Министерство науки и высшего образования РФ

Федеральное государственное бюджетное образовательное

учреждение высшего образования

«Уфимский университет науки и технологий»

Факультет информатики и робототехники

Кафедра вычислительной математики и кибернетики

Отчет Лабораторной работе №3

«Создание веб-приложения. Применение подхода AJAX для обмена данными с сервером.»

Выполнили:

студенты группы МО-321 Шарипов А.Г., Дементьев Д.В.

Проверил:

Юдинцев Б.С.

Уфа, 2023

**Задание:**

1. Создать базовый проект веб-приложения, используя языки программирования Python или Node.js.
2. Реализовать созданные ранее контентные html-страницы с помощью любого популярного шаблонизатора, который поддерживается соответствующими фреймворками:

* создать мастер-страницу и вынести изменяемый контент в отдельные шаблоны;
* заголовок и титул страницы передавать в виде строкового аргумента в шаблон;
* используемые списки (элементы <ul>) реализовать с помощью циклических конструкций в шаблоне.

1. Вынести встроенный JavaScript код и статичные файлы в соответствующие папки веб-приложения.
2. Привязать к вашим шаблонам страниц соответствующие маршруты и обработчики маршрутов.
3. Ознакомиться с понятием «метод HTTP-запроса», рассмотреть методы GET и POST.
4. Создать контентную страница с формой для отправки данных (если еще не создана), предусмотреть на данной странице элемент для вставки контента/текста. Реализовать заполнение данного элемента, используя подход AJAX для фоновой обработки запроса к вашему веб-приложению.

**Теоретические сведения:**

**Колбэк-функция** - это функция, переданная в другую функцию в качестве аргумента, которая затем вызывается по завершению какого-либо действия.

**AJAX** - подход к построению интерактивных пользовательских интерфейсов веб-приложений, заключающийся в «фоновом» обмене данными браузера с веб-сервером. В результате при обновлении данных веб-страница не перезагружается полностью, и веб-приложения становятся быстрее и удобнее.

При использовании **AJAX**:

1. Пользователь заходит на веб-страницу и нажимает на какой-нибудь её элемент;
2. Скрипт (на языке JavaScript) определяет, какая информация необходима для обновления страницы;
3. Браузер отправляет соответствующий запрос на сервер;
4. Сервер возвращает только ту часть документа, на которую пришёл запрос;
5. Скрипт вносит изменения с учётом полученной информации (без полной перезагрузки страницы).

**Шаблонизатор -** это программное обеспечение, позволяющее использовать html-шаблоны для генерации конечных html-страниц. Основная цель использования шаблонизаторов — это отделение представления данных от исполняемого кода. Часто это необходимо для обеспечения возможности параллельной работы программиста и дизайнера-верстальщика. Такой подход значительно ускоряет время разработки и прототипирования приложения, дизайнеру не нужно вникать в программирование, а программисту беспокоиться об интерфейсе.

**Маршрутизация** - процесс, который отвечает за определение обработчика для конкретной запрашиваемой страницы

**Виды HTTP-запросов:**

* GET- Позволяет запросить некоторый конкретный ресурс. Дополнительные данные могут быть переданы через строку запроса (Query String) в составе URL.О составляющих URL мы поговорим чуть позже.
* POST - Позволяет отправить данные на сервер. Поддерживает отправку различных типов файлов, среди которых текст, PDF-документы и другие типы данных в двоичном виде. Обычно метод POST используется при отправке информации (например, заполненной формы логина) и загрузке данных на веб-сайт, таких как изображения и документы.
* HEAD - Здесь придется забежать немного вперед и сказать, что обычно сервер в ответ на запрос возвращает заголовок и тело, в котором содержится запрашиваемый ресурс. Данный метод при использовании его в запросе позволит получить только заголовки, которые сервер бы вернул при получении GET запроса к тому же ресурсу. Запрос с использованием данного метода обычно производится для того, чтобы узнать размер запрашиваемого ресурса перед его загрузкой.
* PUT - Используется для создания (размещения) новых ресурсов на сервере. Если на сервере данный метод разрешен без надлежащего контроля, то это может привести к серьезным проблемам безопасности.
* DELETE - Позволяет удалить существующие ресурсы на сервере. Если использование данного метода настроено некорректно, то это может привести к атаке типа «Отказ в обслуживании» (Denial of Service, DoS) из-за удаления критически важных файлов сервера.
* OPTIONS - Позволяет запросить информацию о сервере, в том числе информацию о допускаемых к использованию на сервере HTTP-методов.
* PATCH - Позволяет внести частичные изменения в указанный ресурс по указанному расположению.

|  |  |  |
| --- | --- | --- |
| var indexRouter = require('./routes/index'); | <http://127.0.0.1:3000/index> | Путь к главной странице. |
| var aboutRouter = require('./routes/about'); | <http://127.0.0.1:3000/about> | Путь к странице с информацией о разработчиках. |
| var usersRouter = require('./routes/contact'); | <http://127.0.0.1:3000/contact> | Путь к странице с информацией о контактах. |
| var unityRouter = require('./routes/unity'); | <http://127.0.0.1:3000/unity> | Путь к странице с информацией о проектах на Unity. |
| var androidRouter = require('./routes/android'); | <http://127.0.0.1:3000/android> | Путь к странице с информацией о проектах на Android. |
| var blogRouter = require('./routes/blog'); | <http://127.0.0.1:3000/blog> | Путь к станице с блогом. |

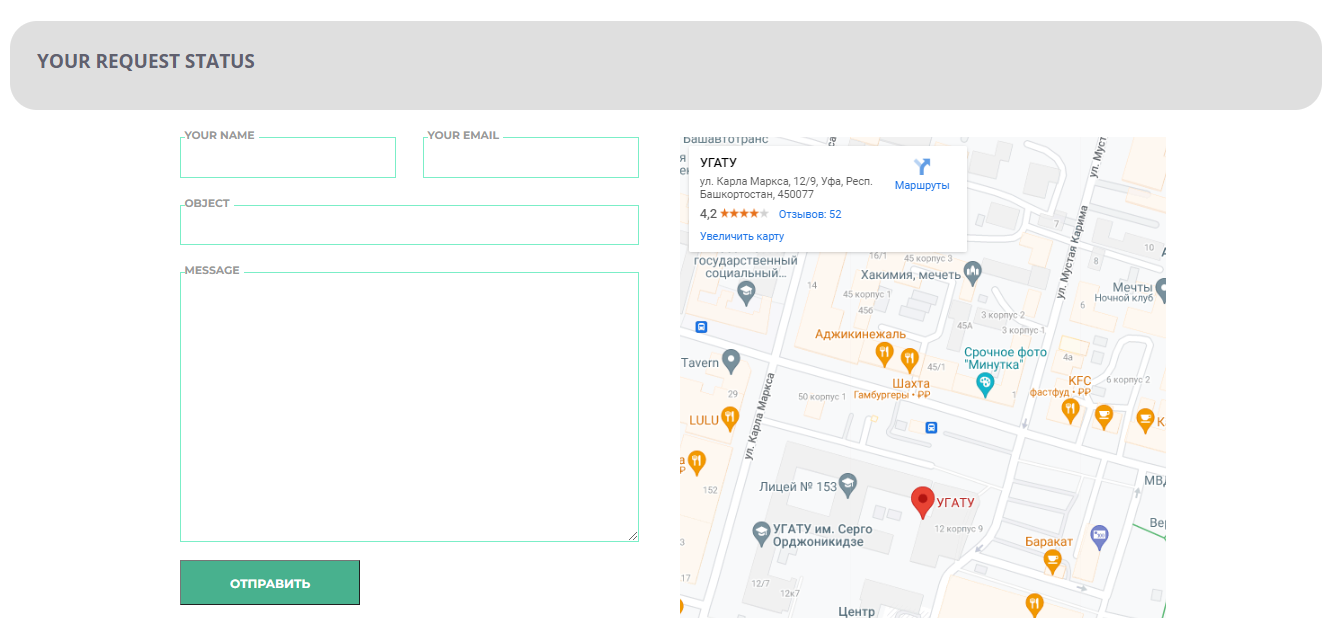
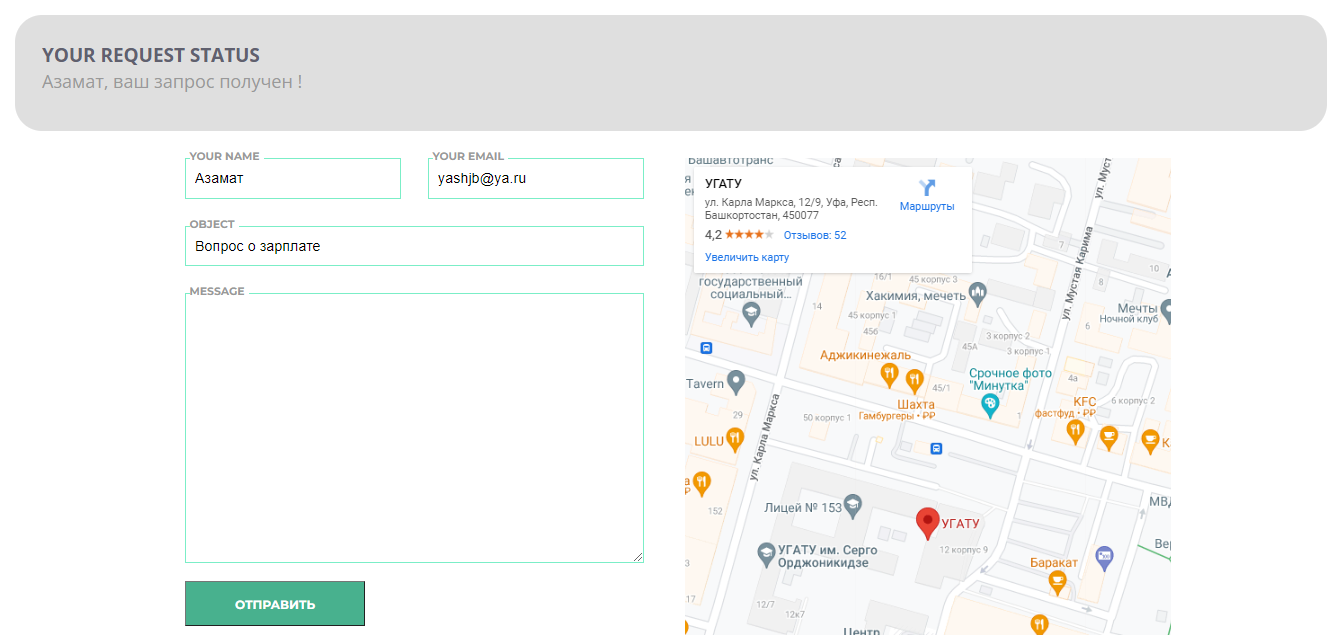


Рис.1(Интерактивная форма с регистрацией до ввода данных)

 Рис.2(Интерактивная форма с регистрацией после ввода данных )

var sendbtn = document.getElementById("sendbtn");    *// выбираем DOM-елемент (кнопку)*

console.log(sendbtn)

*// Привязываем к элементу обработчик события "click"*

sendbtn.addEventListener("click", function (e) {

*/\* Инструкция preventDefault позволяет переопределить стандартное поведение браузера,*

*если ее убрать, то браузер по-умолчанию обновит страницу после отправки данных формы \*/*

    e.preventDefault();

*// Получаем данные полей формы*

    let fname = document.getElementsByName("name")[0].value;

    let lname = document.getElementsByName("email")[0].value;

    let reqtype = document.getElementsByName("object")[0].value

*// Преобразуем полученные данные в JSON*

    let formdata = JSON.stringify({ firstname: fname, lastname: lname, reqtype: reqtype});

*/\* XMLHttpRequest предоставляет простой способ получения данных по ссылке без перезагрузки страницы.*

*Это позволяет обновлять только часть веб-страницы не прерывая пользователя.*

*XMLHttpRequest используется в AJAX запросах и особенно в single-page приложениях.\*/*

    let request = new XMLHttpRequest();

*// Инициализируем POST-запрос на адрес "/", аргумент true - асинхронный запрос*

    request.open("POST", "/contact/ajaxrequest", true);

*// Устанавливаем заголовки запроса (тип отсылаемых данных)*

    request.setRequestHeader("Content-Type", "application/json");

*// Привязываем к запросу обработчик события "load" - окончание загрузки данных*

    request.addEventListener("load", function () {

*// Получаем и парсим ответ сервера*

        let receivedUser = JSON.parse(request.response);

*// Выводим ответ сервера в элементе c id = "statusfield"*

        let statfield = document.getElementById("statusfield");

        statfield.textContent = receivedUser.message;

        statfield.textContent.bold();

    });

*// После привязки обработчика событий к запросу отправляем данные*

    request.send(formdata);

});

**Вывод:**

В ходе работы были рассмотрены методы создания web приложений с использованием шаблонизаторов и маршрутизаторов. Был применен метод AJAX для фоновой обработки запроса к веб-приложению.