактные данные, чтобы облегчить процесс возврата утерянной вещи. |
| 14 | Рейтинг пользователя | Оценка доверия к автору объявления или лицу, которое нашло вещь. | Рейтинг помогает определить надежность пользователей и предотвратить мошенничество. |
| 15 | Система комментариев | Функция, позволяющая оставлять отзывы и комментарии к объявлениям. | Комментарии могут содержать полезные сведения о находке или помочь определить владельца утерянной вещи. |
| 16 | Фильтры поиска | Настройка параметров поиска для более быстрого и удобного поиска утерянной вещи. | Фильтры помогают исключить ненужные объявления и найти нужное быстрее. |
| 17 | Система уведомлений | Уведомления о новых объявлениях или ответах на размещенные объявления. | Это помогает быть в курсе событий и получать важную информацию своевременно. |
| 18 | Публичный профиль | Профиль пользователя, на котором отображаются все его объявления и комментарии. | Публичный профиль упрощает взаимодействие пользователей и повышает их доверие друг к другу. |
| 19 | Услуги по доставке | Предложения о доставке утерянных или найденных вещей до их владельцев. | Доставка может быть удобным способом вернуть утерянную вещь или передать найденное имущество. |
| 20 | Платная поддержка | Услуги с дополнительной платой для расширенного объявления или привилегированных возможностей. | Платная поддержка может быть полезна для повышения видимости объявления или быстрой находки утерянной вещи. |
| 21 | Система отслеживания | Технология отслеживания утерянных вещей с помощью мобильного приложения или сайта. | Это удобный способ контролировать местоположение утерянной вещи и помогает быстрее ее найти. |
| 22 | Чат для связи | Функция общения между пользователями онлайн бюро находок. | Чат упрощает обсуждение деталей об объявлениях и ускоряет процесс возврата утерянной вещи. |
| 23 | Информационные ресурсы | Полезная информация о возврате утерянных вещей и советы по безопасному хранению ценностей. | Информационные ресурсы помогают пользователям уменьшить риск потери вещей и улучшить шансы их возврата. |
| 24 | Оценка утерянных вещей | Описание особенностей и характеристик утерянной вещи для правильной идентификации | Оценка помогает точно описать утерянную вещь и ускоряет ее поиск. |
| 25 | Карта находок | Интерактивная карта с обозначением мест, где были потеряны или найдены вещи. | Карта помогает визуально представить места утери и находки вещей, что упрощает поиск. |
| 26 | Советы по безопасности | Рекомендации по сохранению вещей и предотвращению их потери. | Соблюдение советов по безопасности поможет избежать утери ценностей и повысить шансы на их возврат. |
| 27 | Возврат вещи | Процесс возвращения утерянной вещи ее законному владельцу. | Онлайн бюро находок способствует быстрому возврату утраченных вещей, сокращая время и усилия, необходимые для обратной передачи. |
| 28 | Идентификация владельца | Определение легального владельца утерянной вещи для успешного возврата. | Идентификация владельца, основанная на предоставленных данными источниках, облегчает возврат утерянных вещей и избегает недопониманий. |
| 29 | Маркер места утери | Функция, позволяющая отметить место утери вещи на карте. | Маркер места утери помогает пользователям лучше ориентироваться и описывать окружающую обстановку для более успешного поиска утерянной вещи. |
| 30 | Характеристики вещи | Детальное описание особенностей и состояния утерянной или найденной вещи. | Чем более точно описаны характеристики вещи, тем выше вероятность ее успешного возврата владельцу или найденного. |
| 31 | Уникальный код вещи | Индивидуальный идентификационный код, нанесенный на вещь для облегчения ее поиска и идентификации. | Уникальный код помогает ускорить возврат утерянных вещей, поскольку облегчает процесс идентификации и связи с владельцем. |
| 32 | Временное хранение | Временное хранение утерянных вещей до успешного возврата владельцу. | Возможность временного хранения вещей на онлайн бюро находок помогает обеспечить их безопасность и удобство при ожидании владельца. |
| 33 | Подтверждение собственности | Предоставление доказательств права собственности на утерянную вещь для возврата. | Подтверждение собственности обеспечивает безопасность процесса возврата вещей и предотвращает недобросовестные попытки завладения чужим имуществом. |
| 34 | Размещение объявлений | Постановка информации о потерянных или найденных вещах на онлайн бюро. | Размещение объявлений помогает связать владельцев с их утерянными вещами или лиц, нашедшими предметы, и облегчает процесс возврата. |
| 35 | Система наград | Предоставление поощрений и вознаграждений для пользователей, успешно возвращающих утерянные вещи. | Система наград стимулирует активное участие в процессе возврата вещей и повышает мотивацию пользователей. |
| 36 | Общественные объявления | Публичные сообщения о потерянных или найденных вещах, доступные для просмотра всем пользователям онлайн бюро. | Общественные объявления увеличивают шансы на обнаружение владельцев утерянных вещей или находчиков и способствуют успешному возврату. |
| 37 | Удобство пользования | Легкость и удобство пользования онлайн бюро находок для размещения и поиска информации о вещах. | Удобство пользования повышает эффективность работы и привлекательность сервиса для пользователей. |
| 38 | Идентификация находок | Процесс установления подлинности найденных вещей и описания их особенностей. | Идентификация находок помогает точно описать предметы и ускоряет их успешный возврат владельцам. |
| 39 | Приватность данных | Защита личной информации пользователей и конфиденциальности данных о находках. | Предоставление высокого уровня приватности данных способствует доверию пользователей к сервису и обеспечивает безопасность информации о вещах. |
| 40 | Правила пользования | Установленные правила и ограничения для использования онлайн бюро находок. | Правила пользования помогают поддерживать порядок и безопасность на платформе, обеспечивая удовлетворительный опыт для всех пользователей. |
| 41 | Механизм вознаграждения за находку | Система вознаграждения для лиц, нашедших утерянные вещи и вернувших их владельцам. | Механизм вознаграждения стимулирует честное поведение и способствует активному поиску и возврату утерянных предметов. |
| 42 | Система подтверждения владения | Механизм проверки правомерности владения утерянным предметом для исключения вымогательства. | Система подтверждения владения помогает снизить риск оспаривания собственности и обеспечивает правовую защиту. |
| 43 | Система автоматического уведомления | Автоматическая отправка уведомлений владельцам о нахождении их утерянных вещей. | Система автоматического уведомления повышает оперативность и эффективность в процессе возврата утерянных предметов. |
| 44 | Регистрация | Процесс создания учетной записи на онлайн бюро находок для доступа к функциям сайта. | Регистрация позволяет пользователям размещать объявления, взаимодействовать с другими пользователями и повысить эффективность поиска утерянных вещей. |
| 45 | Аутентификация | Процесс проверки подлинности пользователя при входе на сайт или при совершении определенных действий. | Аутентификация обеспечивает безопасность данных пользователей и защищает их от несанкционированного доступа. |
| 46 | Объявление | Информация, размещенная на онлайн бюро находок о потерянных или найденных вещах. | Объявления помогают связать владельцев утерянных вещей с находчиками, ускоряя возврат утерянных предметов. |
| 47 | Статистика | Анализ данных об объявлениях, поиске и возврате вещей для оценки успешности деятельности онлайн бюро находок. | Статистика позволяет выявить тенденции и улучшить эффективность работы сервиса по возврату утерянных вещей. |
| 48 | Статус | Пометка о состоянии объявления, указывающая на текущее положение вещи (найдено или потеряно). | Статус помогает пользователям отслеживать изменения в объявлениях и делать выводы о дальнейших действиях. |
| 49 | Местоположение | Определение географического положения утерянной вещи или места, где была найдена. | Знание местоположения помогает сузить область поиска и повысить шансы на успешное возвращение утерянной вещи владельцу. |
| 50 | Поиск | Функция, позволяющая пользователям искать утерянные вещи или найденные предметы на онлайн бюро находок. | Удобная система поиска увеличивает эффективность сервиса и помогает быстрее идентифицировать утерянные вещи. |
| 51 | Контактная информация | Данные, оставляемые в объявлениях для связи между утерявшим и нашедшим. | Правильно указанная контактная информация является ключевым элементом успешного возврата утерянных вещей и обеспечивает эффективное взаимодействие между пользователями. |
| 52 | Peer-to-peer | Способ взаимодействия пользователей, когда они обмениваются информацией или другими составляющими, без использования 3х лиц в виде администрации или работников предприятия | Такой способ взаимодействия, позволит снизить нагрузку на работников предприятия, а также увеличит эффективность взаимодействия пользователей |

## 1.3. Общие требования к проектируемой системе

***Архитектурные и эксплуатационные требования***

* Масштаб развертывания системы — локальная сеть учебного заведения или кафедры
* Приложение: двухуровневая клиент-серверная архитектура. Клиентские приложения пользователей развернуты на персональных компьютерах.
* Хранилище данных: централизованное, состоит из 1й базы данных, управляемое единственным сервером баз данных.
* Требования безопасности — реализация механизмов аутентификации и авторизации пользователей, а также управление правами доступа к функциональным возможностям приложения.

***Требования к функциональным характеристикам***

* Отображение находок: Приложение должно предоставлять возможность просмотра найденных и утерянных вещей для пользователей и модерации.
* Управление доступом пользователей к компонентам и информационным ресурсам системы.
* Поддержка актуальных находок: Приложение должно обеспечивать возможность формирования и редактирования заявок найденных и утерянных вещей, включая информацию о времени находки, данных заявителя, предположительном месте находки.
* Поддержка фильтрации по датам: Реализация функции фильтрации по датам, позволяет пользователям быстрее находить вещи.

## 1.4. Категории пользователей проектируемой системы

Схема взаимодействия пользователей системы с ее функциональными модулями, реализующими основные бизнес-процессы (табл. 1), приведена на обобщенной диаграмме вариантов использования (рис. 1).

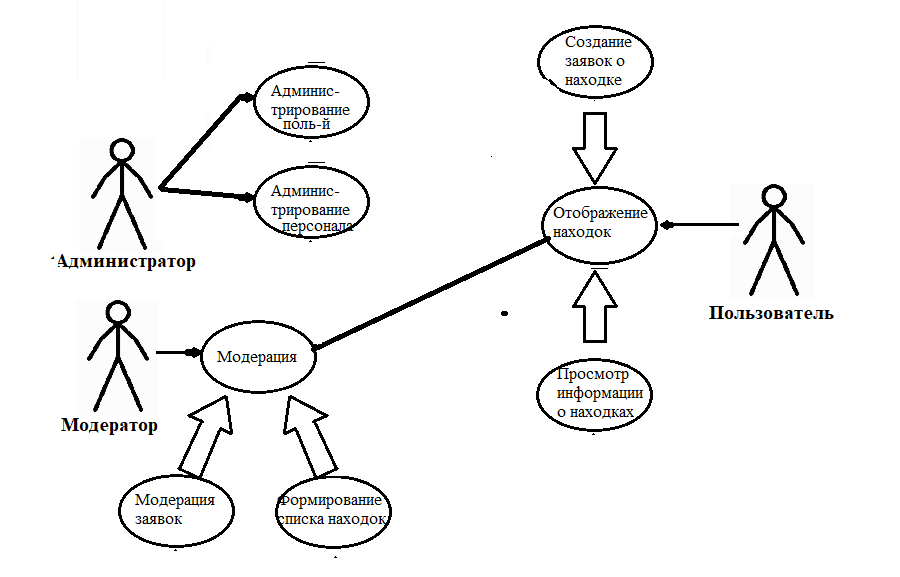


Рис. 1. Обобщенная UML-диаграмма вариантов использования

**Администратор** выполняет функции управления персоналом, имеет доступ к редактированию профилей модераторов и пользователей.

**Модератор** выполняет функции принятия заявок, производит модерацию заявок о находке, выполняет функции обновления и поддержания актуальности бюро находок,

**Пользователь** выполняет функции просмотра списка находок, имеет возможность составления заявки о находке или утери вещи.

# 2. МОДЕЛЬ ДАННЫХ СИСТЕМЫ

## 2.1 Выбранная модель данных системы

В данной работе используется уже ставшая классической модель Web-приложений на базе AJAX (см. рис. 1). AJAX — это аббревиатура, которая означает Asynchronous Javascript and XML. Основная идея AJAX – возможность для приложения получать информацию и отдавать ее другим компьютерам без перезагрузки веб-страницы.



Рис. 2. Модель Web-приложений на базе AJAX

Страница посредством JavaScript в браузере пользователя, по какому-либо событию с помощью объекта XMLHttpRequest подает запрос серверу, обращаясь к некоему серверному обработчику, передавая ему некоторые параметры. Обработчик принимает этот запрос и обрабатывает его. По завершению обработки данных, обработчик выдает эти данные обратно браузеру пользователя.

## 2.2 Преимущества выбранной системы

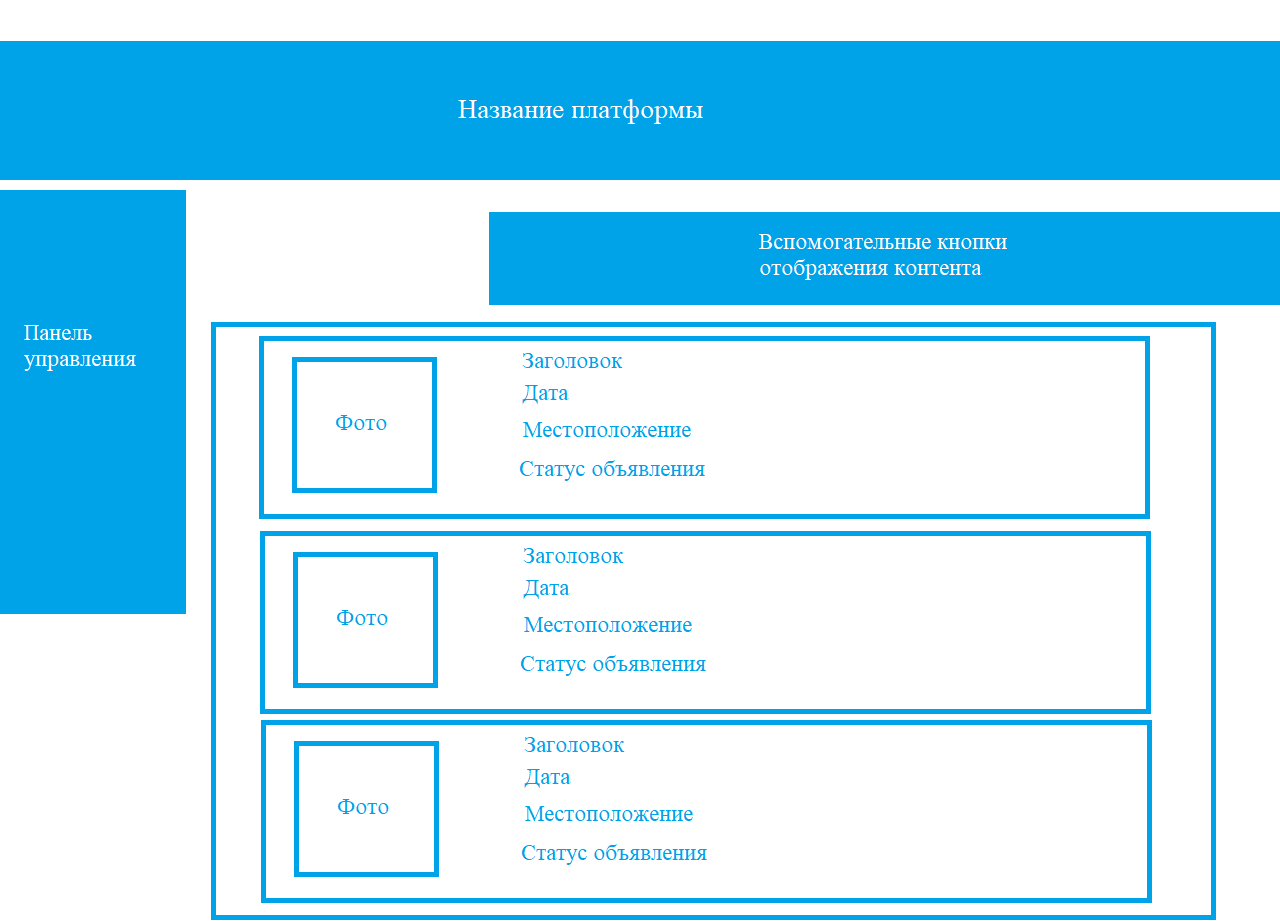
Обобщим достоинства AJAX:

* Возможность создания удобного Web-интерфейса
* Активное взаимодействие с пользователем
* Частичная перезагрузка страницы, вместо полной
* Удобство использования

# 3. МАКЕТЫ HTML-СТРАНИЦ СИСТЕМЫ

## 3.1 Макет главной страницы

Макет главной страницы разрабатываемого Интернет-ресурса, выполненный в приложении “Paint”, изображен на рис. 3а:

  
Рис. 3а. Макет главной страницы

## 3.2 Макет страницы личного кабинета

Макет страницы деталей объявления разрабатываемого Интернет-ресурса, выполненный в приложении “Paint”, изображен на рис. 3б:

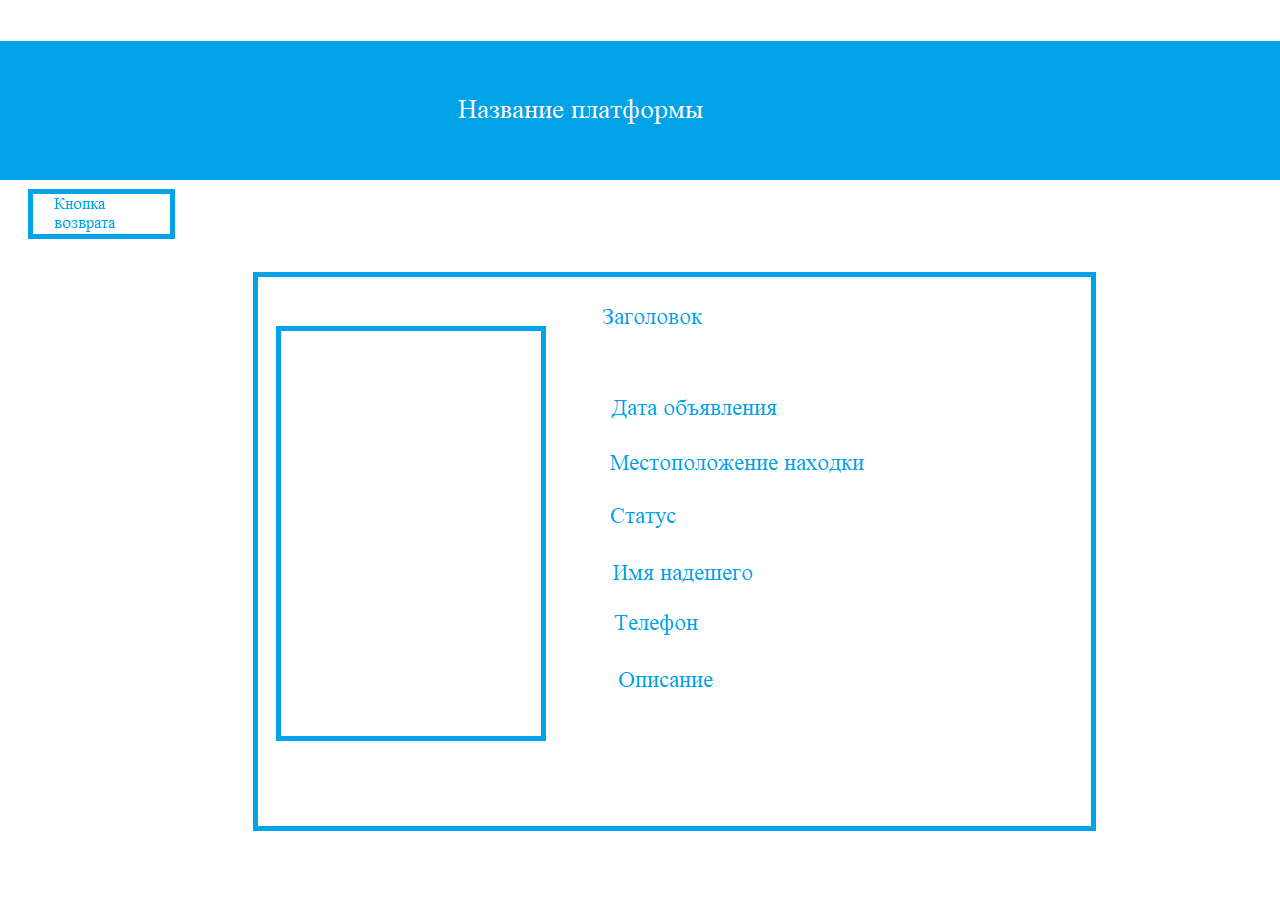
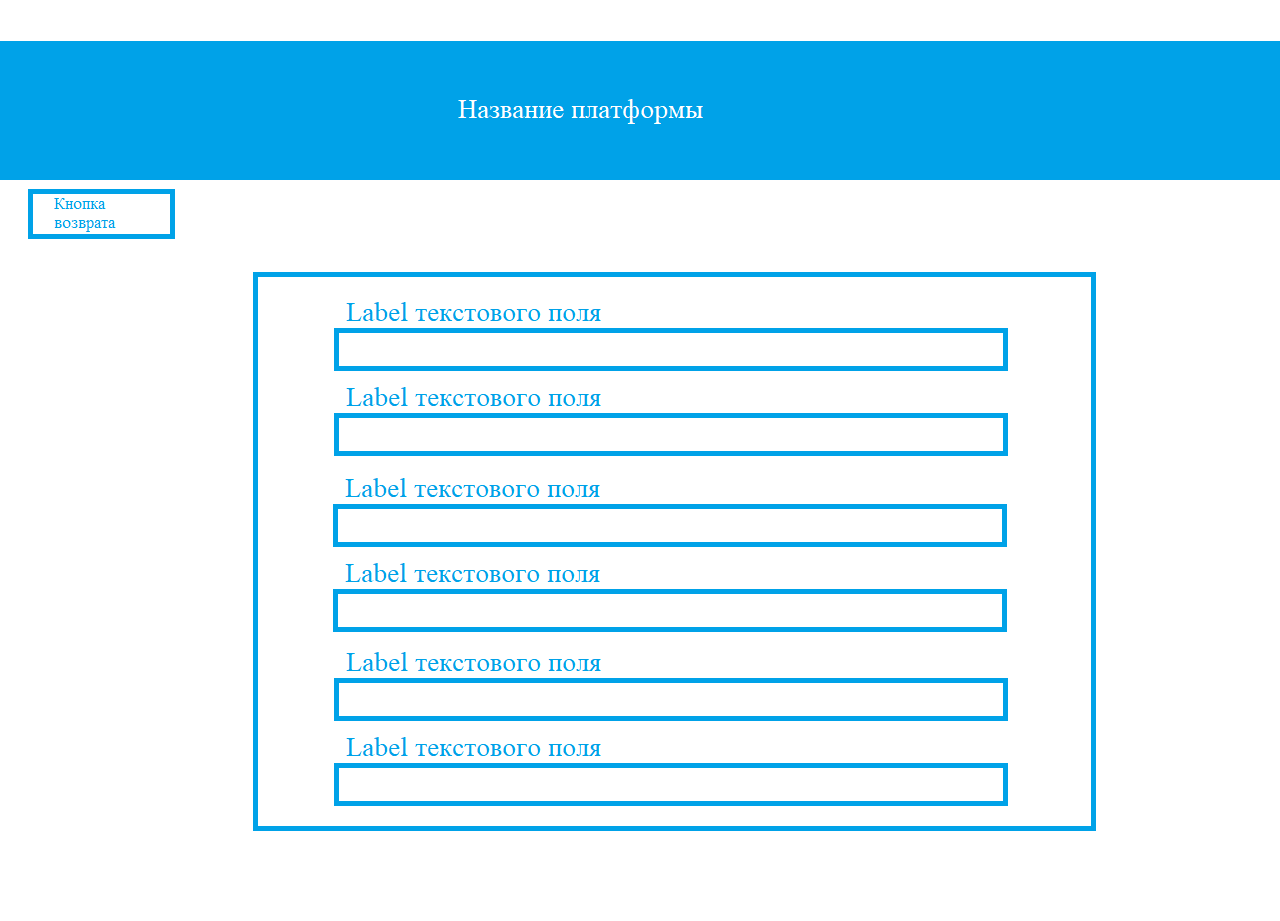


Рис. 3б. Макет страницы личного кабинета

Макеты страниц профиля, создания объявлений разрабатываемого Интернет-ресурса, выполненный в приложении “Paint”, изображен на рис. 3в:

  
Рис.3в Макет страниц профиля и создания объявления

# 4. СТРУКТУРНАЯ СХЕМА МОДУЛЕЙ ПО, СОСТАВЛЯЮЩИХ СИСТЕМУ

## 4.1 Структурная модель приложения. Модель MVC

Наша система использует шаблон «модель – представление – контроллер» (Model – View – Controller (MVC)) (см. рис. 4).

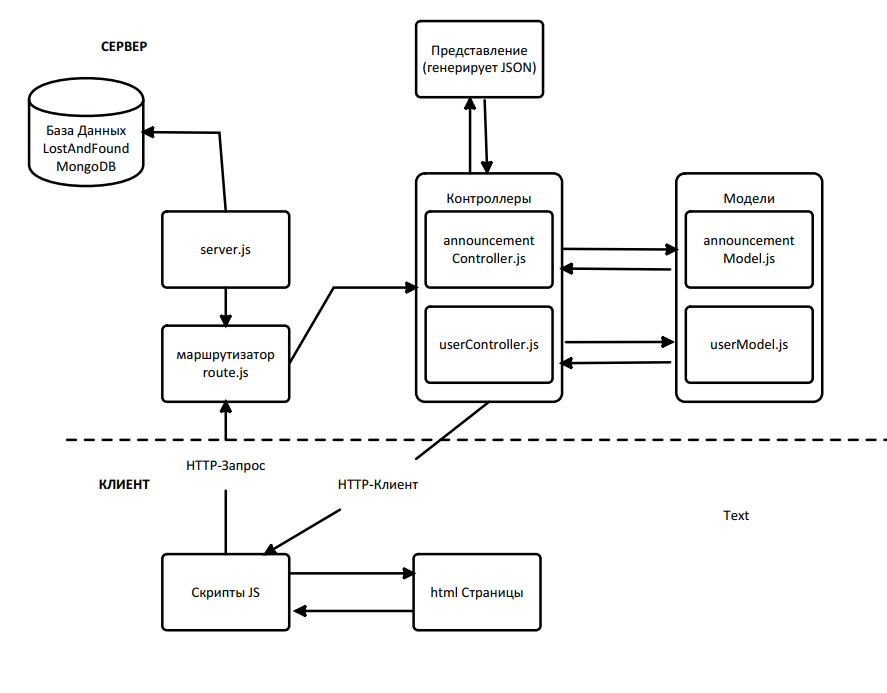


Рис. 4. Структурная схема приложения

Скрипты JS клиента (CreateUser.js, CreateAd.js, editAd.js, logreg.js, profDate.js) в зависимости от действий пользователя генерируют HTTP запросы.

Маршрутизатор – обрабатывает HTTP запрос с соответствующим действием контроллера.

Контроллер – преобразует запрос к серверу в действие. Как правило, соотносится с действием базы данных через модель. После этого отправляет ответ через представление.

Модель – это объектная абстракция элементов в базе данных, с ней взаимодействует контроллер.

Представление – в нашем случае это код HTML и CSS на клиентской стороне. Контроллер отправляет данные в виде JSON клиенту, а клиент решает, как их представить.

## 4.2 Структурная схема HTML страниц на стороне клиента

Структурная схема клиента, представленная HTML страницами, представлена на рис. 5а и 5б

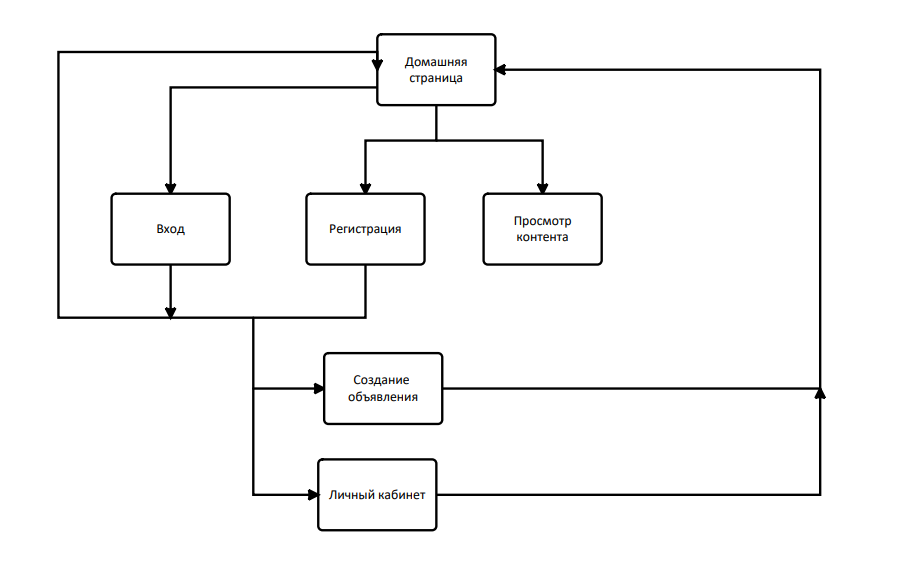


Рис. 5а. Структурная схема клиента

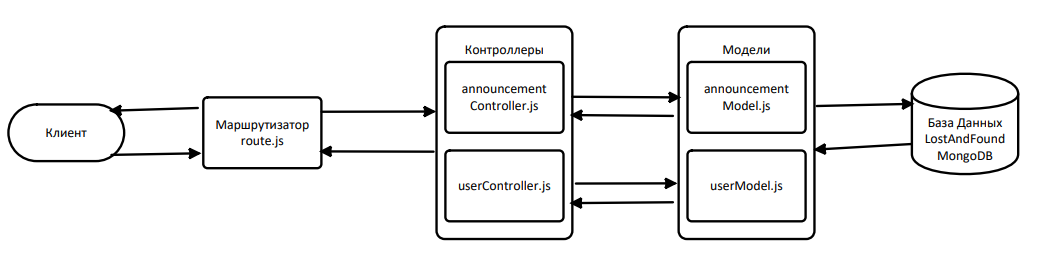


Рис.5б. Структурная схема клиента (с использованием MVC составляющих)

# 5. ДИАГРАММА КЛАССОВ РАЗРАБОТАННОЙ СИСТЕМЫ

Диаграмма классов разработанной системы изображена на рис. 6:

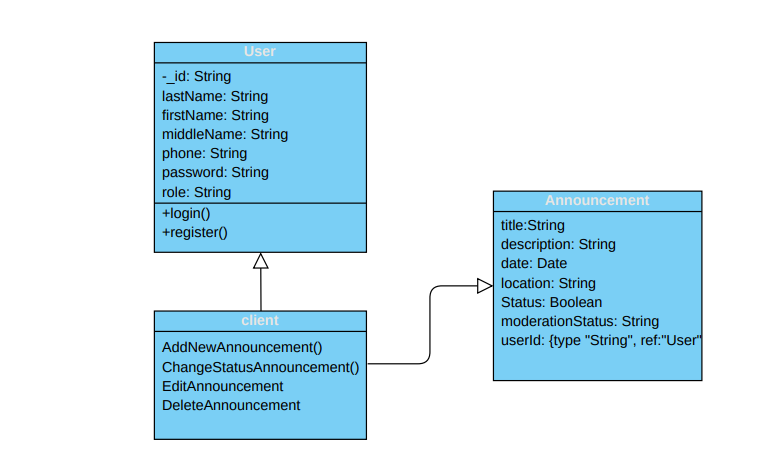


Рис. 6. Диаграмма классов разработанной системы

# 6. ОБЗОР ПО, ВЫБРАННОГО ДЛЯ РЕАЛИЗАЦИИ СИСТЕМЫ

## 6.1 Среда разработки – Visual Studio Code

Visual Studio Code (VS Code) — текстовый редактор, разработанный Microsoft для Windows, Linux и macOS. Позиционируется как «лёгкий» редактор кода для кроссплатформенной разработки веб- и облачных приложений.

Включает в себя :

отладчик

инструменты для работы с Git,

подсветку синтаксиса,

IntelliSense

Cредства для рефакторинга.

Имеет широкие возможности для кастомизации: пользовательские темы, сочетания клавиш и файлы конфигурации. Распространяется бесплатно, разрабатывается как программное обеспечение с открытым исходным кодом, но готовые сборки распространяются под проприетарной лицензией.

## 6.2 Веб-браузер Yandex Browser и инструменты разработчика

Яндекс.Браузер — браузер, созданный компанией «Яндекс» на основе движка Blink, используемого в открытом браузере Chromium.

Так как Яндекс.Браузер является родственным Chromium'у, ему по определению присуща значительная часть преимуществ и недостатков последнего. Из-за этого браузер слабо выделяется основной функцией на фоне других многочисленных браузеров на базе WebKit и Blink.

Инструменты разработчика позволяют быстро анализировать содержание и ресурсы веб-страницы.

## 6.3 Программная платформа Node.js

Node или Node.js — программная платформа, основанная на движке V8 (транслирующем JavaScript в машинный код), превращающая JavaScript из узкоспециализированного языка в язык общего назначения. Node.js добавляет возможность JavaScript взаимодействовать с устройствами ввода-вывода через свой API (написанный на C++), подключать другие внешние библиотеки, написанные на разных языках, обеспечивая вызовы к ним из JavaScript-кода. Node.js применяется преимущественно на сервере, выполняя роль веб-сервера, но есть возможность разрабатывать на Node.js и десктопные оконные приложения (при помощи NW.js, AppJS или Electron для Linux, Windows и macOS) и даже программировать микроконтроллеры (например, tessel, low.js и espruino). В основе Node.js лежит событийно-ориентированное и асинхронное (или реактивное) программирование с неблокирующим вводом/выводом.

В состав Node.js входит собственный установщик пакетов npm.

## 6.4 СУБД MongoDB

MongoDB — документоориентированная система управления базами данных с открытым исходным кодом, не требующая описания схемы таблиц. Классифицирована как NoSQL, использует JSON-подобные документы и схему базы данных. Написана на языке C++. Используется в веб-разработке, в частности, в рамках JavaScript-ориентированного стека MEAN.

Поддерживается JavaScript в запросах, функциях агрегации.

# 7. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

## 7.1 Руководство пользователя

Для запуска программы необходимо в адресной строке веб-браузера необходимо ввести адрес сайта «База Данных» и нажать клавишу Enter. Будет запущена главная страница сайта (см. рис. 7).

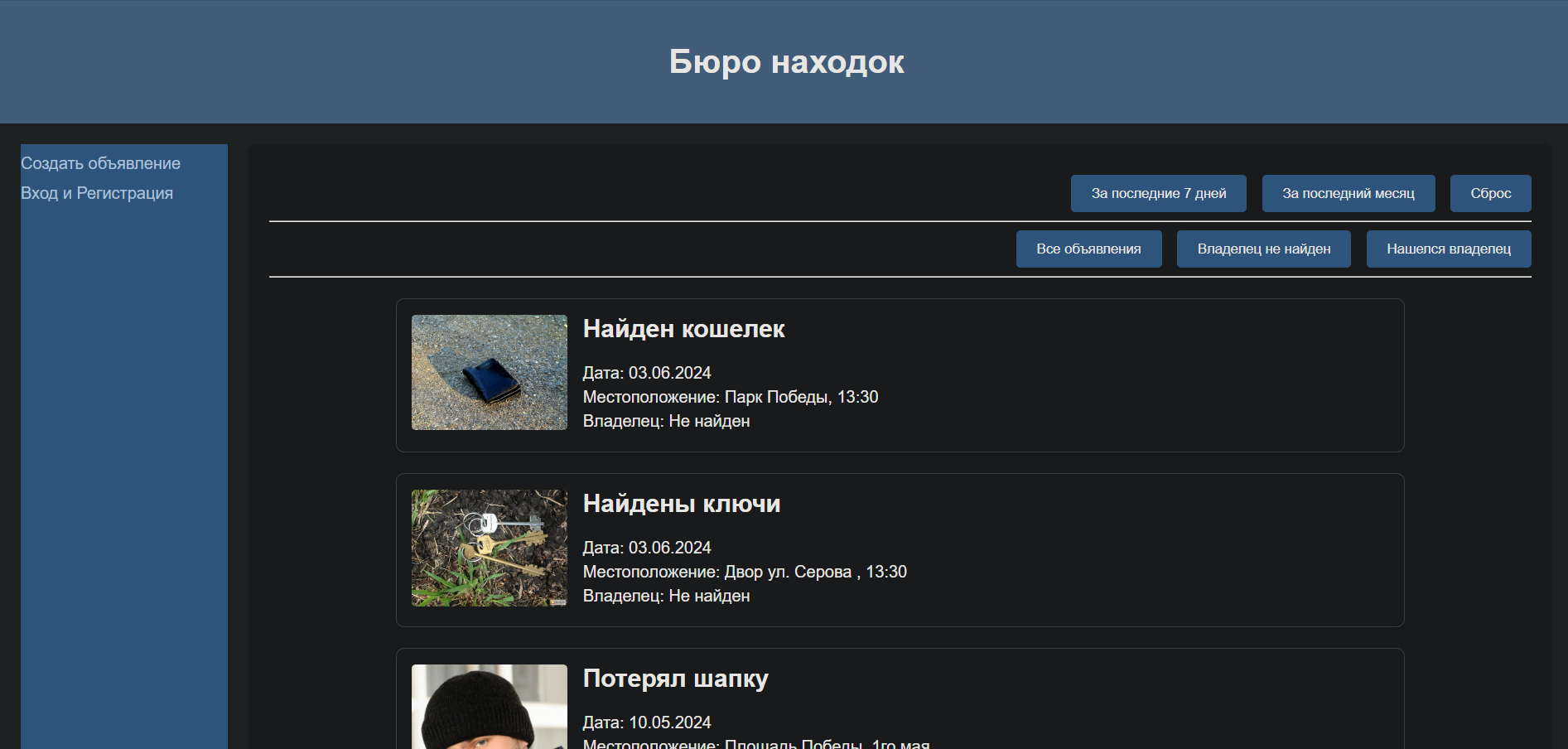


Рис. 7. Главная страница сайта

В «шапке» страницы изображена название «Бюро Находок»

В центральной части страницы находятся объявления о пропаже или находке. Вы также их можете отсортировать путем нажатия кнопок в правой верхней части экрана.

В левой части экрана находятся кнопки управления. Одна из таких кнопка Входа и Регистрации.

При нажатии на кнопку Входа и Регистрации вы сможете выбрать, что именно вы бы хотели сделать. Окно входа представлено на рисунке 8а.

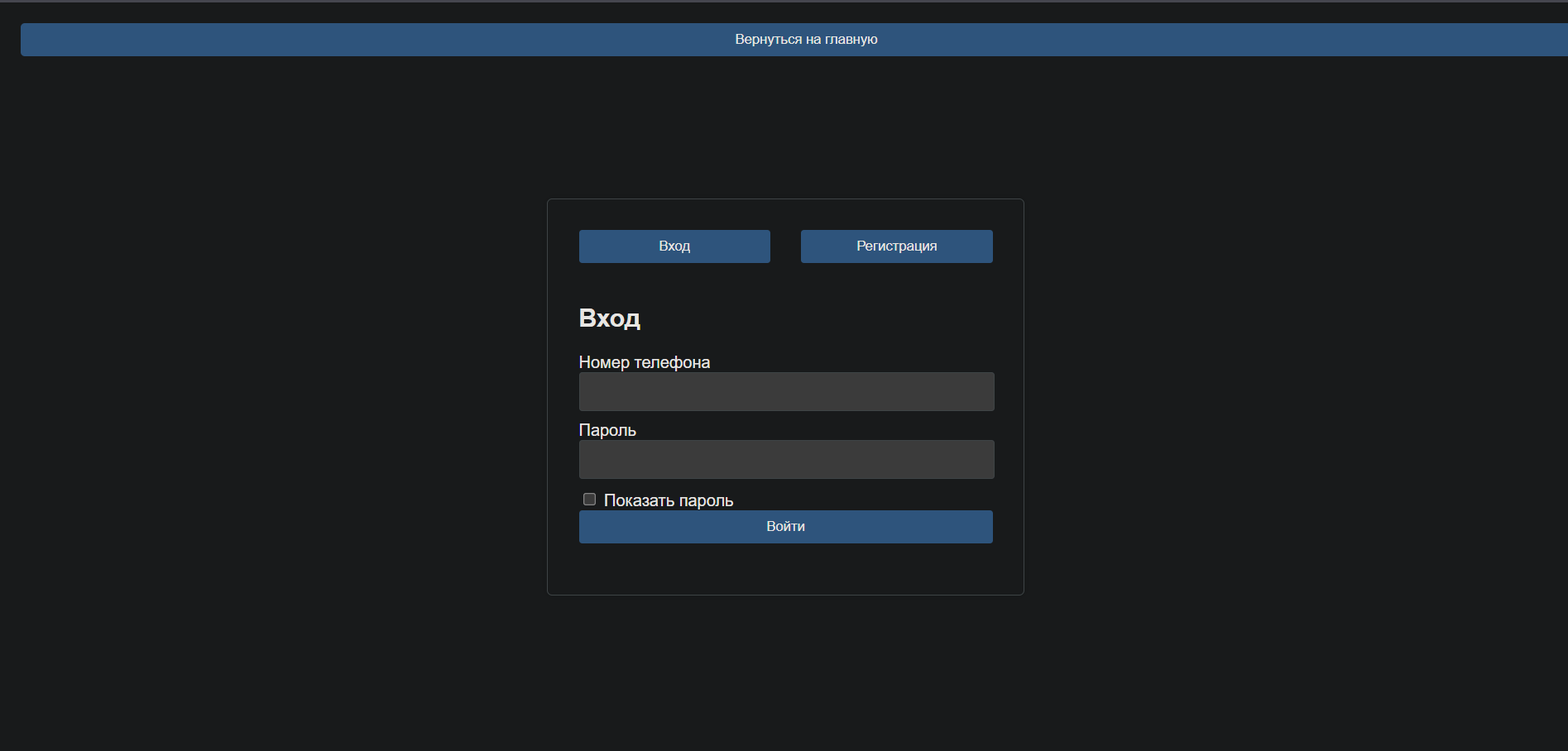
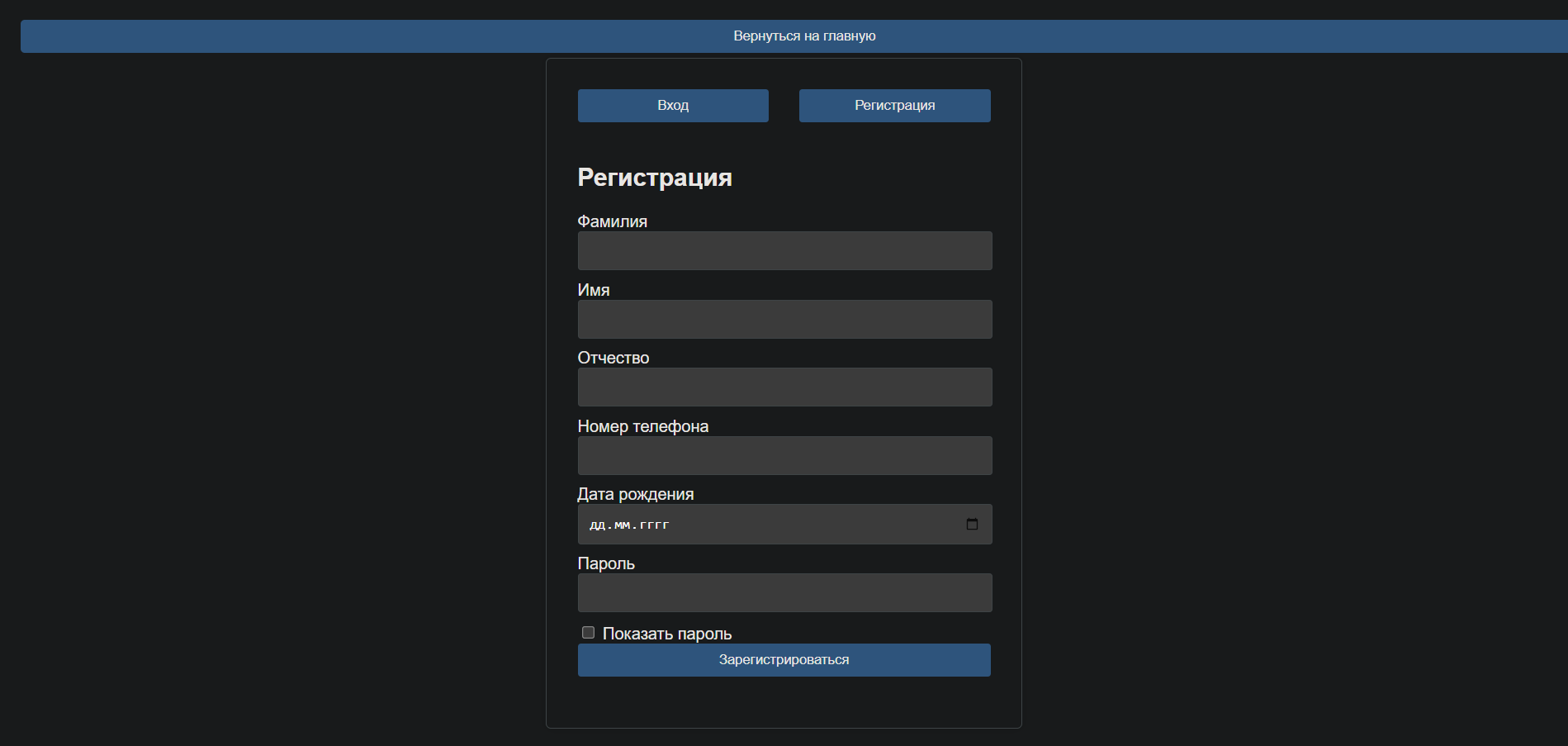


Рис. 8а. Окно входа

При необходимости вы можете переключится на окно регистрации нажав на кнопку «Регистрация» (рис. 8б)

  
Рис. 8б. Окно регистрации

При успешном входе или регистрации вам придет уведомление от сайта

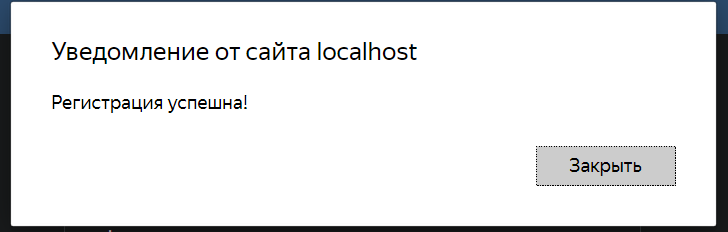


Рис. 9. Уведомление при регистрации на сайта

Если же вы уже ранее регистрировались в системе сайта, вам нужно будет нажать на кнопку "войти", после чего вы окажетесь на главной странице.

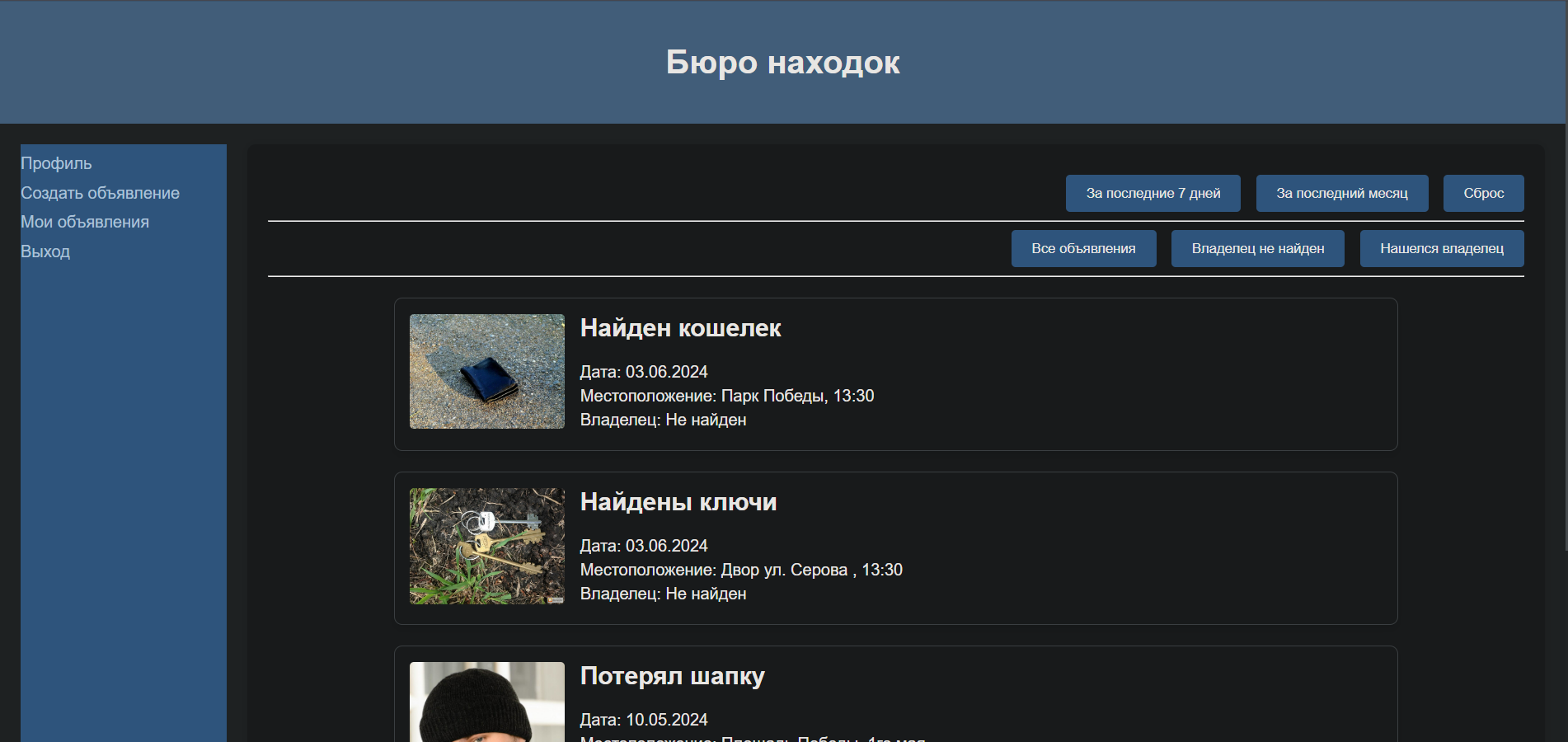
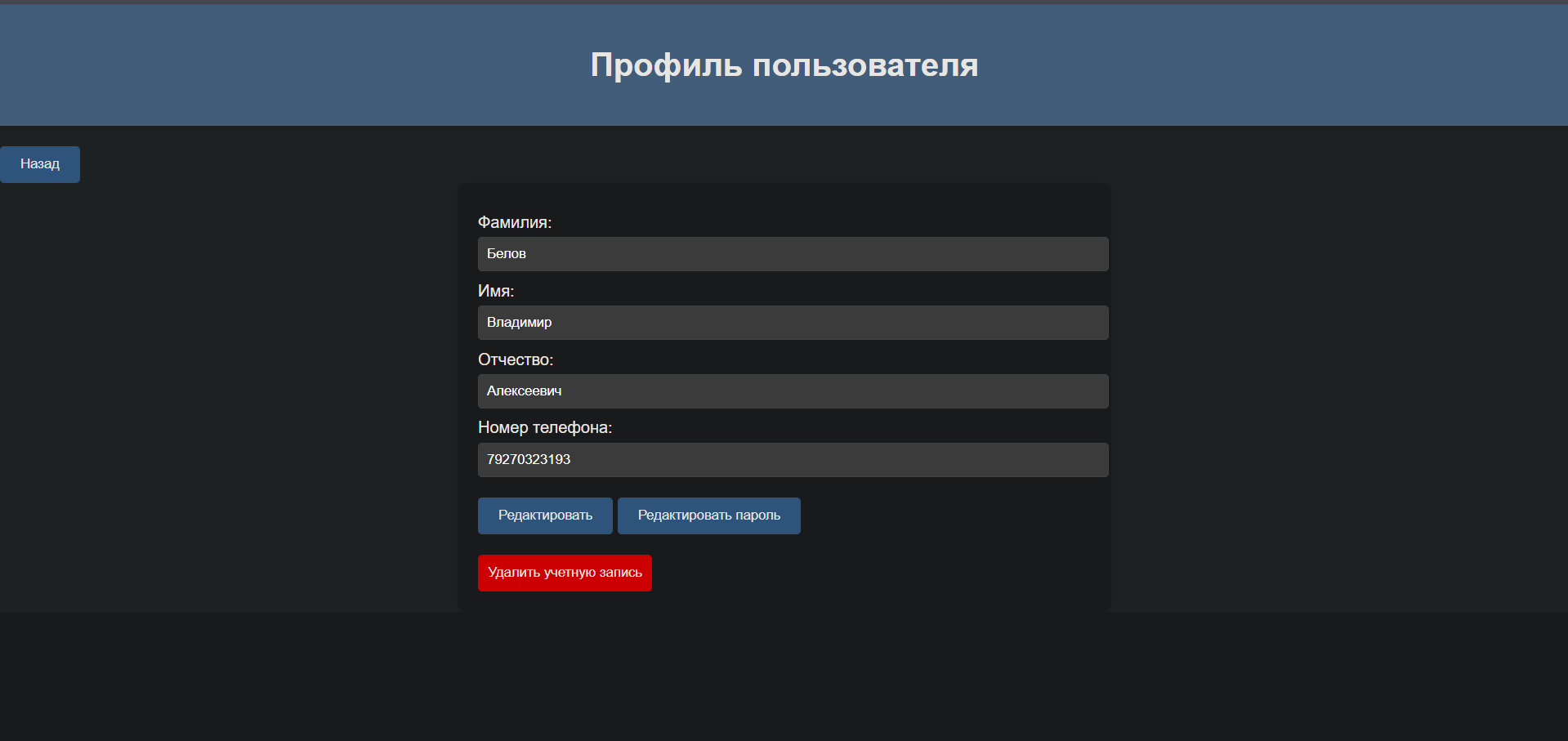


Рис.10. Отображение новых вкладок при входе

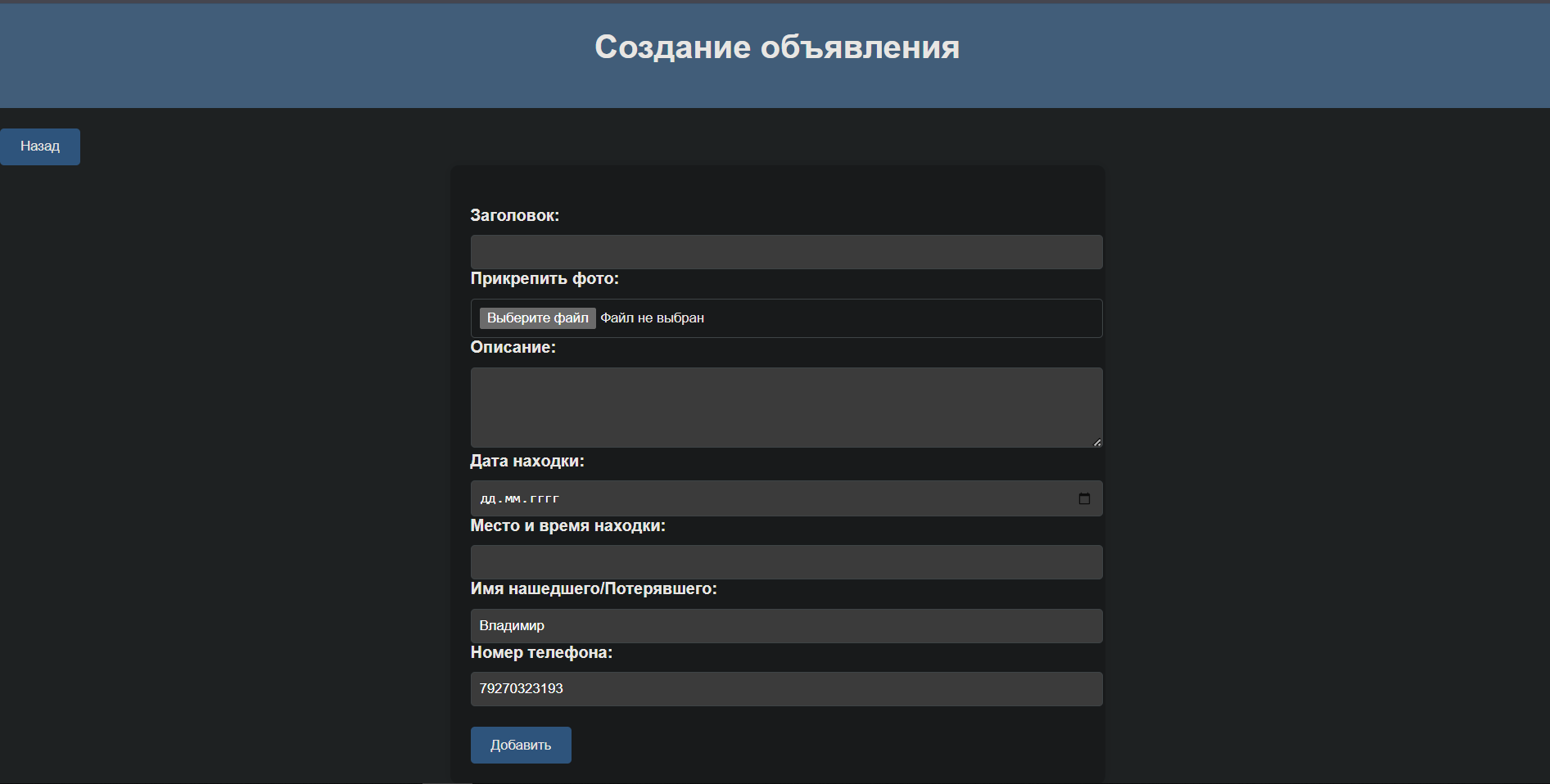
Также вы можете посмотреть и поменять данные своего аккаунта, нажав на кнопку Профиля (рис. 11). Вам будет представлена возможно редактировать данные, изменить пароль и удалить учетную запись.



11. Страница личного кабинета

После того, как будет выполнен вход помимо личного кабинета вам будет доступна вкладка Создание объявлений и мои объявления.

При создании объявления вы сможете указать разные необходимые данные о находке или пропаже, а также прикрепить фото. (рис. 12)

  
Рис. 12. Окно создания объявления

После создания объявления, оно должно пройти модерацию, чтобы успешно отобразиться. Это вы можете отследить на вкладке «Мои объявления» (Рис. 13)

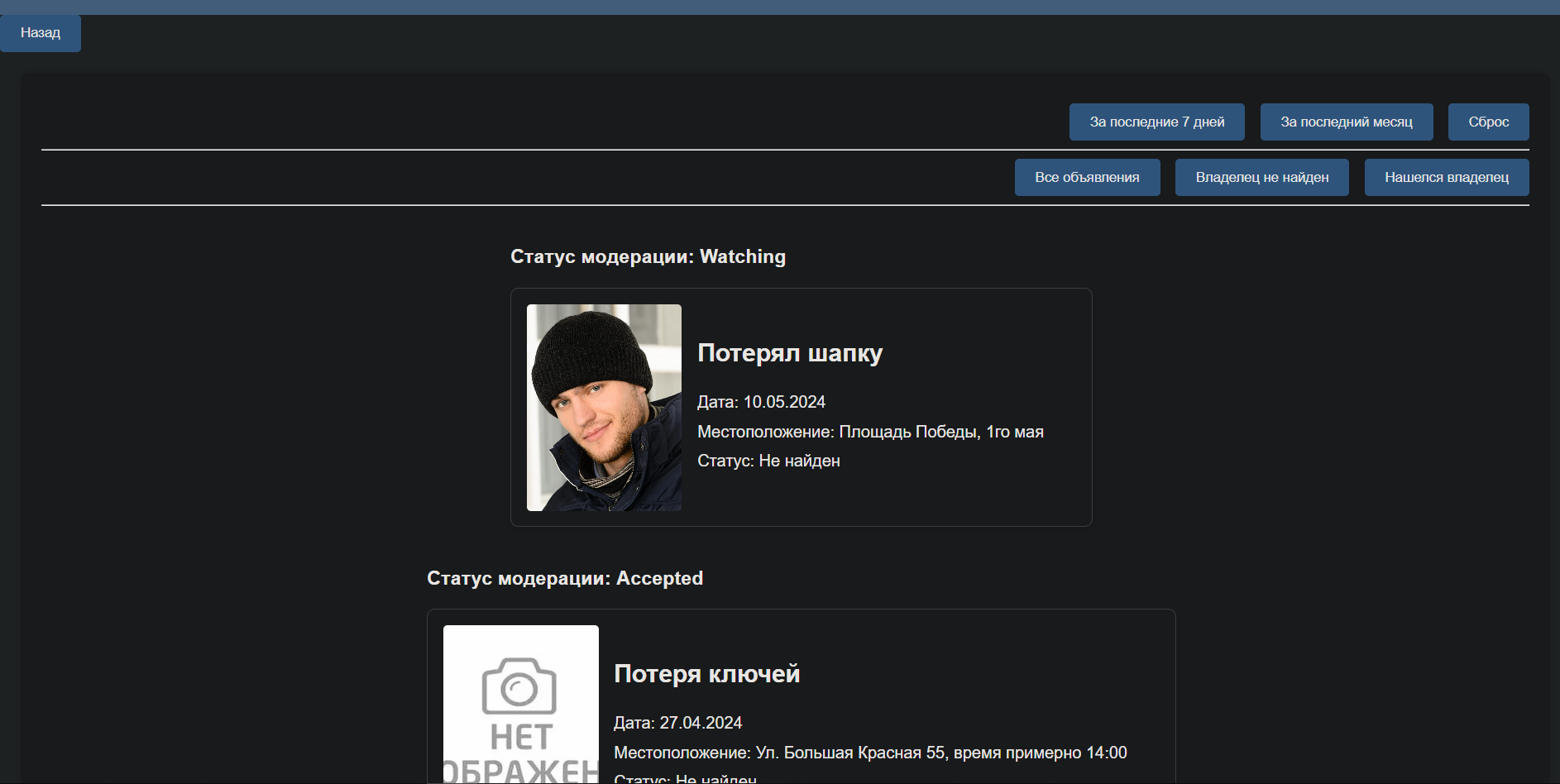
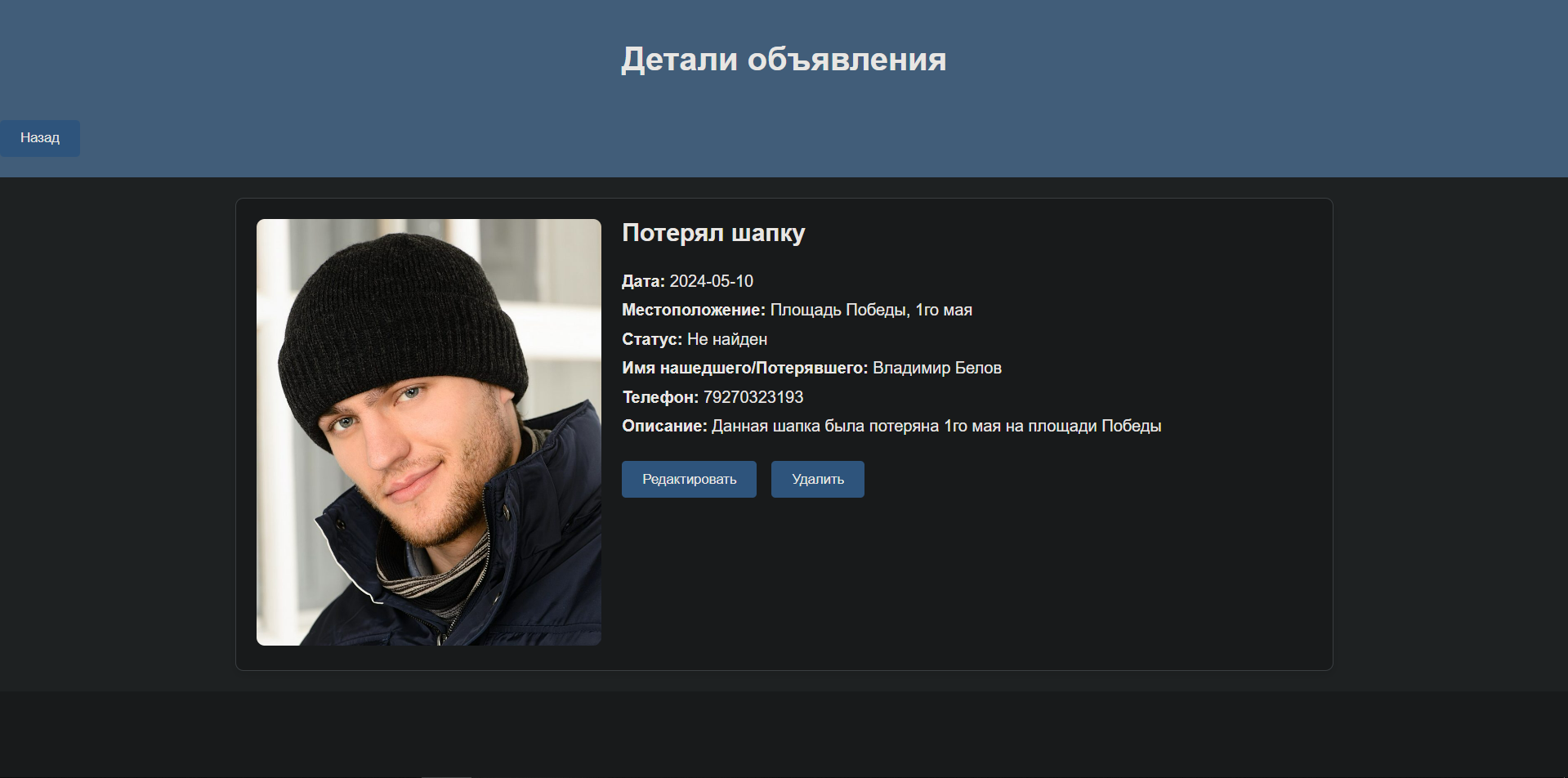


Рис. 13 Окно «Мои объявления»

На вкладке с моими объявлениями (рис.13) вы сможете посмотреть детали объявления, а также редактировать его или удалить (рис.14).

  
Рис.14. Редактирование объявления

На основной странице (рис.10) при нажатии на объявление, вы сможете просмотреть его детали (Рис. 15)

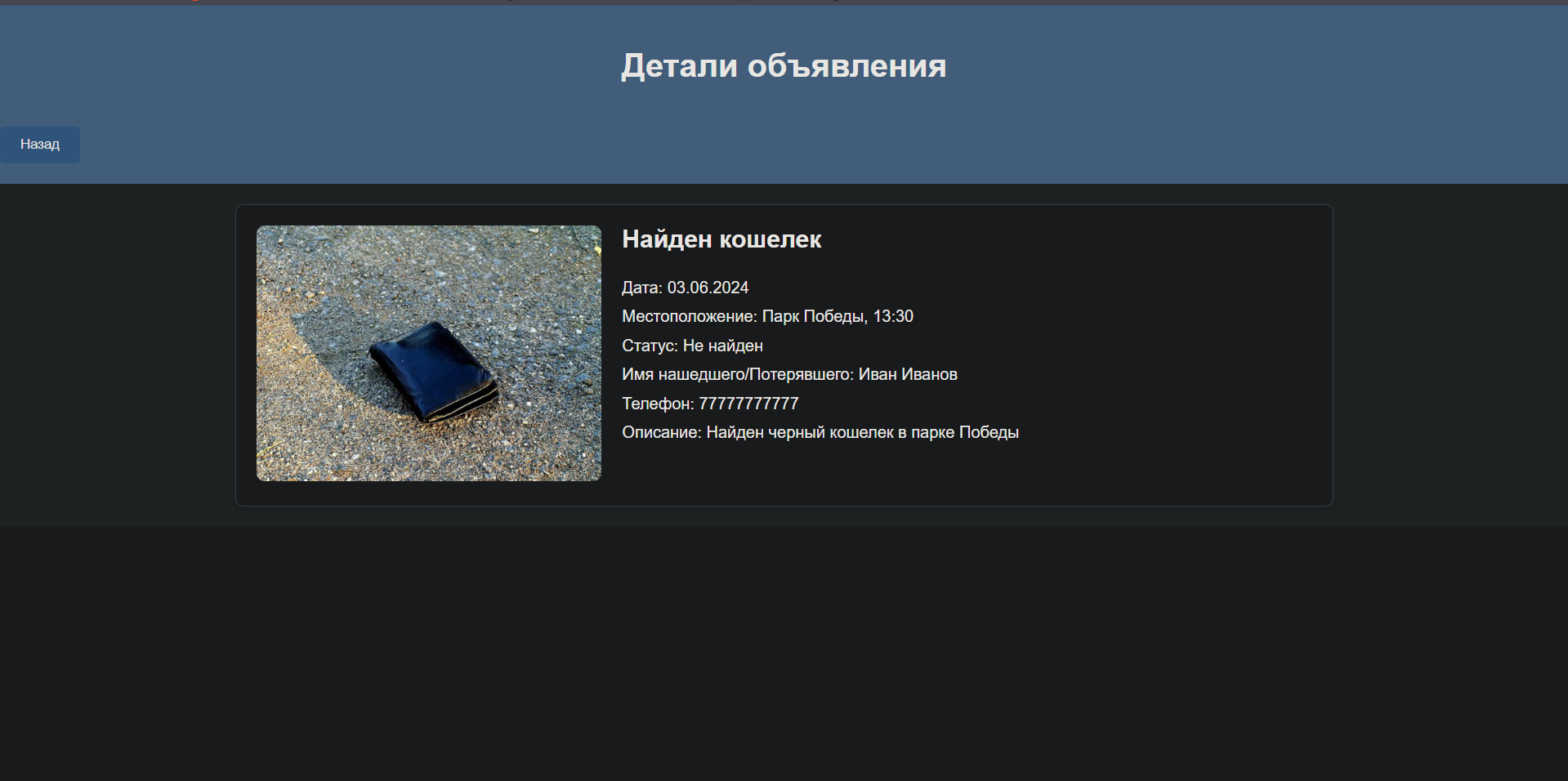


Рис. 15. Просмотр деталей объявления

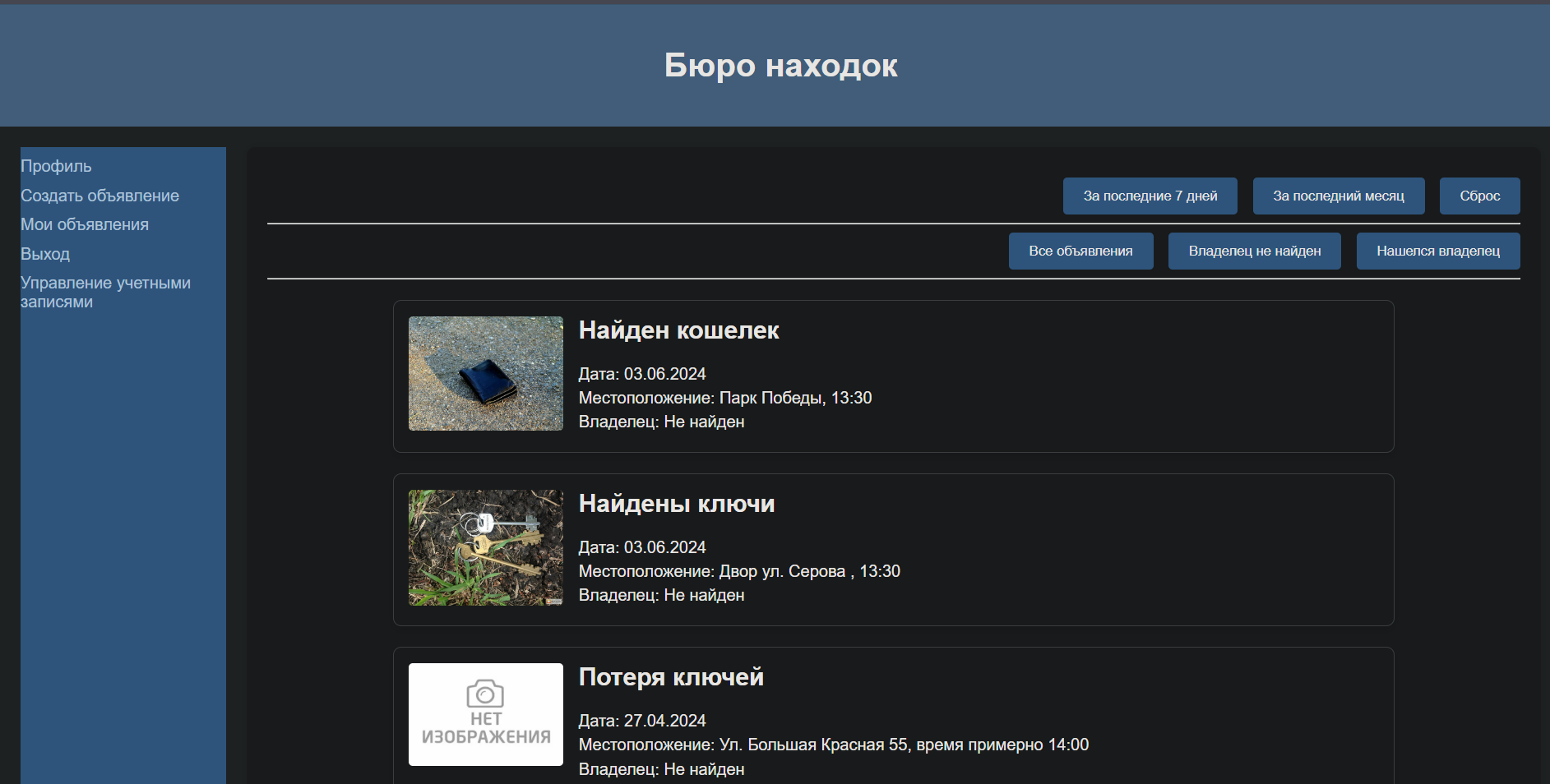
Нажав на кнопку «Выход» из главного меню, вы сможете выйти из вашей учетной записи.

## 7.2 Руководство для работников

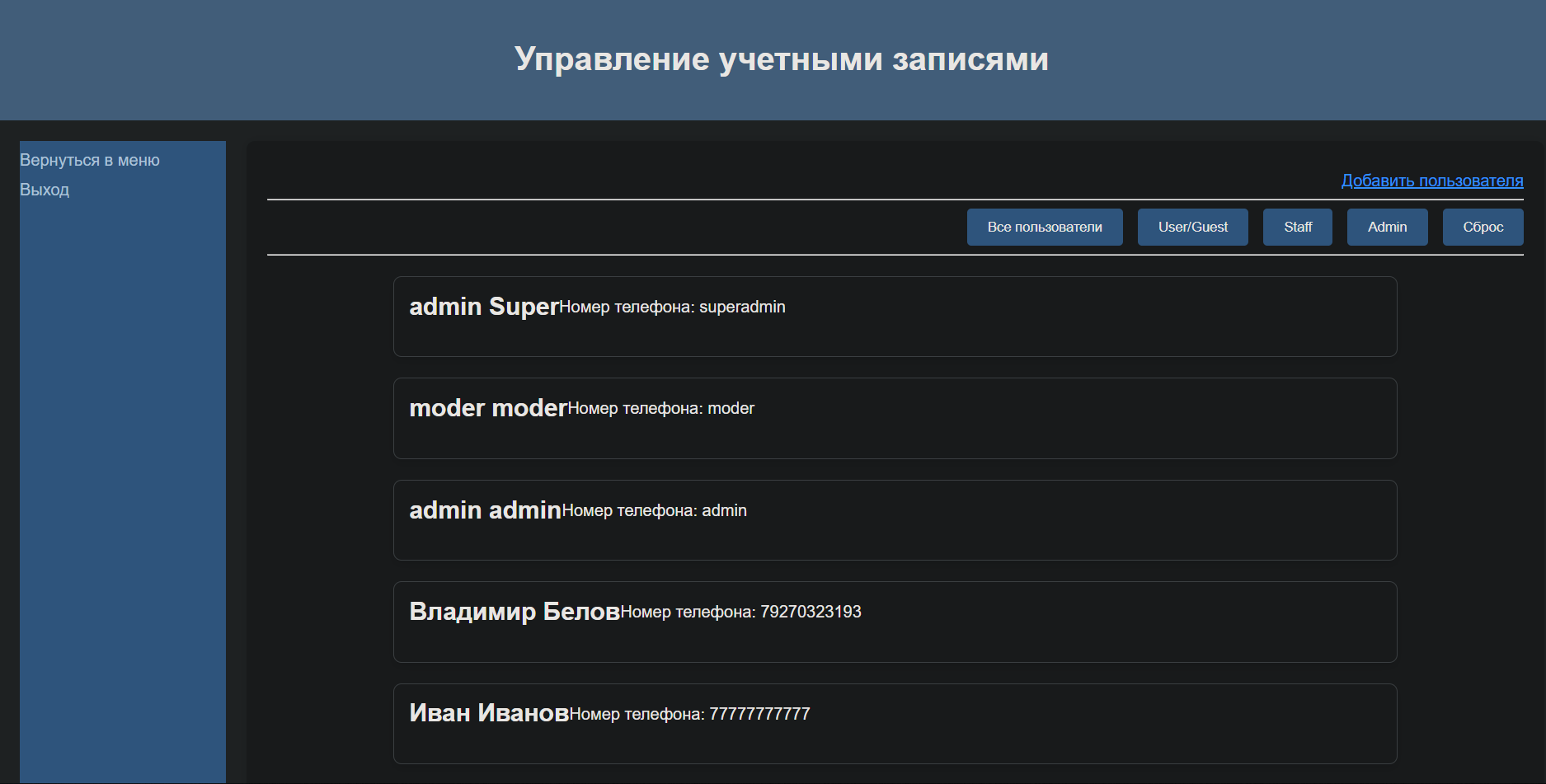
Далее будет описано руководство для работников. Каждый пользователь имеет роль:

* Superadmin – может создавать пользователей с ролью «admin, moder, user», а также редактировать и удалять учетные записи
* admin - может создавать пользователей с ролью «moder, user», а также редактировать и удалять учетные записи, которые обладают данными ролями
* moder - может принимать или отклонять полученные объявления

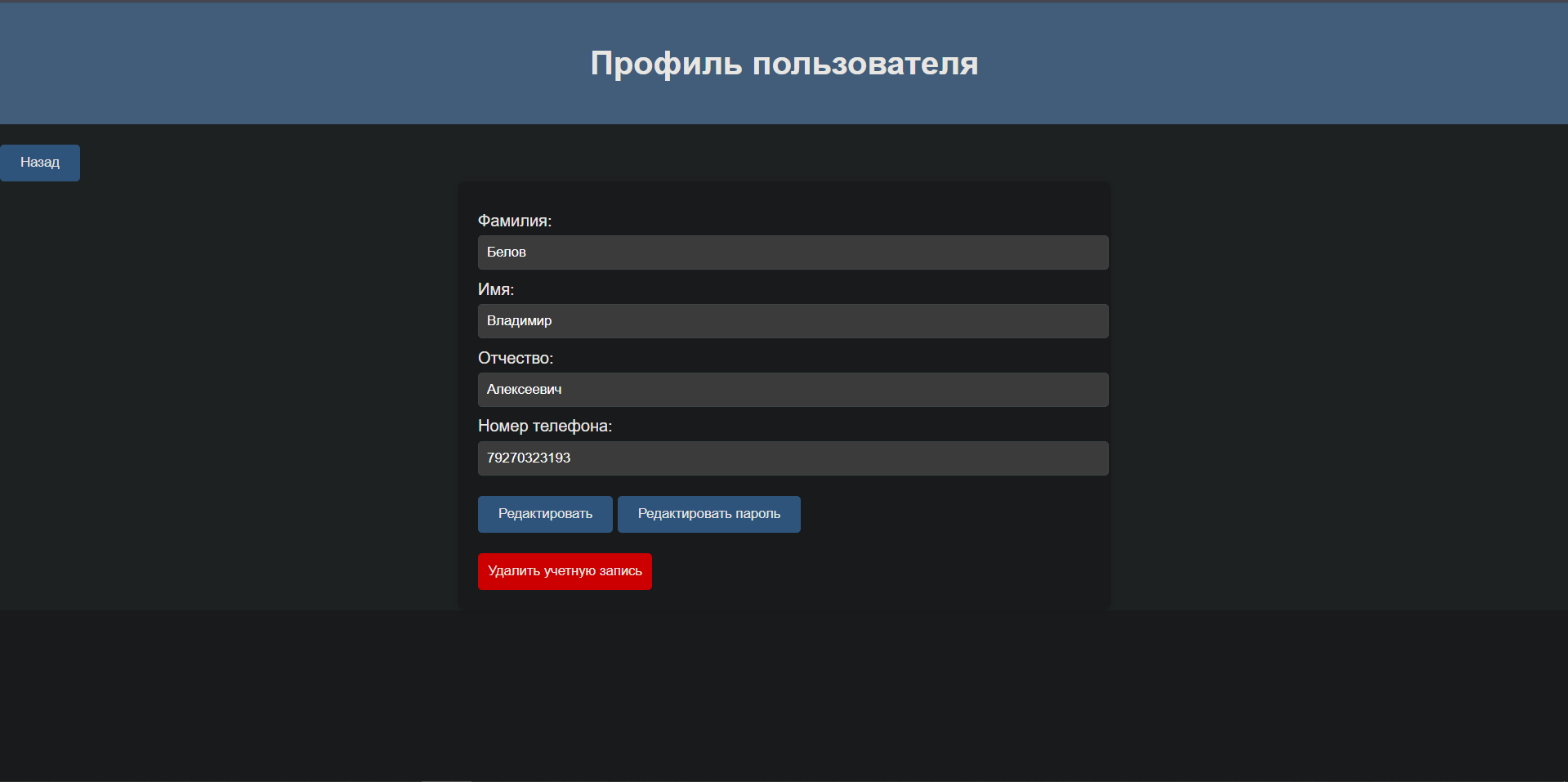
При входе в учетную запись с ролью superadmin или admin. Вам будет доступна новая кнопка «управление учетными записями» (Рис. 16)

  
Рис. 16. Окно входа с ролью «admin»

При нажатии на кнопку «Управление учетными записями» вам будет представлено окно (рис. 17) со списком пользователей, который вы сможете отсортировать по роли.

  
Рис. 17. Окно «Управление учетными записями»

При нажатии на пользователя, вы сможете просмотреть его данные, а также удалить, либо редактировать их (рис. 18)

  
Рис.18. Окно редактирования данных пользователя

При входе пользователя с ролью «moder» вам будет доступна кнопка «Моедрация объявлений» (рис. 19)

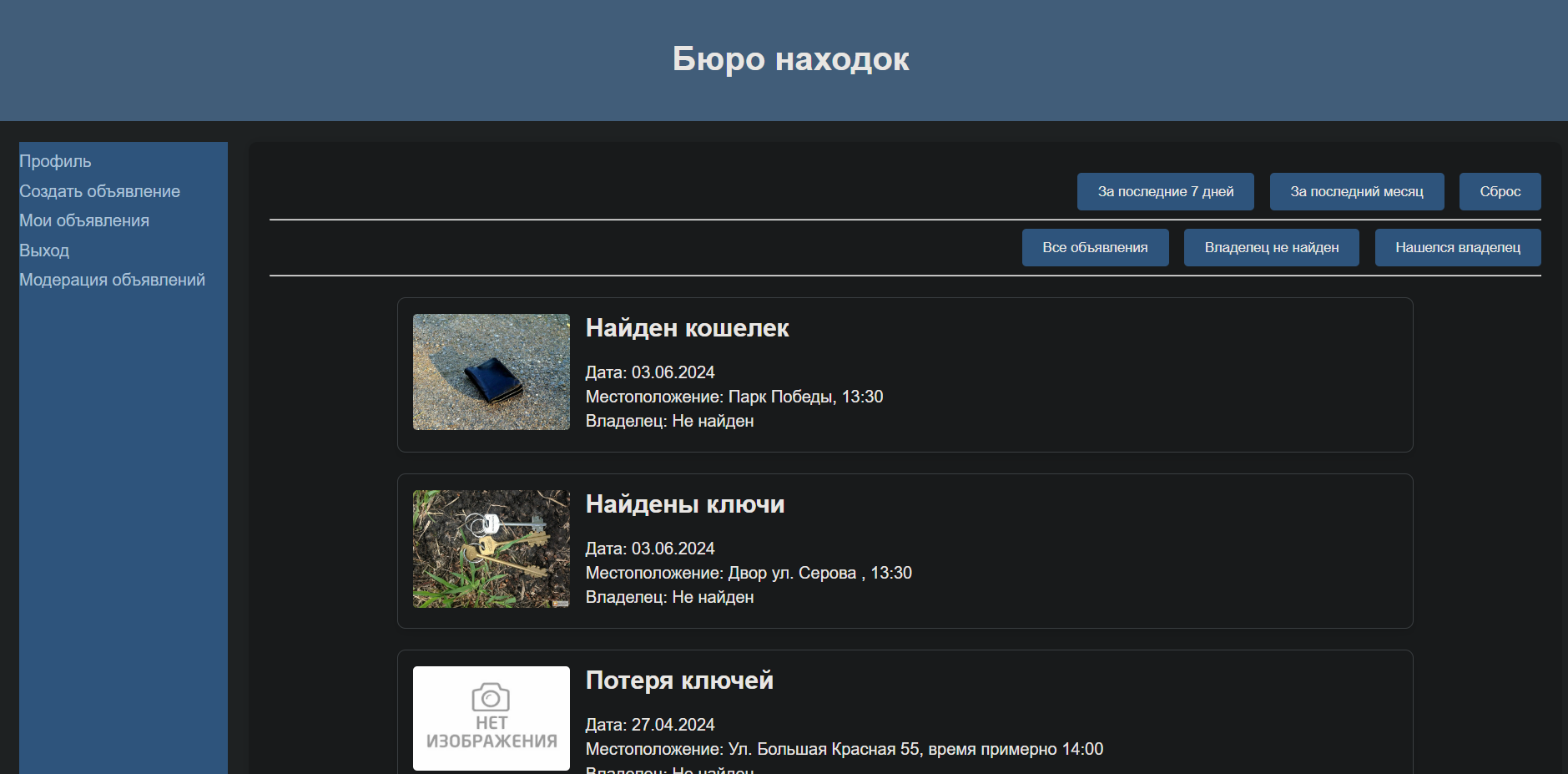
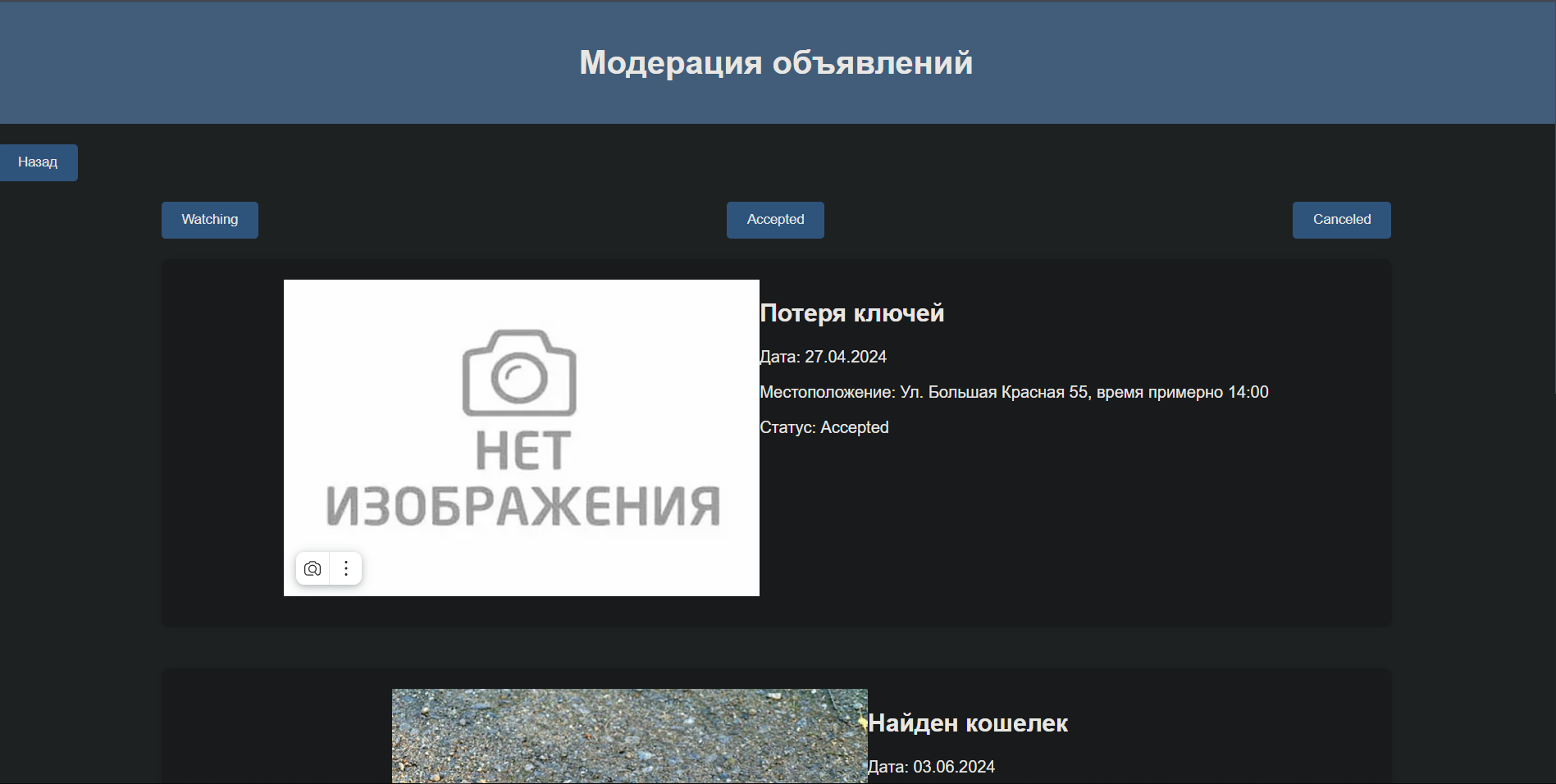
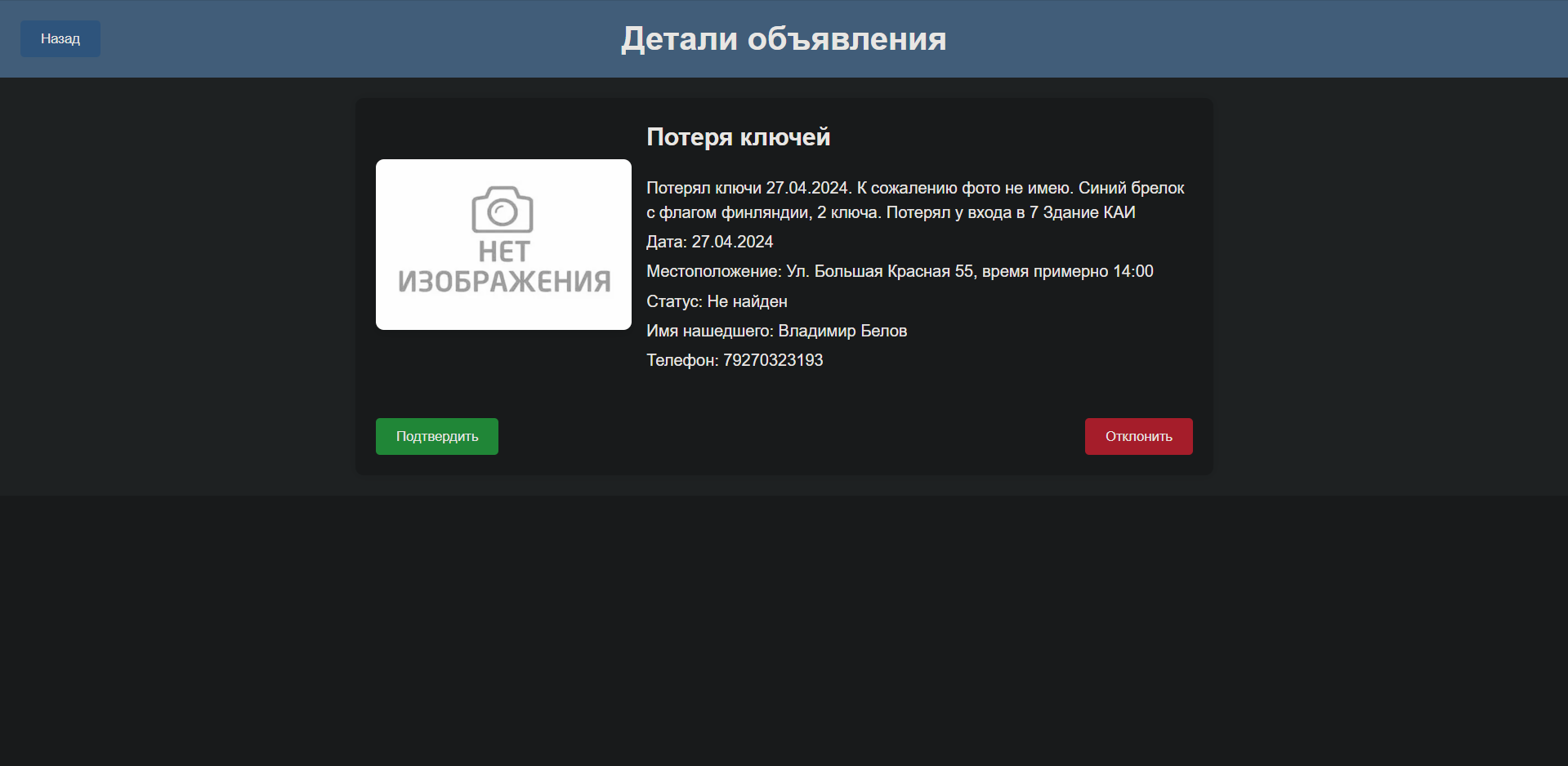


Рис. 19. Вход с ролью «moder»

При нажатии на данную кнопку вы увидите список объявлений на рассмотрении. Также вы сможете просмотреть список принятых и отклоненных объявлений. (Рис. 20)

  
Рис. 20. Окно модерации объявлений

При нажатии на объявлений вам откроются детали объявлений, а также будут доступны кнопки принятия и отклонения объявления

  
Рис. 21 Окно принятия или отклонения объявления

# ИСПОЛЬЗОВАННЫЕ ИСТОЧНИКИ

1. Справочник по HTML. // HTMLBOOK [Электронный ресурс]. – Режим доступа https://htmlbook.ru/html. – (Дата обращения: 5.06.2024).

2. Руководство по Node Js . // METANIT – сайт о программировании [Электронный ресурс]. - Режим доступа https://metanit.com/web/nodejs/ – (Дата обращения: 5.06.2024).

3. Онлайн-руководство по MongoDB. // METANIT – сайт о программировании [Электронный ресурс]. – Режим доступа: https://metanit.com/nosql/mongodb/. – (Дата обращения: 6.06.2024)

4. AJAX для новичков // Статья. – Режим доступа: https://habr.com/ru/articles/14246/. – (Дата обращения: 05.06.2024)

5. Model-View-Controller // Википедия [Электронный ресурс]. Дата обновления: 29.01.2020. – Режим доступа: https://ru.wikipedia.org/wiki/Model-View-Controller (дата обращения: 05.06.2024).

6. JQuery // Википедия [Электронный ресурс]. - Режим доступа: https://ru.wikipedia.org/wiki/JQuery (дата обращения: 05.06.2024).   
7. Online Visual Paradigm // Сайт для создания диаграмм - Режим доступа: https://online.visual-paradigm.com (дата обращения: 05.06.2024).  
8. Aspose // Сайт для создания схем - Режим доступа: https://products.aspose.app/diagram (дата обращения: 05.06.2024).

# ПРИЛОЖЕНИЕ. Ресурсы с исходным кодом

Весь код будет представлен на электронном ресурсе GitHub <https://github.com/belovVA/Web-Kyrsach> .Ниже представлен код сервера, моделей, контроллеров и основной страницы

# ПРИЛОЖЕНИЕ 1.Файл server.js

const express = require('express');

const mongoose = require('mongoose');

const bodyParser = require('body-parser');

const path = require('path');

const fs = require('fs');

const routes = require('./routes/routes');

const app = express();

const PORT = 3000;

// MongoDB connection

mongoose.connect('mongodb://localhost:27017/lostAndFound', { useNewUrlParser: true, useUnifiedTopology: true });

// Middleware

app.use(bodyParser.json({ limit: '50mb' }));

app.use(bodyParser.urlencoded({ limit: '50mb', extended: true }));

app.use('/uploads', express.static(path.join(\_\_dirname, 'uploads')));

// Serve static files from FrontEnd directory

app.use(express.static(path.join(\_\_dirname, '../client')));

// Serve uploads directory

// Routes

app.use('/', routes);

// Start the server

app.listen(PORT, () => {

  // console.log(`Server is running on port ${PORT}`);

});

# ПРИЛОЖЕНИЕ 2.Файл routes.js

const express = require('express');

const fs = require('fs');

const path = require('path');

const multer = require('multer');

const userController = require('../controllers/userController');

const announcementController = require('../controllers/announcementController');

const router = express.Router();

const uploadsDir = path.join(\_\_dirname, '../uploads');

if (!fs.existsSync(uploadsDir)) {

    fs.mkdirSync(uploadsDir);

}

const storage = multer.diskStorage({

  destination: function (req, file, cb) {

    cb(null, uploadsDir);

  },

  filename: function (req, file, cb) {

    const ext = path.extname(file.originalname);

    cb(null, `${Date.now()}${ext}`);

  }

});

const upload = multer({ storage: storage });

// User routes

router.post('/register', userController.register);

router.post('/login', userController.login);

router.get('/profile/:userId', userController.getProfile);

router.put('/profile/:userId', userController.updateProfile);

router.delete('/profile/:userId', userController.deleteProfile);

router.post('/usersNew', userController.userNewCreate);

router.get('/users', userController.usersList);

router.get('/usersByRole',userController.sortedUserByRole);

// Announcement routes

router.post('/createAd', announcementController.createAd);

router.get('/adDetail', announcementController.getAdDetail);

router.put('/updateAd', announcementController.updateAd);

router.delete('/deleteAd', announcementController.deleteAd);

router.get('/user-ads', announcementController.getUserAds);

router.get('/ads', announcementController.getAds);

router.post('/updateAdStatus', announcementController.updateAdStatus);

router.get('/adsModeration', announcementController.getAdsModeration);

// Photo upload route

router.post('/uploadPhoto', upload.single('photo'), (req, res) => {

  if (!req.file) {

    return res.status(400).send('Не удалось загрузить файл');

  }

  const photoUrl = req.file.filename;

  res.json({ photoUrl });

});

router.use(express.static(path.join(\_\_dirname, '../client/')));

// Routes for serving HTML files

const setRoute = (routePath, filePath) => {

  router.get(routePath, (req, res) => {

    res.sendFile(path.join(\_\_dirname, '..', 'client', filePath));

  });

};

setRoute('/adDetail', 'AdDetail/adDetail.html');

setRoute('/admin/createUser', 'admin/createUser.html');

setRoute('/admin/manageAccounts', 'admin/manageAccounts.html');

setRoute('/advert/createAd', 'Advert/createAd.html');

setRoute('/editAd', 'EditAd/editAd.html');

setRoute('/login', 'LoginOrRegistration/login.html');

setRoute('/moderator/checkAds', 'moderator/checkAds.html');

setRoute('/moderator/moderationAds', 'moderator/moderationAds.html');

setRoute('/myAds', 'MyAds/myAds.html');

setRoute('/profDate', 'profDate/profDate.html');

setRoute('/main', 'main/main.html', 'main/main.css', 'main/main.js');

module.exports = router;

# 

# ПРИЛОЖЕНИЕ 3.Файл announcementModel.js

const mongoose = require('mongoose');

const announcementSchema = new mongoose.Schema({

    title: String,

    photoUrl: String,

    description: String,

    date: Date,

    location: String,

    userId: { type: mongoose.Schema.Types.ObjectId, ref: 'User' },

    status: { type: Boolean, default: false },

    moderationStatus: {

      type: String,

      enum: ['Watching', 'Accepted', 'Canceled'],

      default: 'Watching'

    }

});

const Announcement = mongoose.model('Announcement', announcementSchema);

module.exports = Announcement;

# ПРИЛОЖЕНИЕ 4.Файл userModel.js

const mongoose = require('mongoose');

const userSchema = new mongoose.Schema({

    lastName: String,

    firstName: String,

    middleName: String,

    phone: { type: String, unique: true },

    birthday: Date,

    password: String,

    role: { type: String, default: 'user' }

});

const User = mongoose.model('User', userSchema);

module.exports = User;

# ПРИЛОЖЕНИЕ 5. Файлы main.html, main.js, main.css

// main.html

<!doctype html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Бюро находок</title>

    <link rel="stylesheet" type="text/css" href="main.css">

</head>

<body>

    <header>

        <h1>Бюро находок</h1>

    </header>

    <div class="container">

      <aside>

        <nav id="userNav">

          <a id="profileLink" href="../profDate/profDate.html?userId=" style="display:none;">Профиль</a>

            <a id="AddAdvLink" href="../Advert/createAd.html">Создать объявление</a>

            <a id="myAdsLink" href="../myads/myads.html" style="display:none;">Мои объявления</a>

            <a id="logoutLink" href="#" style="display:none;">Выход</a>

            <a id="loginRegisterLink" href="../LoginOrRegistration/logreg.html">Вход и Регистрация</a>

            <!-- Элементы для администратора -->

            <div id="adminControls" style="display:none;">

                <a href="../admin/manageAccounts/manageAccounts.html">Управление учетными записями</a>

            </div>

            <!-- Элементы для модератора -->

            <div id="moderatorControls" style="display:none;">

                <a href="../moderator/moderationAds/moderationAds.html">Модерация объявлений</a>

            </div>

        </nav>

    </aside>

        <main>

            <div class="controls">

                <button id="last7DaysButton">За последние 7 дней</button>

                <button id="lastMonthButton">За последний месяц</button>

                <button id="resetButton">Сброс</button>

                <hr>

                <button id="allAdsButton">Все объявления</button>

                <button id="ownerNotFoundButton">Владелец не найден</button>

                <button id="ownerFoundButton">Нашелся владелец</button>

                <hr>

            </div>

            <div class="container" id="adContainer">

                <h4>Объявления</h4>

            </div>

        </main>

    </div>

    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>

    <script type="text/javascript" src="main.js"></script>

</body>

</html>

// main.js

$(document).ready(function() {

  // const urlParams = new URLSearchParams(window.location.search);

  const userId = localStorage.getItem('userId');

  if (userId) {

    $('#profileLink').attr('href', `../profDate/profDate.html?userId=${userId}`).show();

      // console.log('Пользователь с ID ' + userId + ' вошел в систему.');

      // Получаем информацию о пользователе с сервера

      fetch(`/profile/${userId}`)

          .then(response => {

              if (!response.ok) {

                  throw new Error('Ошибка при загрузке профиля пользователя');

              }

              return response.json();

          })

          .then(user => {

              // console.log('Получен профиль пользователя:', user);

              // Проверяем роль пользователя

              if (user.role === 'admin' || user.role === 'superadmin') {

                  $('#profileLink').show();

                  $('#myAdsLink').show();

                  $('#logoutLink').show();

                  $('#loginRegisterLink').hide();

                  // Показываем элементы для администратора

                  $('#adminControls').show();

              } else if (user.role === 'moderator') {

                  $('#profileLink').show();

                  $('#myAdsLink').show();

                  $('#logoutLink').show();

                  $('#loginRegisterLink').hide();

                  // Показываем элементы для модератора

                  $('#moderatorControls').show();

              } else {

                  $('#profileLink').show();

                  $('#myAdsLink').show();

                  $('#logoutLink').show();

                  $('#loginRegisterLink').hide();

              }

          })

          .catch(error => {

              console.error('Ошибка:', error);

              $('#loginRegisterLink').show();

          });

  } else {

      // console.log('Пользователь не вошел в систему.');

  }

  function getAdHTML(ad) {

      const adContainer = document.createElement('div');

      adContainer.classList.add('ad');

      adContainer.setAttribute('data-id', ad.\_id);

      const image = document.createElement('img');

    image.src = ad.photoUrl ? `../uploads/${ad.photoUrl}` : '../uploads/no-image-thumb.jpg'; // Загрузка изображения по пути из объявления

    image.alt = ad.title;

    adContainer.appendChild(image);

      image.alt = ad.title;

      adContainer.appendChild(image);

      const infoContainer = document.createElement('div');

      infoContainer.classList.add('ad-info');

      const title = document.createElement('h2');

      title.textContent = ad.title;

      infoContainer.appendChild(title);

      const date = document.createElement('p');

      date.textContent = `Дата: ${new Date(ad.date).toLocaleDateString()}`;

      infoContainer.appendChild(date);

      const location = document.createElement('p');

      location.textContent = `Местоположение: ${ad.location}`;

      infoContainer.appendChild(location);

      const status = document.createElement('p');

      status.textContent = `Владелец: ${ad.status ? 'Найден' : 'Не найден'}`;

      infoContainer.appendChild(status);

      adContainer.appendChild(infoContainer);

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

          const adId = adContainer.getAttribute('data-id');

          window.location.href = `../AdDetail/adDetail.html?adId=${adId}`;

      });

      return adContainer;

  }

  function loadAds(sortByDate = false, filterStatus = null, daysRange = 365) {

      const container = document.getElementById('adContainer');

      container.innerHTML = '';

      // console.log("Попытка загрузить объявления");

      fetch(`/ads?sortByDate=${sortByDate}&filterStatus=${filterStatus}&daysRange=${daysRange}`)

          .then(response => {

              if (!response.ok) {

                  throw new Error('Ошибка при загрузке данных');

              }

              return response.json();

          })

          .then(data => {

              // console.log('Fetched ads:', data);

              container.innerHTML = '';

              data.forEach(ad => {

                  container.appendChild(getAdHTML(ad));

              });

          })

          .catch(error => {

              console.error('Ошибка:', error);

              container.innerHTML = '';

              const textMessage = document.createElement('p');

              textMessage.textContent = 'Ошибка при загрузке данных. Пожалуйста, попробуйте еще раз позже.';

              container.appendChild(textMessage);

          });

  }

  $('#resetButton').click(function() {

      loadAds();

  });

  $('#allAdsButton').click(function() {

      loadAds(false, null);

  });

  $('#ownerNotFoundButton').click(function() {

      loadAds(false, false);

  });

  $('#ownerFoundButton').click(function() {

      loadAds(false, true);

  });

  $('#last7DaysButton').click(function() {

      loadAds(false, null, 7);

  });

  $('#lastMonthButton').click(function() {

      loadAds(false, null, 31);

  });

  $('#logoutLink').click(function(event) {

      event.preventDefault();

      localStorage.removeItem('userId');

      window.location.href = '../LoginOrRegistration/logreg.html';

  });

  loadAds();

});

// main.css

body {

  font-family: Arial, sans-serif;

  background-color: #f4f4f4;

  margin: 0;

  padding: 0;

}

header {

  background-color: #6d8eb0;

  color: white;

  text-align: center;

  padding: 20px 0;

}

.container {

  display: flex;

  margin: 20px;

}

aside {

  width: 200px;

  background-color: #3a699b;

  margin-right: 20px;

}

aside nav a {

  display: block;

  color: rgb(187, 198, 223);

  text-decoration: none;

  margin: 10px 0;

}

main {

  flex: 1;

  background-color: #ffffff;

  padding: 20px;

  border-radius: 8px;

  box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

}

.controls {

  margin-top: 10px;

  text-align: right;

}

.controls button {

  margin-left: 10px;

}

button {

  padding: 10px 20px;

  background-color: #3a699b;

  color: white;

  border: none;

  border-radius: 4px;

  cursor: pointer;

}

button:hover {

  background-color: #0056b3;

}

#adContainer {

  display: flex;

  flex-direction: column;

  align-items: center;

}

#adContainer .ad {

  background-color: #ffffff;

  border: 1px solid #ddd;

  border-radius: 8px;

  padding: 15px;

  margin-bottom: 20px;

  width: 80%;

  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);

  display: flex;

  align-items: flex-start;

}

#adContainer .ad img {

  max-width: 150px;

  height: auto;

  border-radius: 4px;

  margin-right: 15px;

}

#adContainer .ad h2 {

  margin-top: 0;

}

#adContainer .ad p {

  margin: 5px 0;

}

.ad-info {

  flex: 1;

}

#adImage {

  max-width: 100%; /\* Убедитесь, что изображение не выходит за пределы контейнера \*/

  height: auto; /\* Сохраняет пропорции изображения \*/

  border-radius: 8px; /\* Закругленные углы \*/

  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1); /\* Добавляет тень \*/

  margin-bottom: 15px; /\* Добавляет отступ снизу \*/

  display: block; /\* Устанавливает блочный элемент \*/

}

# ПРИЛОЖЕНИЕ 6. Файл announcementController.js

const Announcement = require('../models/announcementModel');

const User = require('../models/userModel');

const createAd = async (req, res) => {

  const { title, description, date, location, userId, photoUrl } = req.body;

  try {

      const announcement = new Announcement({

          title,

          description,

          date,

          location,

          userId,

          photoUrl

      });

      await announcement.save();

      res.status(201).send('Объявление успешно добавлено!');

  } catch (error) {

      console.error('Ошибка при добавлении объявления:', error);

      res.status(500).send('Ошибка при добавлении объявления');

  }

};

const getAdDetail = async (req, res) => {

  const adId = req.query.id;

  try {

      const ad = await Announcement.findById(adId);

      if (!ad) {

          return res.status(404).send('Объявление не найдено');

      }

      const user = await User.findById(ad.userId);

      if (!user) {

          return res.status(404).send('Пользователь не найден');

      }

      const adDetail = {

          ...ad.toObject(),

          name: `${user.firstName} ${user.lastName}`,

          phone: user.phone

      };

      res.json(adDetail);

  } catch (error) {

      console.error(error);

      res.status(500).send('Ошибка при получении объявления');

  }

};

const updateAd = async (req, res) => {

  const { id, title, description, date, location, status, moderationStatus, photoUrl } = req.body;

  try {

      const updatedAd = await Announcement.findByIdAndUpdate(id, {

          title,

          description,

          date,

          location,

          status,

          moderationStatus,

          photoUrl

      }, { new: true });

      res.json(updatedAd);

  } catch (error) {

      console.error('Ошибка при обновлении объявления:', error);

      res.status(500).send('Ошибка при обновлении объявления');

  }

};

const deleteAd = async (req, res) => {

  const { id } = req.body;

  try {

      const ad = await Announcement.findById(id);

      if (ad) {

          await Announcement.findByIdAndDelete(id);

          res.status(200).send({ message: 'Объявление успешно удалено' });

      } else {

          res.status(404).send({ message: 'Объявление не найдено' });

      }

  } catch (error) {

      console.error('Ошибка при удалении объявления:', error);

      res.status(500).send('Ошибка при удалении объявления');

  }

};

const getUserAds = async (req, res) => {

  const { userId, sortByDate, filterStatus, daysRange } = req.query;

  const dateLimit = new Date();

  dateLimit.setDate(dateLimit.getDate() - (daysRange || 365));

  let filter = { date: { $gte: dateLimit }, userId: userId };

  if (filterStatus !== undefined) {

      if (filterStatus === 'true' || filterStatus === 'false') {

          filter.status = filterStatus === 'true';

      }

  }

  try {

      const ads = await Announcement.find(filter).sort(sortByDate ? { date: -1 } : {});

      const groupedAds = ads.reduce((acc, ad) => {

          const status = ad.moderationStatus;

          if (!acc[status]) {

              acc[status] = [];

          }

          acc[status].push(ad);

          return acc;

      }, {});

      res.json(groupedAds);

  } catch (error) {

      console.error('Ошибка при получении объявлений:', error);

      res.status(500).send('Ошибка при получении объявлений');

  }

};

const getAds = async (req, res) => {

  const { sortByDate, filterStatus, daysRange } = req.query;

  const dateLimit = new Date();

  dateLimit.setDate(dateLimit.getDate() - (daysRange || 365));

  let filter = {

      date: { $gte: dateLimit },

      moderationStatus: 'Accepted'

  };

  if (filterStatus !== undefined) {

      if (filterStatus === 'true' || filterStatus === 'false') {

          filter.status = filterStatus === 'true';

      }

  }

  try {

      const ads = await Announcement.find(filter).sort(sortByDate ? { date: -1 } : {});

      res.json(ads);

  } catch (error) {

      console.error('Ошибка при получении объявлений:', error);

      res.status(500).send('Ошибка при получении объявлений');

  }

};

const updateAdStatus = async (req, res) => {

  const { id, moderationStatus } = req.body;

  if (!id || !moderationStatus) {

      return res.status(400).send('Invalid request');

  }

  try {

      const ad = await Announcement.findByIdAndUpdate(id, { moderationStatus }, { new: true });

      if (!ad) {

          return res.status(404).send('Announcement not found');

      }

      res.json(ad);

  } catch (error) {

      console.error('Ошибка при обновлении статуса объявления:', error);

      res.status(500).send('Ошибка при обновлении статуса объявления');

  }

};

const getAdsModeration = async (req, res) => {

  const { moderationStatus } = req.query;

  const filter = {};

  if (moderationStatus) {

      filter.moderationStatus = moderationStatus;

  }

  try {

      const ads = await Announcement.find(filter);

      res.json(ads);

  } catch (error) {

      console.error('Ошибка при получении объявлений:', error);

      res.status(500).send('Ошибка при получении объявлений');

  }

};

module.exports = {

  createAd,

  getAdDetail,

  updateAd,

  deleteAd,

  getUserAds,

  getAds,

  updateAdStatus,

  getAdsModeration

};

# ПРИЛОЖЕНИЕ 7. Файл userController.js

const bcrypt = require('bcryptjs');

const jwt = require('jsonwebtoken');

const User = require('../models/userModel');

const JWT\_SECRET = 'your\_jwt\_secret';

const register = async (req, res) => {

  const { lastName, firstName, middleName, phone, birthday, password } = req.body;

  if (!lastName || !firstName || !phone || !birthday || !password) {

      return res.status(400).send('All fields are required');

  }

  try {

      const hashedPassword = await bcrypt.hash(password, 10);

      const user = new User({

          lastName,

          firstName,

          middleName,

          phone,

          birthday,

          password: hashedPassword,

          role: 'user'

      });

      await user.save();

      res.status(201).send('User registered successfully');

  } catch (error) {

      res.status(500).send('Error registering user');

  }

};

const login = async (req, res) => {

  const { phone, password } = req.body;

  if (!phone || !password) {

      return res.status(400).send('All fields are required');

  }

  try {

      const user = await User.findOne({ phone });

      if (!user) {

          return res.status(400).send('Invalid credentials');

      }

      const isPasswordValid = await bcrypt.compare(password, user.password);

      if (!isPasswordValid) {

          return res.status(400).send('Invalid credentials');

      }

      const token = jwt.sign({ userId: user.\_id }, JWT\_SECRET, { expiresIn: '1h' });

      res.json({ userId: user.\_id, token });

  } catch (error) {

      res.status(500).send('Error logging in');

  }

};

const getProfile = async (req, res) => {

  const userId = req.params.userId;

// console.log(userId);

  try {

      const user = await User.findById(userId);

      if (!user) {

          return res.status(404).send('User not found');

      }

      res.json(user);

  } catch (error) {

      res.status(500).send('Error loading user profile');

  }

};

const updateProfile = async (req, res) => {

  const userId = req.params.userId;

  const { lastName, firstName, middleName, phone, password } = req.body;

  let updatedProfile = { lastName, firstName, middleName, phone };

  if (password) {

      const hashedPassword = await bcrypt.hash(password, 10);

      updatedProfile.password = hashedPassword;

  }

  try {

      await User.findByIdAndUpdate(userId, updatedProfile);

      res.status(200).send('Profile updated successfully');

  } catch (error) {

      res.status(500).send('Error updating user profile');

  }

};

const deleteProfile = async (req, res) => {

  const userId = req.params.userId;

  try {

      await User.findByIdAndDelete(userId);

      res.status(200).send('User deleted successfully');

  } catch (error) {

      res.status(500).send('Error deleting user');

  }

};

const userNewCreate =  async (req, res) => {

  const { lastName, firstName, middleName, phone, role, password } = req.body;

  try {

    // console.log('Received data:', req.body);

      const hashedPassword = await bcrypt.hash(password, 10);

      const newUser = new User({

          lastName,

          firstName,

          middleName,

          phone,

          role,

          password: hashedPassword

      });

      await newUser.save();

      res.status(201).send('User created successfully');

  } catch (error) {

      res.status(500).send('Error creating user');

  }

};

// Получение списка пользователей

const usersList =  async (req, res) => {

  try {

      const users = await User.find();

      res.json(users);

  } catch (error) {

      res.status(500).send('Ошибка при получении списка пользователей');

  }

};

// Получение списка пользователей по роли

const sortedUserByRole =  async (req, res) => {

  const { role } = req.query;

  try {

      const users = await User.find({ role });

      res.json(users);

  } catch (error) {

      res.status(500).send('Ошибка при получении списка пользователей');

  }

};

module.exports = {

  register,

  login,

  getProfile,

  updateProfile,

  deleteProfile,

  userNewCreate,

  usersList,

  sortedUserByRole

};