Национальный исследовательский технологический университет «МИСиС» Институт информационных технологий и компьютерных наук (ИТКН)

**Курс «Разработка клиент-серверных приложений»**

Курсовая работа

**«Разработка клиент-серверного приложения Ask me»**

Выполнила: студентка группы БИВТ-21-8

Робиш А.С.

Проверил: Рзазаде У.А.

Москва, 2023

**Оглавление**

[Цели работы 3](#_Toc136373792)

[Задачи 3](#_Toc136373793)

[Введение 3](#_Toc136373794)

[1. Предметная область 3](#_Toc136373795)

[2. Постановка задачи 4](#_Toc136373796)

[3. Описание архитектуры 5](#_Toc136373797)

[4. Описание структуры БД 6](#_Toc136373798)

[5. Описание серверной части 8](#_Toc136373799)

[6. Описание клиентской части 10](#_Toc136373800)

[7. Заключение 17](#_Toc136373801)

[8. Список литературы 18](#_Toc136373802)

Цели работы

Изучение основных понятий веб-разработки, получение навыков разработки клиент-серверных приложений и работы с базами данных. Изучение возможностей оптимизации работы приложения как со стороны клиента, так и со стороны сервера, путем использования дополнительных библиотек.

Задачи

В ходе планирования работы были выделены следующие задачи:

* Изучение концепции клиент-серверной модели;
* Разработка функциональных требований к приложению;
* Проектирование архитектуры приложения;
* Разработка клиентской и серверной частей приложения;
* Ручное тестирование приложения;
* Оформление отчета по выполненной работе [1].

Введение

Клиент-серверное приложение – это программное обеспечение, состоящее из двух основных компонентов: клиента и сервера, которые взаимодействуют между собой через сеть. Клиентское приложение предназначено для использования конечным пользователем и позволяет ему отправлять запросы на сервер для получения данных или выполнения определенных операций. Серверное приложение отвечает за обработку этих запросов и предоставляет клиентам необходимые ресурсы, такие как файлы, базы данных или другие данные.

Клиент-серверные приложения широко используются в различных сферах, таких как банковское дело, медицина, образование, транспорт и т.д. Они позволяют эффективно организовывать работу с данными и ресурсами в распределенной среде, где несколько пользователей могут одновременно работать с одними и теми же данными. Это делает область разработки клиент-серверных приложений актуальной как для специалистов, так и для пользователей.

1. Предметная область

Одним из наиболее популярных видов клиент-серверных приложений на данный момент являются сайты-социальные сети. Они используются как для личного общения, так и для бизнес-целей, таких как продвижение товаров и услуг.

При разработке социальных сетей необходимо обеспечить высокую производительность приложения, чтобы оно работало стабильно и могло обрабатывать большое количество запросов. Также нужно обеспечить безопасность пользователей и возможность авторизации для сохранения созданных пользователем данных, поскольку утечка данных и паролей является частой проблемой в области социальных сетей. Важным аспектом является удобный и интуитивно понятный интерфейс, что обеспечит комфортную навигацию по приложению и позволит быстро находить нужную информацию.

1. Постановка задачи

Целью работы является разработка клиент-серверного приложения, имеющего функциональность социальной сети. Социальная сеть, разработанная в данной работе, представляет собой платформу для ответов на вопросы от других пользователей. Чтобы решить проблему безопасности пользователей, указанную в предметной области, платформа должна поддерживать функционал авторизации с шифрованием паролей.

Приложение должно иметь следующий функционал:

* Регистрация пользователей с возможностью авторизации;
* Возможность задавать вопросы другим пользователям и получать ответы на свои вопросы;
* Наличие системы анонимных вопросов и ответов, когда пользователь не авторизован;
* Возможность просмотра профилей пользователей, включая список заданных им вопросов и ответов на них;
* Возможность редактирования, скрытия и удаления заданных вопросов;
* Наличие системы хэштегов, позволяющей пользователям добавлять хэштеги в свои вопросы и ответы к вопросам, а также отображать все вопросы с заданным хэштегом на одной странице.

Важно отметить, что редактировать, скрывать и удалять вопросы пользователь должен иметь возможность только в собственном профиле, к действиям над вопросами других у него не должно быть доступа.

1. Описание архитектуры

Архитектура разработанного приложения основывается на клиент-серверной модели. На серверной стороне приложение работает на базе языка программирования Ruby (2.7.2) и фреймворка Ruby on Rails (5.2.8) [2]. В качестве базы данных для хранения информации о пользователях, вопросах и ответах используется SQLite3.

Клиентская часть приложения использует HTML и CSS для оформления интерфейса пользовательской части. Для оптимизации приложения используется библиотека Turbolinks, позволяющая ускорить загрузку при переходе по ссылкам внутри приложения, и Puma, поддерживающая многопоточную обработку HTTP-запросов.

Приложение поддерживает кэширование данных, что позволяет уменьшить нагрузку на сервера и ускорить работу приложения. Инструментом кэширования, используемым в данной работе, является библиотека Bootsnap.

Безопасность приложения обеспечивается через механизмы проверки подлинности, включая аутентификацию и авторизацию пользователей.

Помимо вышеперечисленных, при разработке (в окружении development) использовались следующие библиотеки (гемы [3]):

* Bcrypt – библиотека для шифрования паролей;
* Coffee-rails – стандартная библиотека, включаемая в проект по умолчанию при его создании. Предоставляет поддержку CoffeeScript в приложении;
* Font-awesome-rails – предоставляет поддержку обширной библиотеки иконок и шрифтов для веб-приложений;
* Gravtastic – позволяет пользователям добавлять аватар в свой профиль через сторонний сервис Gravatar;
* Jbuilder – предоставляет возможность создавать сложные JSON-структуры, используя Ruby-код. В версии Ruby on Rails 2.7.2 включен в Gemfile по умолчанию;
* Rails – фреймворк для веб-разработки на языке Ruby;
* Sass-rails – интеграция проектов Ruby on Rails с шаблонизатором Sass;
* Hirb – предоставляет инструменты для удобного вывода и форматирования данных в консоли;
* Byebug – предоставляет инструменты для отладки кода в приложении;
* Sqlite3 – отвечает за интеграцию приложения с базой данных SQLite3;
* Web-console – предоставляет интерактивную консоль приложения, доступную из браузера;
* Listen – следит за изменениями файлов и автоматически обновляет сервер при сохранении новых изменений;
* Spring – запускает приложение в фоновом режиме, что позволяет запускать миграции и тесты без необходимости останавливать сервер;
* Spring-watcher-listen – необходим для связи библиотек Listen и Spring.

Основные сущности данного приложения – Пользователь (User), Вопрос (Question), Хэштег (Hashtag).

Сущность Пользователя отвечает за логику, связанную с аккаунтами, их создание, редактирование и удаление. В ней содержатся валидации на e-mail пользователя, а также на длину и формат никнейма. Сущность Вопрос отвечает за создание, редактирование, удаление и скрытие вопросов. Она содержит валидацию на присутствие текста вопроса и его длину. Сущность Хэштег отвечает за хэштеги, содержащиеся в тексте вопросов и ответов. По ним можно находить вопросы с такими же хэштегами, таким образом можно группировать вопросы по определенным темам.

1. Описание структуры БД

Структуру базы данных, используемой в данном приложении, можно представить в виде схемы, приведенной на рис. 1.

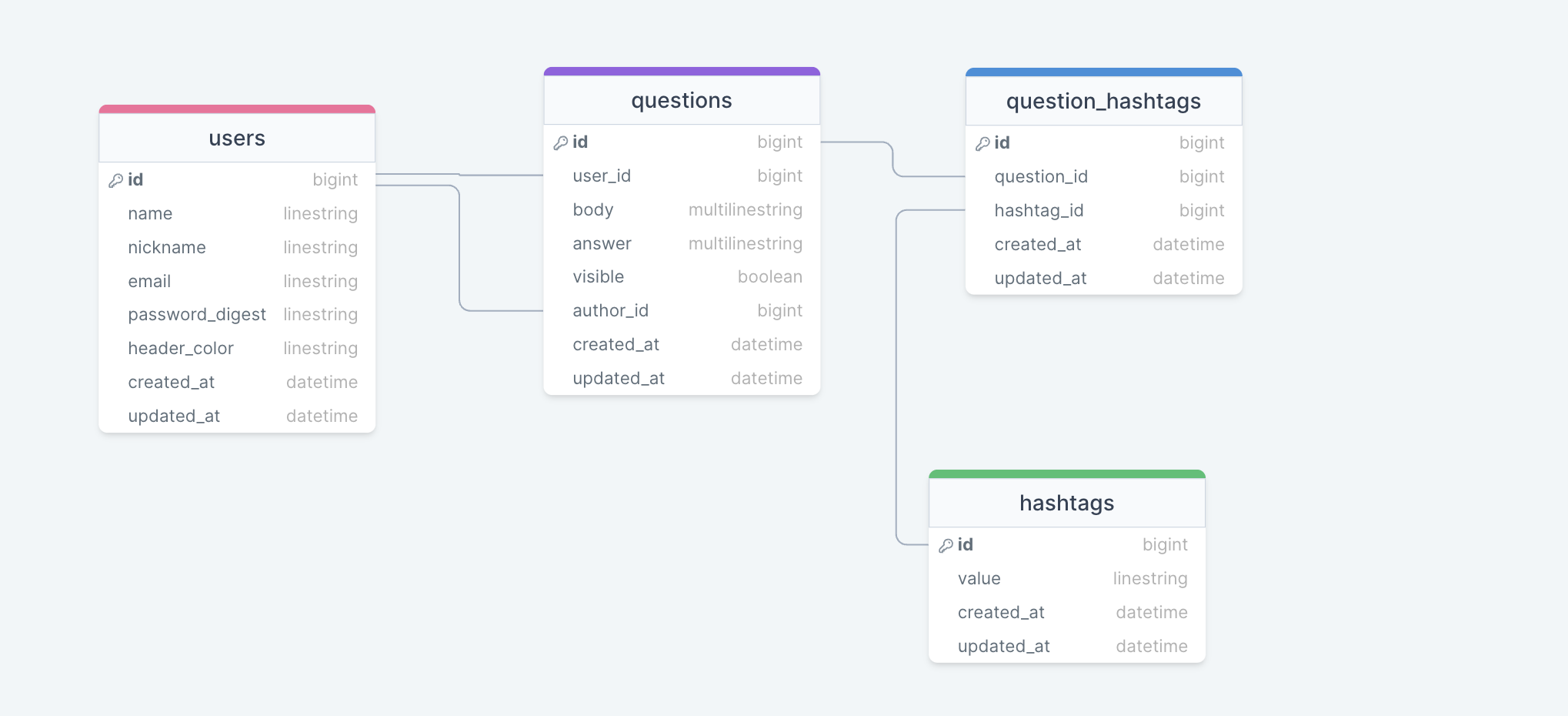


Рисунок 1 – Структура базы данных

Главными сущностями являются: Пользователь (User), Вопрос (Question), Хэштег (Hashtag). Между таблицами users и questions реализована связь one-to-many через поля user\_id и author\_id. С точки зрения логики приложения это означает, что пользователь может задавать более одного вопроса и получать более одного вопроса. Между таблицами questions и hashtags реализована связь many-to-many через дополнительную таблицу question\_hashtags. С точки зрения логики приложения это означает, что в одном вопросе может содержаться несколько хэштегов, и один и тот же хэштег может присутствовать в нескольких вопросах.

Также между пользователями и вопросами прописана зависимость dependent: destroy, которая отмечена в схеме как force: :cascade. Это позволит при удалении пользователя удалить также все вопросы, созданные им. Аналогичная зависимость прописана для сущностей Question – QuestionHashtag и Hashtag – QuestionHashtag. Это позволяет удалить связь вопроса или хэштега с удаленной связанной сущностью, и не отображать ее в общем списке.

Полный код схемы приложения приведен ниже на Листинге 1.

ActiveRecord::Schema.define(version: 2022\_10\_24\_160337) do  
  
 create\_table "hashtags", force: :cascade do |*t*|  
 *t*.string "value"  
 *t*.datetime "created\_at", null: false  
 *t*.datetime "updated\_at", null: false  
 end  
  
 create\_table "question\_hashtags", force: :cascade do |*t*|  
 *t*.integer "question\_id"  
 *t*.integer "hashtag\_id"  
 *t*.datetime "created\_at", null: false  
 *t*.datetime "updated\_at", null: false  
 *t*.index ["hashtag\_id"], name: "index\_question\_hashtags\_on\_hashtag\_id"  
 *t*.index ["question\_id"], name: "index\_question\_hashtags\_on\_question\_id"  
 end  
  
 create\_table "questions", force: :cascade do |*t*|  
 *t*.text "body"  
 *t*.integer "user\_id"  
 *t*.datetime "created\_at", null: false  
 *t*.datetime "updated\_at", null: false  
 *t*.boolean "visible", default: true  
 *t*.text "answer"  
 *t*.integer "author\_id"  
 *t*.index ["user\_id"], name: "index\_questions\_on\_user\_id"  
 end  
  
 create\_table "users", force: :cascade do |*t*|  
 *t*.string "name"  
 *t*.string "nickname"  
 *t*.string "email"  
 *t*.datetime "created\_at", null: false  
 *t*.datetime "updated\_at", null: false  
 *t*.string "password\_digest"  
 *t*.string "header\_color", default: "#370617"  
 *t*.index ["email"], name: "index\_users\_on\_email", unique: true  
 *t*.index ["name"], name: "index\_users\_on\_name", unique: true  
 end  
end

Листинг 1 – Схема базы данных

1. Описание серверной части

При разработке приложения использовался паттерн MVC. Связь между моделями, контроллерами и представлениями показана на Схеме 1.

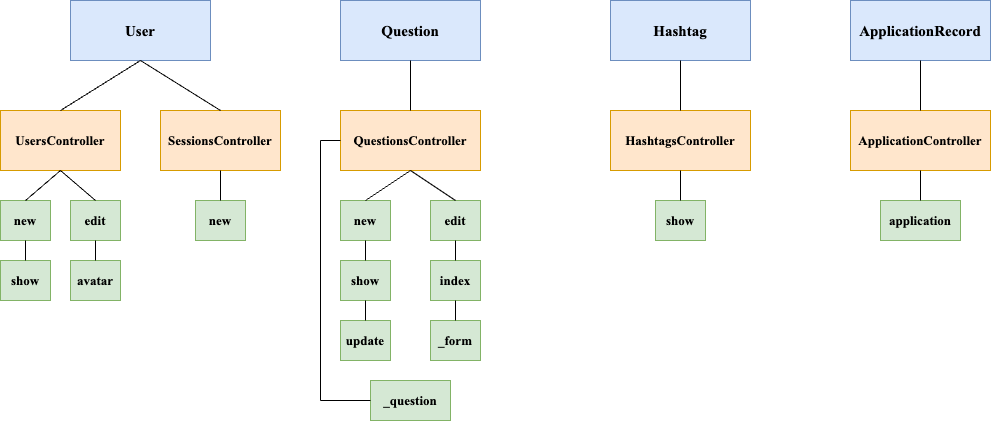


Схема 1 – Связь между моделями, контроллерами и представлениями

На приведенной схеме модели отмечены синим цветом, контроллеры – оранжевым цветом, представления – зелёным цветом. Каждое представление соответствует одноименному действию (action) контроллера. Представления вопросов \_form и \_question являются частичными представлениями (partials) и встраиваются в другие представления во избежание дублирования кода.

Модель ApplicationRecord в Ruby on Rails является базовой моделью для всех остальных моделей в приложении. Она может содержать основные или вспомогательные функции и не записывается в базу данных, по этой причине модель ApplicationRecord не была включена в разделы 3 и 4.

Контроллер ApplicationController является базовым контроллером, содержащим основные или вспомогательные методы для остальных контроллеров в приложении. В данном приложении он содержит методы, позволяющие определить текущего пользователя и задать базовый текст для перенаправления из контроллера с ошибкой.

Представление application.html.erb содержит в себе главную структуру страницы – название, подключенные теги, шапку сайта с кнопками для авторизации или регистрации.

Контроллер QuestionsController содержит в себе восемь публичных методов и два приватных. Из публичных методов есть следующие:

* Create – создает вопрос из параметров, заданных пользователем;
* Update – позволяет обновить вопрос после редактирования;
* Hide – скрывает вопрос;
* Destroy – удаляет вопрос;
* Show – страница вопроса с заданным id;
* Index – главная страница со всеми вопросами. Совпадает с главной страницей сайта;
* New – форма нового вопроса;
* Edit – позволяет редактировать вопрос.

Из приватных:

* Ensure\_current\_user – вызывается перед методами update, destroy, edit, hide и проверяет наличие права пользователя на изменение вопроса;
* Set\_question\_for\_current\_user – вызывается перед методами update, destroy, edit, hide и находит в базе данных вопрос для текущего пользователя с id из параметров.

Контроллер UsersController содержит методы CRUD: create, new, edit, update, destroy, show.

Контроллер HashtagsController содержит в себе один метод – show. Он отвечает за отображение страницы хэштега, на которой пользователь может увидеть все вопросы с данным хэштегом.

Контроллер SessionsController содержит три публичных метода: new, create, destroy, отвечающие за вход в аккаунт и выход из него.

1. Описание клиентской части

Клиентская часть приложения представляет собой веб-интерфейс, доступный через браузер на компьютере или мобильном устройстве. Главный экран состоит из навигационной панели со ссылками на главную страницу, регистрацию или вход для неавторизованных пользователей и ссылкой для выхода из аккаунта для авторизованных.

На главной странице содержится список недавно зарегистрированных пользователей и последних хэштегов, а также последние 10 вопросов (см. рис. 2). Если в базе данных содержится менее 10 вопросов, то выводится это количество с нужным склонением («Последний 1 вопрос», «Последние 2 вопроса», «Последние 5 вопросов»).

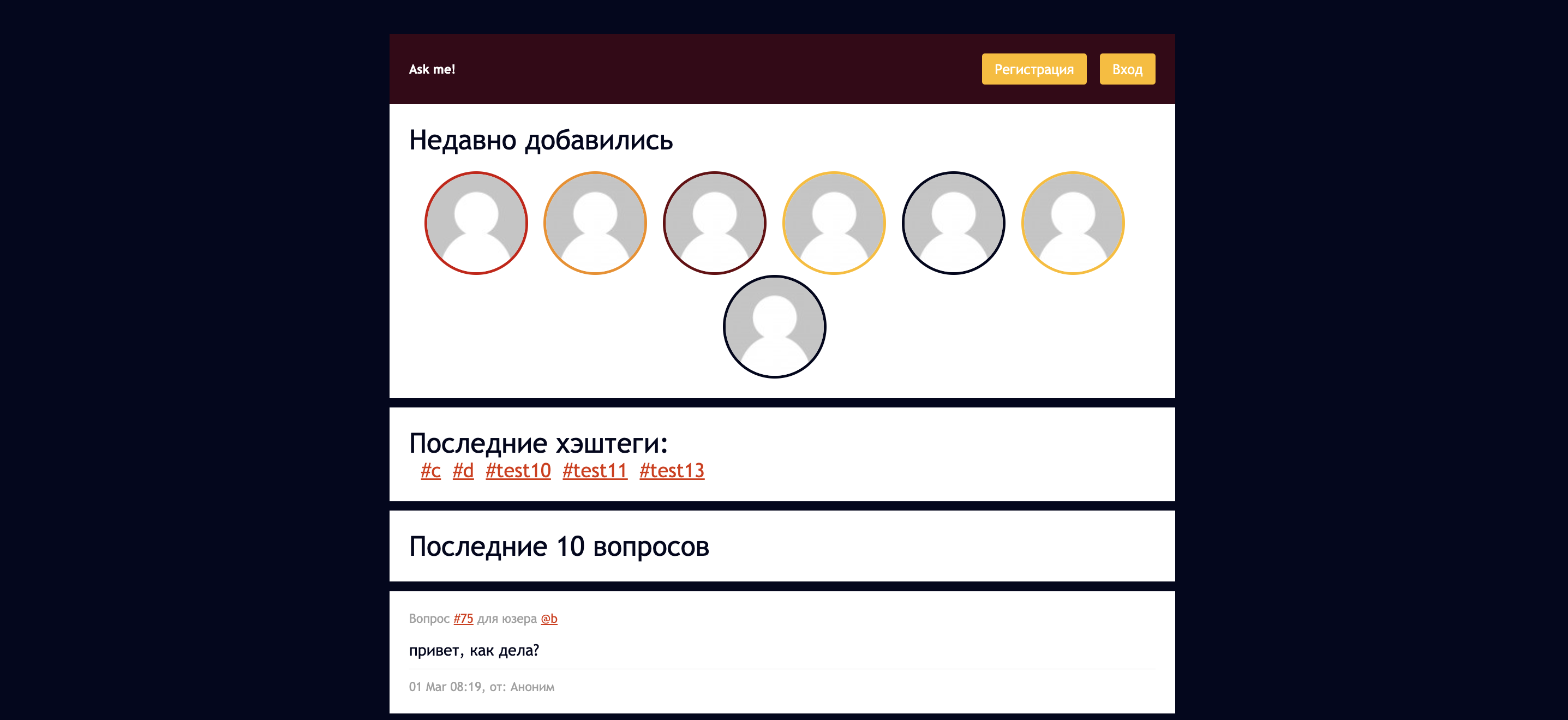


Рисунок 2 – Главная страница приложения

Пользователь может создать новый аккаунт, нажав на кнопку «Регистрация». Страница регистрации приведена на рис. 3. После нажатия кнопки «Зарегистрироваться» пароль пользователя шифруется с помощью гема Bcrypt, и новый пользователь сохраняется в базу данных.

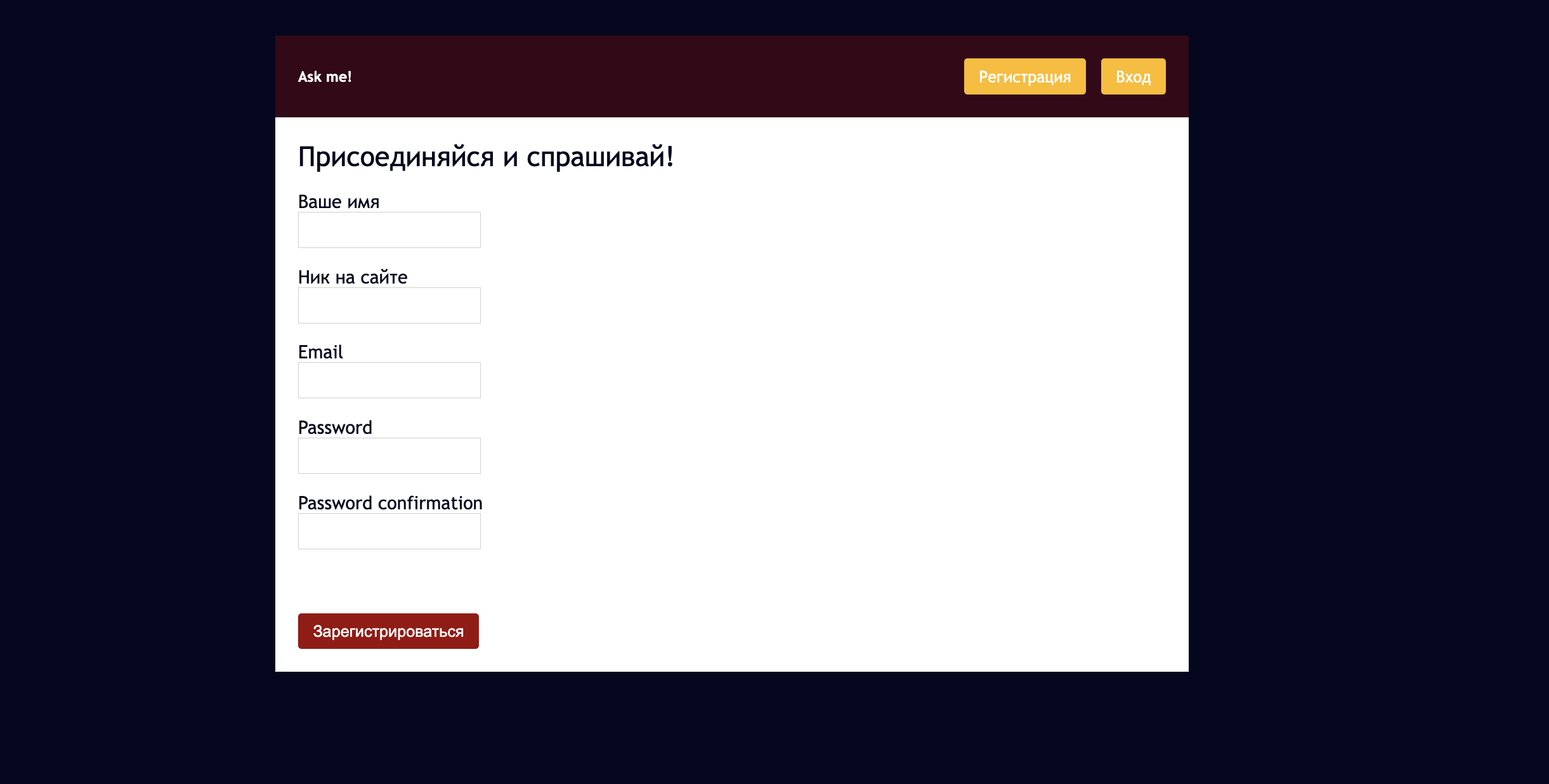


Рисунок 3 – Страница регистрации

При попытке сохранить пользователя с пустыми полями форма высветит ошибку о неправильном заполнении полей. В таком случае пользователь не сохраняется.

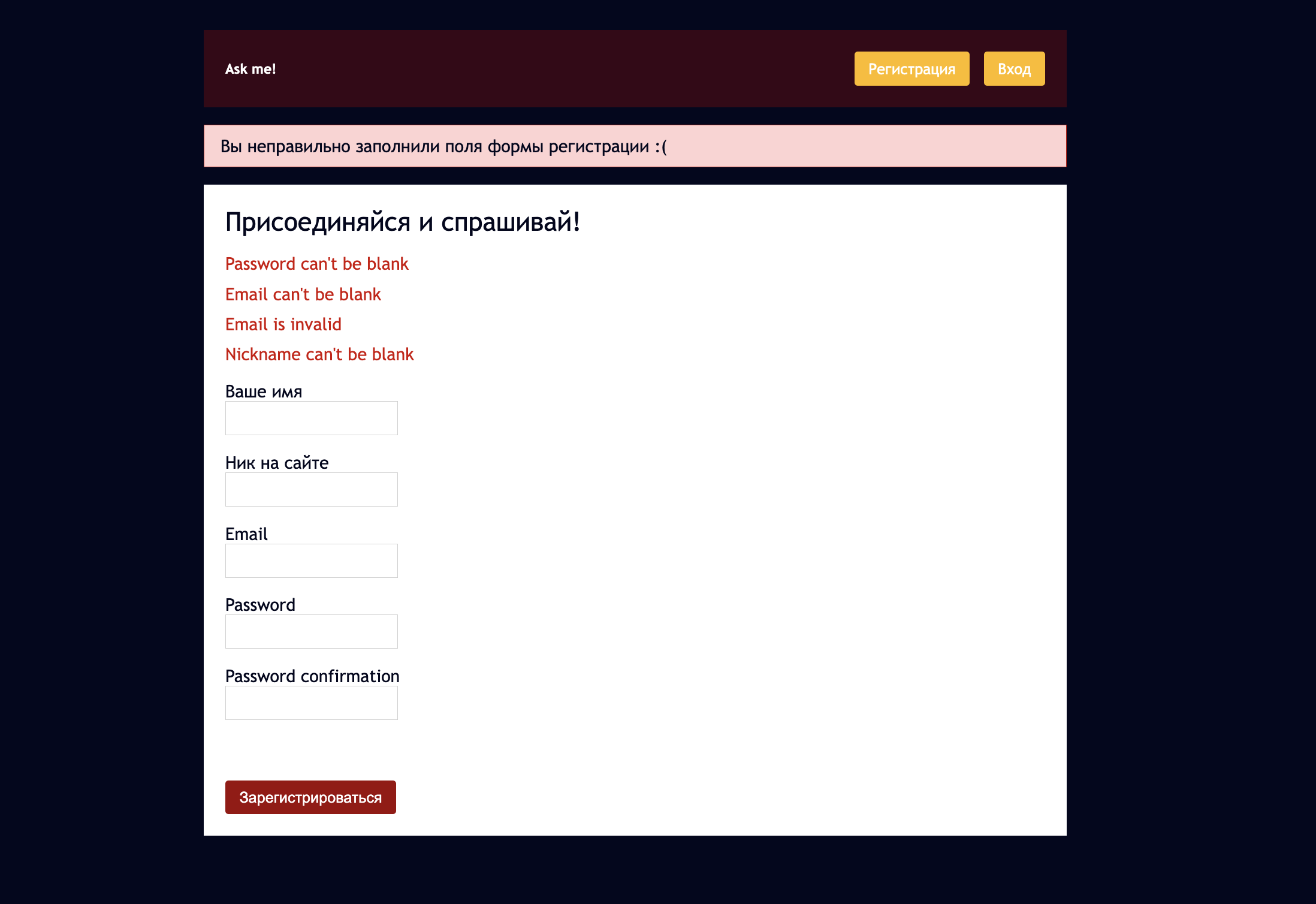


Рисунок 4 – Ошибки формы регистрации

На странице профиля каждого пользователя отображается его аватар, имя и никнейм, а также список заданных ему вопросов и ответов на эти вопросы. Также в профиле видно количество вопросов, заданных пользователю, как показано на рис. 5.

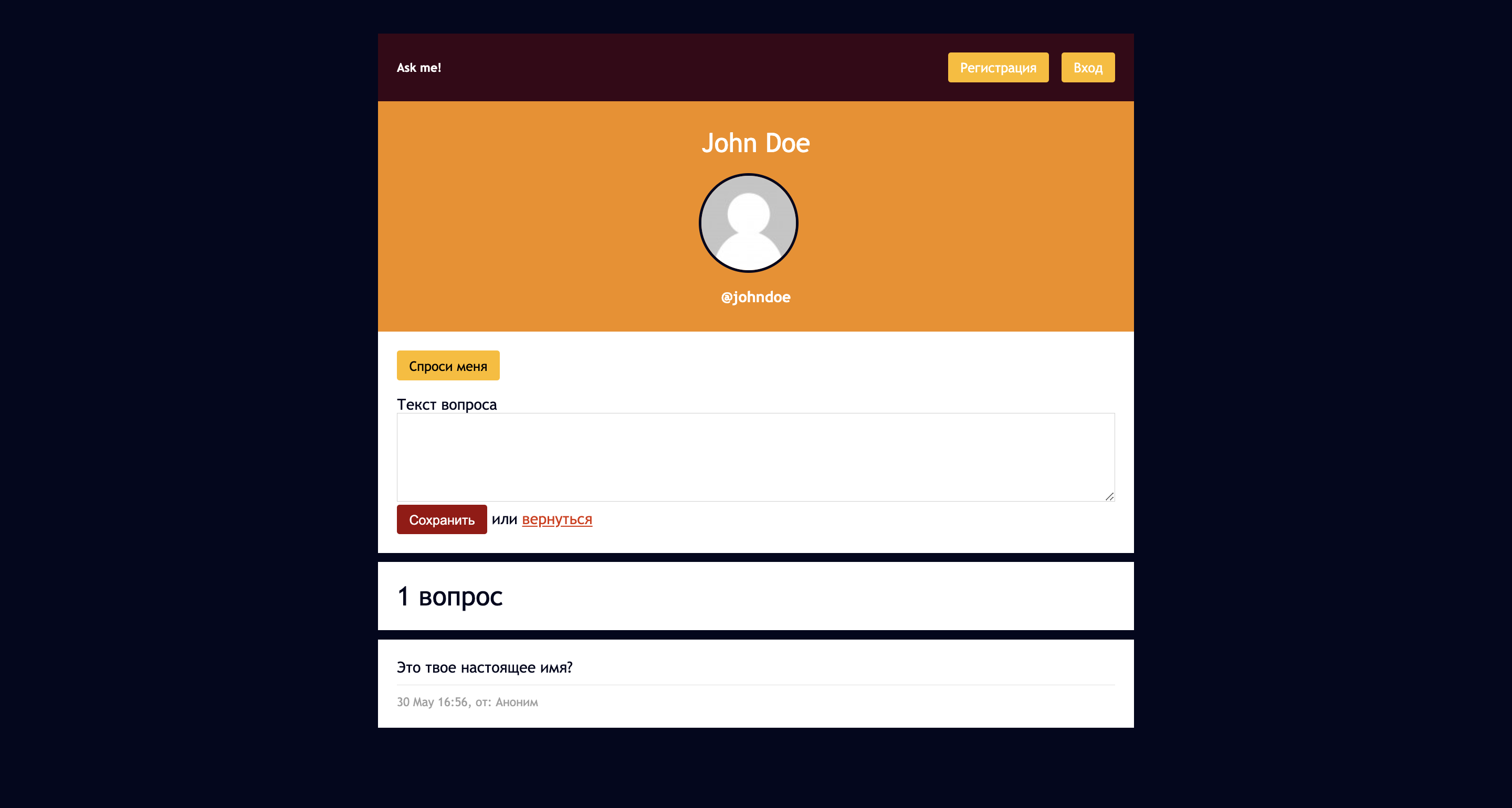


Рисунок 5 – Профиль пользователя

Если пользователь авторизован, в его аккаунте будет отображаться иконка редактирования профиля («карандаш»), при нажатии на которую откроется соответствующая страница – см. рис. 6. Важно отметить, что страница редактирования чужих профилей пользователю недоступна, и на других профилях иконка «карандаша» отображаться не будет.

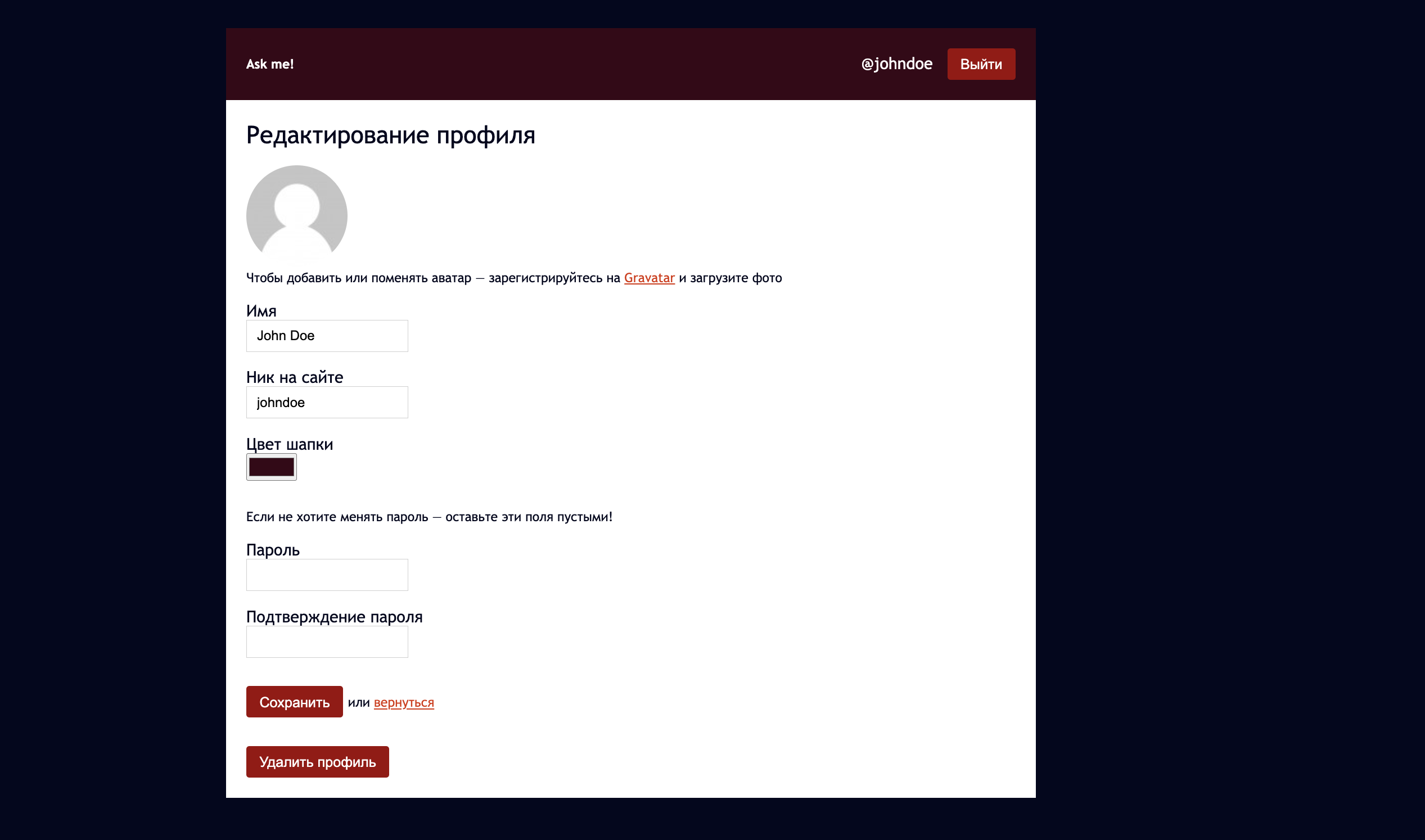


Рисунок 6 – Редактирование профиля

На данной странице можно изменить имя, никнейм и пароль, а также задать пользовательский цвет «шапки» (рис. 7). Также можно удалить профиль, при нажатии на эту кнопку высветится предупреждение (рис. 8).

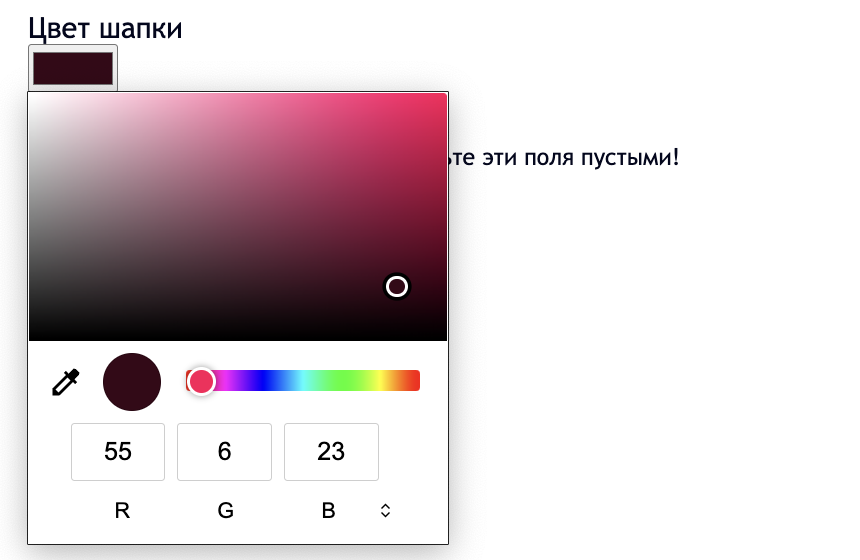


Рисунок 7 – Выбор цвета шапки

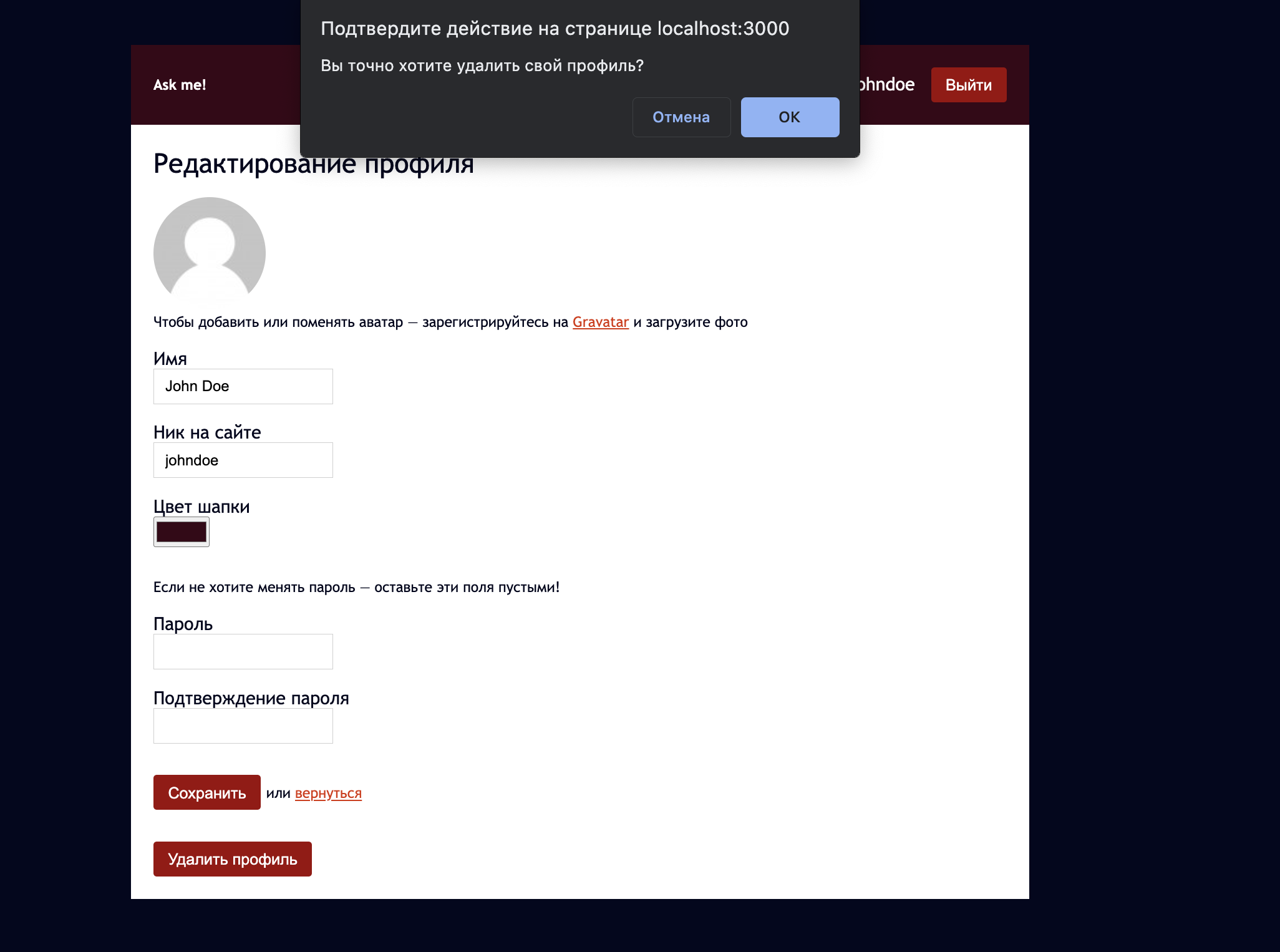


Рисунок 8 – Подтверждение удаления профиля

Чтобы задать вопрос другому пользователю, нужно перейти на его страницу и ввести текст вопроса в поле ввода, либо нажать на кнопку «Спроси меня», как показано на рис. 9.

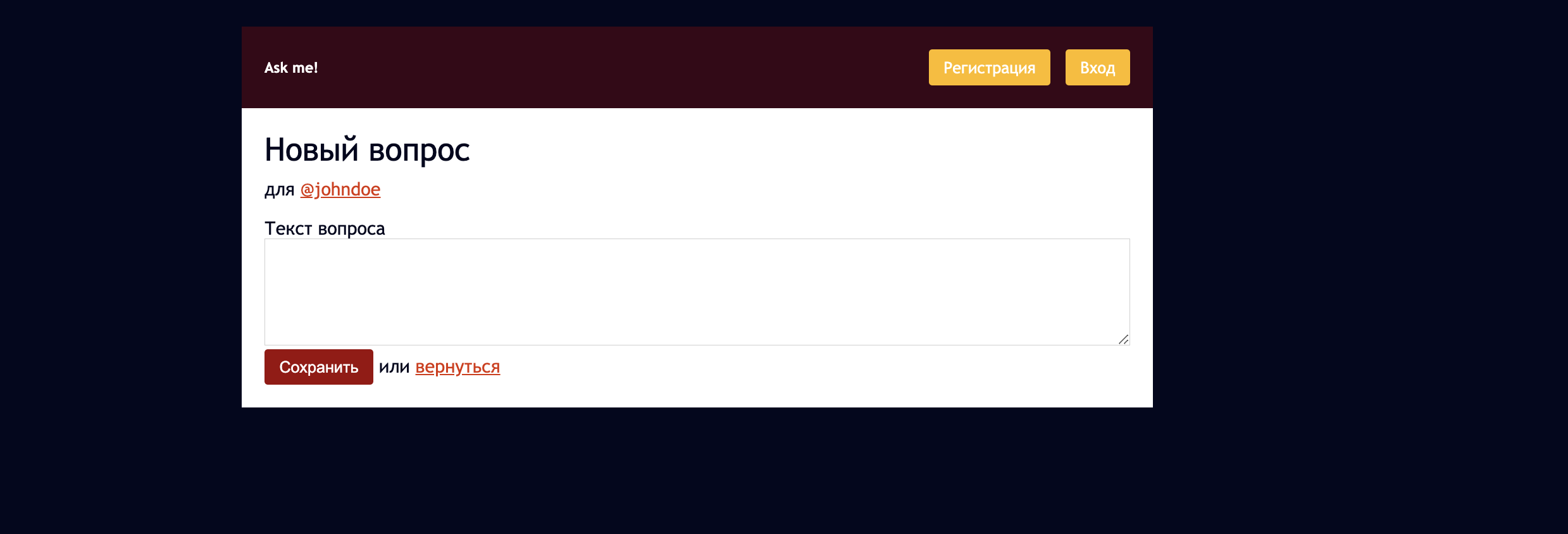


Рисунок 9 – Форма вопроса «Спроси меня»

Также есть возможность просмотреть отдельный вопрос, перейдя по ссылке «Вопрос #номер\_вопроса» на главной странице. На странице отобразится номер вопроса, его текст и ссылка на профиль пользователя, которому он был задан – см. рис. 10.

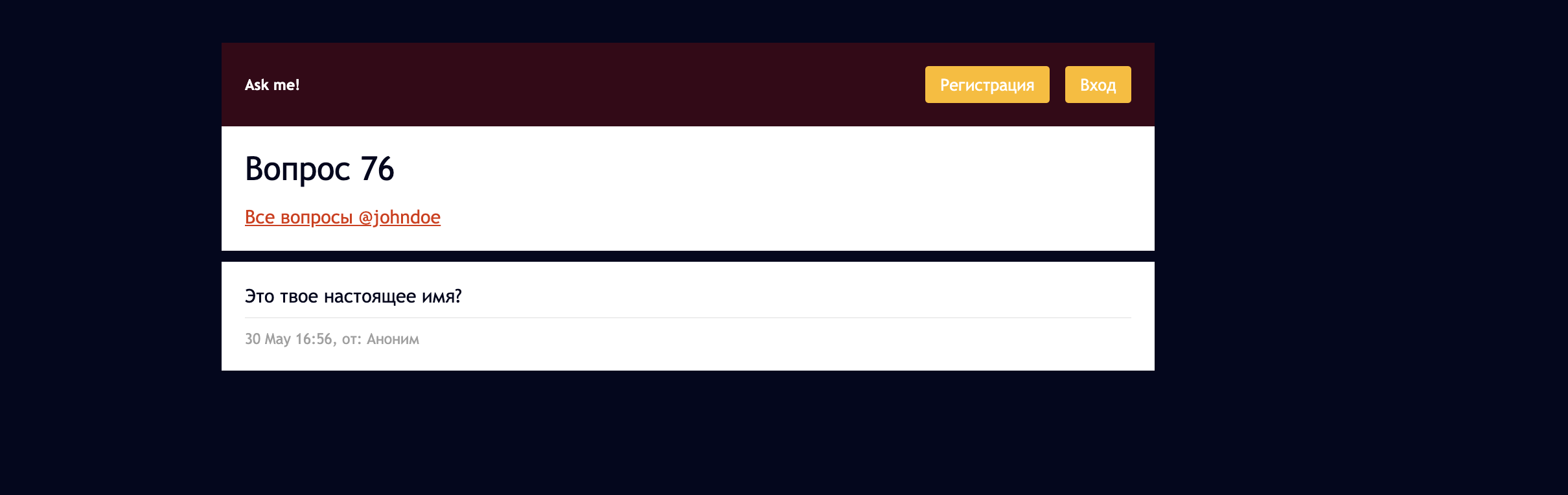


Рисунок 10 – Страница вопроса

У пользователя, получившего вопрос, есть возможность ответить на него, нажав на иконку ответа («стрелка»). После нажатия откроется форма редактирования вопроса с возможностью ввести ответ, как показано на рис. 11.

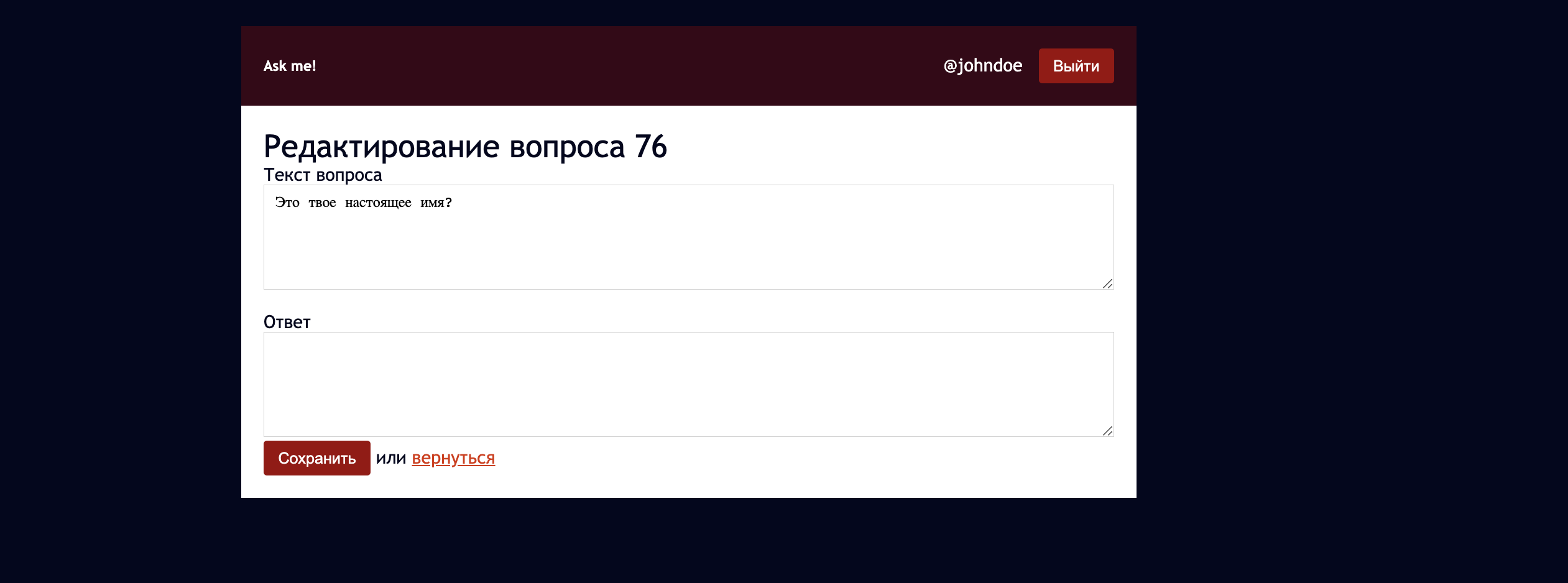


Рисунок 11 – Ответ на вопрос

После ответа на вопрос цвет ответа подсветится красным цветом, а иконка ответа изменится на иконку редактирования ответа («карандаш»), как показано на рис. 12.



Рисунок 12 – Обновленный вопрос с ответом

Также у пользователя, которому задан вопрос, есть возможность сделать вопрос скрытым. После нажатия на иконку скрытия («перечеркнутый глаз») текст вопроса станет полупрозрачным, а сам вопрос отметится в базе данных как скрытый.



Рисунок 13 – Скрытый вопрос

Вопрос можно удалить. При нажатии на иконку удаления вопроса также высветится предупреждение:

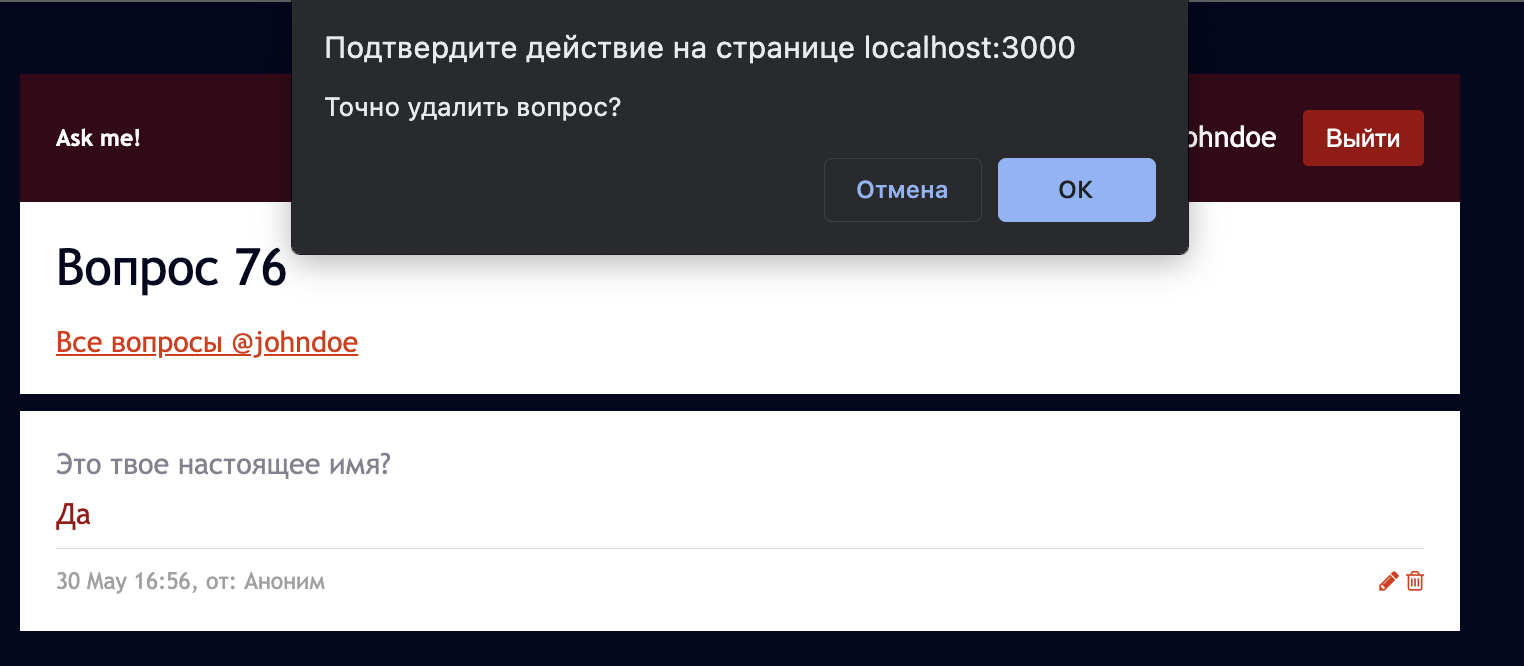


Рисунок 14 – Подтверждение удаления вопроса

В каждом вопросе отображается список хэштегов из текста самого вопроса или ответа (рис. 15). Если в обоих текстах присутствовали одинаковые хэштеги, они не дублируются. Регистр не учитывается. По ссылке можно перейти на страницу указанного хэштега и просмотреть все вопросы с ним (рис. 16).

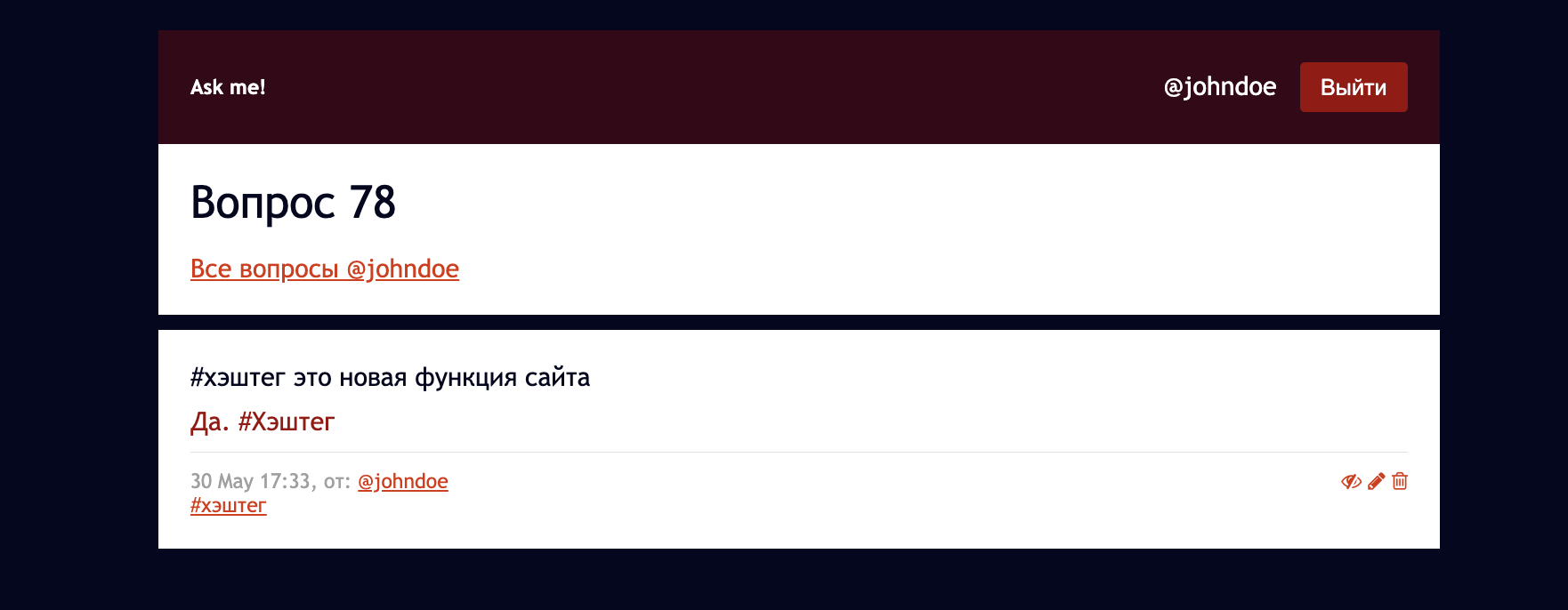


Рисунок 15 – Отображение хэштегов в вопросе



Рисунок 16 – Страница хэштега

1. Заключение

В рамках данной курсовой работы были изучены аспекты веб-разработки, которые являются необходимыми для создания клиент-серверных приложений и работы с базами данных. В процессе изучения материала было рассмотрено множество технологий и инструментов, которые используются в современной веб-разработке.

В работе также были рассмотрены дополнительные библиотеки, которые позволяют оптимизировать работу приложения как со стороны клиента, так и со стороны сервера. Такие библиотеки играют ключевую роль в создании высокопроизводительных веб-приложений.

Полученные знания и описанные в работе технологии могут быть применены при создании различных веб-приложений, начиная от простых сайтов и заканчивая масштабными интернет-сервисами.

1. Список литературы
2. ГОСТ 7.32-2017 «Отчет о научно-исследовательской работе»: [Электронный ресурс]. URL: <https://docs.cntd.ru/document/1200157208?roistat_visit=80026399>
3. Документация Ruby on Rails: [Электронный ресурс]. URL: <https://guides.rubyonrails.org/>
4. Библиотека гемов Ruby: [Электронный ресурс]. URL: <https://rubygems.org/>

Ссылка на git-репозиторий: