МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**
Кафедра математической кибернетики и компьютерных наук

РАЗРАБОТКА КЛИЕНТСКОЙ ЧАСТИ И ПОИСКОВОЙ СИСТЕМЫ САЙТА ДЛЯ МУНИЦИПАЛИТЕТОВ «ЕДИНОЕ МЕСТО»

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 411 группы
направления 02.03.02 — Фундаментальная информатика и информационные технологии
факультета КНиИТ
Мельникова Артемия Дмитриевича

Научный руководитель
зав. каф., доцент, к. ф. - м. н. _____ И. А. Батраева

Заведующий кафедрой
к. ф.-м. н., доцент _____ С. В. Миронов

Саратов 2024

ВВЕДЕНИЕ

В современном цифровом обществе веб-сайты играют ключевую роль в предоставлении информации и взаимодействии с пользователями. Особенно это важно для муниципальных образований, которые обязаны обеспечивать прозрачность своей деятельности и доступность услуг для граждан. Однако многие муниципальные сайты часто страдают от недостаточной функциональности, неудобного интерфейса и неэффективной поисковой системы, что затрудняет доступ к необходимой информации для пользователей.

Проект «Единое место» разрабатывался в качестве выпускной квалификационной работы по программе «Стартап как диплом». Он направлен на создание единой информационной платформы для муниципалитетов, где будут собраны все важные ресурсы в одном месте, что позволит получать всю необходимую информацию как представителям муниципалитетов, так и простым гражданам, заинтересованным в создании и развитии проектов своего города.

В данной дипломной работе рассматриваются основные этапы разработки клиентской части и поисковой системы сайта «Единое место». Исследуются современные подходы и технологии веб-разработки, анализируются требования пользователей и разрабатывается интерфейс, отвечающий этим требованиям. Особое внимание уделяется созданию интуитивно понятного интерфейса и оптимизации поисковых алгоритмов для обеспечения точного и быстрого поиска информации.

Таким образом, данная дипломная работа направлена на создание удобного и функционального веб-ресурса для муниципалитетов, который сможет значительно улучшить взаимодействие граждан с местными органами власти и упростить доступ к необходимой информации и услугам.

1 Описание проекта

1.1 Цель проекта

Цель проекта «Единое место» заключается в разработке удобной и функциональной клиентской части веб-сайта для муниципалитетов, а также эффективной поисковой системы, которая обеспечит гражданам быстрый и простой доступ к информации и услугам местных органов власти. Данная дипломная работа состоит из двух основных частей:

— Разработка быстрой, точной и гибкой поисковой системы для ресурсов сайта, которая позволит осуществлять поиск как по проектам и решениям, так и по документам, хранящимся на сайте. Поисковая система должна работать не только по названию документа, так и непосредственно по его содержанию, позволяя пользователю найти ресурс по информации, содержащейся в документе.
— Разработка клиентской части сайта, ориентированной на потребности и удобство пользователей. Интерфейс должен быть интуитивно понятным, простым и лаконичным, подстраиваться под разрешение экрана, и воспроизводить удобно предоставленную информацию за максимально короткое время.

Реализация этих задач будет способствовать созданию единой информационной платформы, которая упростит взаимодействие граждан с муниципальными органами власти, повысит уровень их уведомленности и удовлетворенности предоставляемыми услугами.

1.2 Архитектура проекта

Так как проект разрабатывался командой из двух человек, было принято решение использовать REST архитектуру для возможности независимо реализовывать отдельные части приложения. Это позволило реализовать легкую масштабируемость, высокую производительность и надёжность, что особенно важно для муниципальных сайтов, где требуется оперативная обработка большого объёма данных и обеспечение доступности для широкого круга пользователей.

Архитектура проекта содержит следующие компоненты:

— Клиент-сервер – отвечает за отображение данных, взаимодействие с поль-

зователем и отправку запросов к серверу;

- Сервер авторизации и регистрации – обеспечивает безопасность и управление доступом пользователей.
- База данных пользователей – обеспечивает хранение и управление данными о пользователях сайта «Единое место».
- Веб-сервер – обеспечивает обработку запросов от пользователей и взаимодействие с различными компонентами системы;
- Поисковая система – позволяет пользователям быстро и эффективно находить нужную информацию.
- База данных ресурсов – отвечает за хранение и управление всей информацией, доступной на сайте.

1.3 Обзор инструментальных средств

Для разработки каждого из «блоков» приложения были выбраны различные средства разработки, каждое из которых направлено на решение конкретной задачи.

Для разработки клиентской части приложения «Единое место» использовались следующие инструменты и технологии: JavaScript, React.js, React Router, React Bootstrap и Axios.

Основным инструментом для создания поисковой системы был выбран Elasticsearch — мощная поисковая и аналитическая система с открытым исходным кодом, созданная для работы с большими объемами данных в реальном времени. Для интеграции с Elasticsearch было принято решение использовать язык Python. За работу с веб-сервером, получением и передачей информации в поисковой системе отвечает фреймворк Flask.

2 Реализация поисковой системы

2.1 Структура поисковой системы

Для создания поисковой системы необходим индекс – основной компонент, обеспечивающий организацию, хранение и эффективный доступ к данным. Поисковая система состоит из трёх индексов:

- document_index – главный индекс, в котором будут содержаться все документы (проекты и решения) для поиска в системе.
- project_index – индекс для проектов.
- decision_index – индекс для решений.

При наличии этих трех индексов пользователь сможет осуществлять поиск по трем отдельным пунктам: документам, проектам и решениям.

Каждый индекс состоит из двух полей:

- id объекта – вся информация о объектах, существующих на сайте (проекты, решения, документы), хранится в базе данных.
- Текстовое содержание документа – основным преимуществом данной поисковой системы является возможность находить интересующий пользователя объект по хранящейся в нем информации. Для этого необходимо было реализовать алгоритм извлечения текста из документа и использовать этот текст как основное поле для поиска в индексе.
  Аналогичная структура присутствует в decision_index и project_index.

2.2 Реализация индексации объектов

. Для реализации индексации объектов в поисковой системе были решены следующие задачи:

1. Получение объекта с веб сервера.
2. Чтение текста из объекта для дальнейшего добавления этого текста в соответствующий элемент индекса.
3. Индексация документа – добавление элемента в поисковик с двумя полями: id документа и текстом, содержащемся в этом документе.

Извлечение текста из документов. Для извлечения текста из докумен-

тов форматов .docx использовались две библиотеки языка python: docx, которая позволяет работать с документами этих форматов, и textract – библиотека, позволяющая извлекать текст из документов.

Для извлечения текста из документов формата .pdf использовалась библиотека PyPDF2 – библиотека на Python, предназначенная для работы с PDF-файлами. Она предоставляет функциональность для чтения, обработки и создания PDF-документов.

Индексация. Основная функция для индексации определяет тип файла, и, исходя из этого, применяет к файлу функцию извлечения текста, работающую с типом этого файла, после чего индексирует документ и проект с соответствующим документом в индексы document_index и project_index соответственно. Эта функция обеспечивает индексацию любого документа независимо от его формата.

Далее необходимо реализовать обработчик запроса с сервера. Разработанная функция upload_file позволяет пользователю загружать файл через POST-запрос на сервер, сохранять его на диске и индексировать документ. Она также обрабатывает различные типы ошибок, обеспечивая надежность работы веб-приложения.

2.3 Поиск объектов

Алгоритм поиска объектов по запросу должен выглядеть следующим образом:

1. Получение GET - запроса с веб-сервера с аргументом query, в котором содержится введенный пользователем запрос.
2. Поиск объектов по соответствующему индексу
3. Возвращение id найденных документов на веб-сервер в формате json.

Так как каждый из индексов имеет одинаковую структуру, для поиска объектов можно реализовать одну функцию, которая будет принимать в качестве аргументов текст запроса и наименование индекса, по которому будет совершен поиск.

Elasticsearch анализирует как текст запроса, так и текст поля, в котором ведется поиск. После этого можно получить результаты поиска и вернуть список id найденных объектов. Если поиск успешен, извлекаются результаты поиска.

Каждый найденный документ (hit) добавляется в список результатов. В случае ошибки функция возвращает пустой список, указывая, что результаты поиска отсутствуют.

Для обработки запросов с веб-сервера было создано три отдельных функции, идентичных по структуре, которые получают GET - запрос, и вызывают функцию поиска с переданными в запросе аргументами, после чего, полученные из поиска данные передаются на веб-сервер в формате json. В случае отсутствия результатов поиска функция возвращает пустой список. Данные о найденных объектах также возвращаются в формате json для того, чтобы минимизировать их обработку на сервере и клиенте.

2.4 Перевод проектов в решения

Реализация перевода проекта в решения в поисковой системе необходима для того, чтобы после этого действия объект переноса переставал выдаваться на сайте в поиске проектов, и стал доступен для поиска на странице «Решения». Для этого нужно удалить информацию об объекте из индекса projects_index в Elasticsearch, и добавить эту информацию в индекс decision_index. Запрос на перевод проекта в решение передается методом POST с телом, в котором содержится id объекта в формате json.

Для обработки запроса с сервера была создана отдельная функция, принимающая POST-запрос, извлекающая из него id объекта для переноса, и передающая его в функцию replace_document.

3 Реализация клиентской части

3.1 Структура клиентской части проекта

Клиентская часть проекта состоит из 7 основных элементов:

1. Главная страница – всречает пользователя при входе на сайт, должна содержать слайдер с выводом трех случайных новостей, с возможностью перехода на каждую из них.
2. Новости – страница, в которой будут отображаться последние события, происходящие на сайте.
3. Проекты – страница, позволяющая осуществлять пользователям поиск по всем проектам, находящимся на сервере.
4. Решения – страница, позволяющая осуществлять пользователям поиск по всем решениям, находящимся на сервере.
5. Документы – страница, позволяющая осуществлять пользователям поиск по всем документам, находящимся на сервере.
6. Контакты – страница, на которой должны находиться все контакты с организациями и деятелями сайта, напимер, переходить на официальные сообщества муниципалитетов в телеграме.
7. Личный кабинет пользователя – страница, в которой пользователь будет видеть всю основную информацию, связанную с ним.

3.2 Навигационная панель сайта и маршрутизация

Навигационная панель сайта. Для создания навигационной панели сайта были использованы две основных библиотеки: react и react bootstrap. Эти инструменты позволяют создать удобный и интуитивно понятный интерфейс для пользователей, а также организовать навигацию по различным страницам приложения.

Основным компонентом для создания навигационной панели являются Navbar и Nav. Компонент Navbar из react-bootstrap используется для создания адаптивной навигационной панели. Навигационная панель остается закрепленной в верхней части экрана и адаптируется к размеру экрана. При измененнии размеров окна навигационная панель сворачивается в компактное меню. Это меню можно развернуть для того, чтобы раскрыть его в вертикальном состоянии. Оно содержит ссылки на основные разделы сайта.

Отдельно отображения ссылок «Вход/Регистрация» и «Личный кабинет»

был создан алгоритм проверки на автоизацию пользователя. При входе пользователя в систему ссылка на регистрацию и вход меняется на ссылку для перехода в личный кабинет пользователя. При выходе пользователя с сайта применяется функция, очищающая локальное хранилище и изменяющая состояние авторизации.

На навигационной панели находится переход на регистрацию и вход. Для реализации этой функции было принято решение использовать модальное окно.

Окна регистрации и авторизации. Страница регистрации выглядит следующим образом:

— Поддерживается вход с помощью логина и пароля.
— Присутствует вход с помощью социальной сети ВКонтакте.
— Присутствует ссылка на регистрацию, если у пользователя отсутствует аккаунт, но он не хочет регистрироваться с помощью социальной сети.
— Присутствует ссылка на изменение пароля в случае его утери.

Окно регистрации в проекте «Единое место» предоставляет пользователям возможность создать новый аккаунт. При нажатии на ссылку для регистрации и входа перед пользователем появляется соответсвующее модальное окно.

При успешном входе в систему ссылка для входа и регистрации в хэдере сайта меняется на ссылку на личный кабинет пользователя.

Маршрутизация на сайте. Маршрутизация является важным аспектом веб-приложений, позволяющим пользователям переходить между различными страницами или компонентами без перезагрузки страницы. В экосистеме React для этой цели широко используется библиотека React Router.

Для настройки маршрутизации были созданы компоненты, в которых осуществляется работа других страниц сайта: главной страницы сайта, проектов, решений, контактов, документов, и новостей. После чего эти компоненты были импортированы для указания их в качестве перехода на соответствующие страницы. Также были импортированы компоненты для управления маршрутизацией. Используя компонент Router, каждая страница была помещена в отдельный маршрут Route, указывая путь маршрутизации path, и компонент element, который должнен открываться при переходе по соответствующему маршруту

на сайте.

3.3 Клиентская часть поисковой системы

Поисковая система необходима для трех страниц сайта: «Решения», «Проекты» и «Документы». Для её реализации на стороне клиента для каждой страницы необходимо было выполнить несколько задач:

— Разработка интерфейса поисковой системы.
— Отправка запроса для поиска.
— Обработка и форматированный вывод объектов поиска.
— Реализация возможности просмотра разного количества результатов поиска.

Интерфейс поисковой системы. Интерфейс поисковой системы состоит из следующих объектов:

— Текстовая строка для ввода информации, которую пользователь хочет найти.
— Кнопка для нахождения информации. По нажатию на нее вызывается основная обрабатывающая функция.
— Поле с выбором количества объектов на одной странице. В нем пользователь сможет выбрать количество объектов, которое должно располагаться на одной странице.
— Кнопки для переключения между страницами поиска и информирование пользователя о том, на какой странице он находится в данный момент.
— Форма для корректного вывода объектов.

Основная функция для поиска устанавливает правильное значения текущей страницы и количества страниц, после чего проверяет, находится ли в данный момент что-то в строке поиска. Если в строке присутствует информация, то функция отправляет GET-запрос на веб сервер со всей необходимой информацией для поиска, а именно: текстом, текущей страницой поиска, и количеством элементов на странице. Для отправки запроса на сервер используется библиотека Axios.

В случае отсутствия информации в запросе вызывается специальная функция, которая получает с сервера все документы, отсортированные по мере их добавления на веб сервер.

Она действует схожим образом с функцией поиска, однако запрос, который она отправляет, не требует текстового параметра, и помимо документов также возвращает их общее количество в базе данных. Эта функция вызывается не только при попытке поиска с пустой поисковой строкой, но и при переходе на любую из поисковых страниц. Реализуется это при помощи хука useEffect из библиотеки React.

Для реализации переключения страниц поиска были созданы две функции: handleNextPage и handlePreviousPage. Функция handlePageSizeChange предназначена для обработки изменения размера страницы, то есть количества элементов, отображаемых на одной странице результатов.

Отображение списка результатов поиска реализовано в виде группы элементов, каждый из которых представляет собой отдельный документ.

3.4 Реализация новостной ленты

Реализация новостной ленты в веб-приложении включает несколько шагов, начиная с создания пользовательского интерфейса и заканчивая интеграцией с сервером для получения новостей. Алгоритм, описывающий основные этапы реализации новостной ленты:

1. Определение требований и структуры данных. Перед началом реализации необходимо определить, какие данные будут отображаться в новостной ленте. Основные поля могут включать:
   - Заголовок новости (title)
   - Дата публикации (date)
   - Автор новости (publisher)
   - Текст новости (text)
   - Документ, связанный с новостью (document)
2. Внутри компонента News определены состояния, которые будут управлять данными новостей, текущей страницей, состоянием загрузки и наличием дополнительных новостей.
3. Для реализации функции загрузки новостей была создана функция fetchNews, которая отправляет GET-запрос на сервер для получения новостей. Эта функция вызывается при загрузке новостей и при прогрузке страницы. Запрос отправляется на сервер по указанному URL, с указанием номера страницы и размера страницы (количество новостей на странице). С по-

мощью хука useEffect данная функция сразу же вызывается при открытии страницы.

Таким образом, функция загрузки данных с сервера и обработка событий прокрутки обеспечивают динамическую загрузку новостей при прокрутке страницы. Такой подход обеспечивает удобный и интерактивный интерфейс для пользователя.

3.5 Создание личного кабинета пользователя

При успешной авторизации у пользователя появляется возможность зайти в личный кабинет. Здесь он может совершать следующие действия:

1. Просматривать свою историю поиска: пользователя будет видеть открытые им ранее проекты, решения, и документы.
2. Если роль позволяет пользователю, он может также совершать следующие действия:
   - Создавать новые проекты.
   - Добавлять шаги к существующим проектам, созданным этим пользователем.
   - Переводить проекты, созданные пользователем, в решения.

Создание проектов. Форма для создания проектов была реализована как модальное окно, которое позволяет пользователю вводить название проекта, описание и загружать файл, а затем отправлять эти данные на сервер.

В качестве прикрепленного файла к проекту можно добавить документы трех форматов: .doc, .docx и .pdf. Это реализовано путем добавления к форме для загрузки файла объекта Form.Control с соответствующими ограничениями.

Для обработки загрузки файла была создана функция handleFileChange. После заполнения формы и нажатия на кнопку «Создать» данные отправляются на сервер. Отправка данных на сервер осуществляется путем POST-запроса с передачей в его тело formData.

Отображение проектов пользователя. Компонент UserProjects отображает список проектов, созданных пользователем. Он поддерживает динамическую подгрузку проектов при прокрутке страницы, а также предоставляет возможность добавления шагов к проектам и перевода проектов в решения. Основная функция для получения проектов текущего пользователя отправля-

ет GET-запрос на сервер с параметрами текущего пользователя, страницей, и номером страницы. При открытии личного кабинета она сразу же вызывается с помощью хука useEffect.

Динамическая подгрузка проектов релизуется аналогичным с динамической подгрузкой новостей образом.

У каждого элемента, в котором содержится проект, помимо основной информации и возможности просмотра проекта присутствуют две новые кнопки: кнопка для добавления шага в проект, и кнопка для перевода проекта в решения. Каждая из этих кнопок вызывает соответствующее модальное окно, которое позволяется выполнить эти действия.

Добавление шагов к проектам. Это модальное окно отображается, когда пользователь хочет добавить шаг к существующему проекту. Пользователю предлагается заполнить название шага и его краткое описание.

После нажатия на кнопку «Добавить» и проверки всех полей на веб сервер отправляется POST-запрос с телом json формата, в котором содержится id документа, название шага, и его описание. Это состояние изменяется при уставовке других данных в поля, а id берется из самого проекта.

За отправку запроса на сервер отвечает основная функция модуля handleSubmit. Она отправляет POST-запрос с телом формата json. В json файле содержится вся информация, находящаяся в состоянии formData до отправки, после чего модальное окно закрывается.

Шаги, добавленые к проекту, можно посмотреть, зайдя на страницу соответствующего проекта.

Перевод проектов в решения. За перевод проекта в решение так же ответственно модальное окно. Оно будет еще раз уточнять у пользователя, действительно ли он желает это сделать.

После подтверждения действия пользователем на веб сервер будет отправляться POST-запрос с телом json формата, в котором содержится id проекта для перевода.

В случае успешного запроса запись в базе данных о проекта и связанная с ним информация для поиска будет изменена. Теперь его можно будет найти на странице для поиска решений.

ЗАКЛЮЧЕНИЕ

В рамках дипломного проекта была разработана и реализована система управления проектами. Основной целью проекта было создание интуитивно понятного и функционального веб-приложения, которое позволяет легко управлять своими проектами, добавлять новые шаги и просматривать существующие проекты.

Данный дипломный проект демонстрирует практическое применение современных веб-технологий для создания сложных интерактивных приложений. Разработанная система управления проектами обладает следующими преимуществами:

- Удобство использования: интуитивно понятный интерфейс и простота в использовании делают систему доступной для широкого круга пользователей.
- Надежность: продуманный механизм обработки ошибок и взаимодействия с сервером обеспечивает стабильную работу приложения.
- Скорость и удобство поиска: благодаря поисковой системе с использованием Elasticsearch пользователи могут максимально быстро найти интересующую их информацию.

В будущем проект можно расширить и улучшить добавлением новых функциональных возможностей, например, интеграцией функций, таких как уведомления о новых шагах проекта, комментарии и совместная работа над проектами.

В заключение, разработанная система управления проектами демонстрирует успешное применение современных технологий и подходов в веб-разработке, предлагая пользователям удобный инструмент для управления и контроля своих проектов.