Лабораторная работа 04

ПСКП

ПОИТ-3

**Задание 01**

1. Разработайте серверное приложение **04-01**, которое принимает HTTP-запросы **GET**, **POST**, **PUT**, **DELETE** на **<http://localhost:5000/api/db>**.
2. Назначение запросов.

|  |  |
| --- | --- |
| **GET** | Получить все строки таблицы базы данных (БД). В теле ответа возвращается массив строк в json-формате. |
| **POST** | Добавить новую строку в таблицу БД; строка передается серверу в json-формате.  В теле ответа возвращается измененная строка в json-формате. |
| **PUT** | Изменить существующую строку таблицы БД; строка передается серверу в json-формате. |
| **DELETE** | Удалить существующую строку (по id) в таблице БД; серверу передается id (в query-строке). В теле ответа возвращается удаленная строка в json-формате.  Пример: **<http://localhost:5000/api/db?id=25>** |

1. Для работы с БД в приложении должен быть разработан специальный объект **DB**, который должен располагаться в отдельном модуле.
2. Объект **DB** должен уметь генерировать и ***асинхронно*** обрабатывать следующие события

|  |  |
| --- | --- |
| **GET** | Получить все строки таблицы базы данных (БД). |
| **POST** | Добавить новую строку в таблицу БД. |
| **PUT** | Изменить существующую строку таблицы БД. |
| **DELETE** | Удалить существующую строку в таблице БД. |

1. Для выполнения операций с БД объект **DB** должен обеспечивать следующие функции

|  |  |
| --- | --- |
| **select** | Получить все строки таблицы базы данных (БД). |
| **insert** | Добавить новую строку в таблицу БД. |
| **update** | Изменить существующую строку таблицы БД. |
| **delete** | Удалить существующую строку в таблице БД. |

1. Строка таблицы содержит:

- идентификатор (**id**);

- имя (**name**);

- дата рождения (**bday**).

1. Для имитации таблицы БД используйте JS-массив, а для имитации операций с БД соответствующие операции с массивом.
2. Проверьте работоспособность приложения с помощью **POSTMAN**.

**Задание 02**

1. Разработайте серверное приложение **04-02**, использовав код приложения **04-01**.
2. Дополнительно к функциональности унаследованной от **04-01** приложение **04-02** должно отвечать на GET-запросы к **<http://localhost:5000/>.**
3. В ответ на GET-запрос к **<http://localhost:5000/>** приложение **(**сервер) пересылает клиенту HTML-страницу, которая должна располагаться в том же директории, что и js-файл приложения**.**
4. HTML-страница должна обеспечивать ввод необходимых данных, отправку всех типов запросов к серверу и обработку полученных результатов с применением функции **fetch**.
5. Проверьте работоспособность приложения с помощью браузера.

**Задание 03.Ответьте на следующие вопросы**

1. **Дайте пояснению понятию «событие программного объекта».**

**Событие** – это сигнал о том, что произошло определённое действие (например, клик мыши, нажатие клавиши, завершение загрузки данных).

Событие состоит из **трёх элементов:**

* **Источник события** – объект, который вызывает событие.
* **Обработчик события** – функция, реагирующая на событие.
* **Подписчик** – объект, который подписан на событие.

**Примеры**:

* В C# событием может быть клик по кнопке.
* В JS событием может быть наведение мыши на элемент.
* В Node.js событием может быть получение запроса на сервер.

1. **Объясните механизм генерации и обработки событий в C#.**

В C# события работают на основе **делегатов**.

Механизм работы:

* Создаём делегат – «шаблон» для событий.
* Создаём событие на основе делегата.
* Подписываем обработчики (методы) на это событие.
* Вызываем событие (генерируем его).

1. **Поясните как самостоятельно реализовать механизм генерации и обработки событий на JS или С++.**

В JS события обрабатываются через **EventEmitter** (Node.js) или встроенные механизмы браузера (**addEventListener**).

В C++ можно реализовать механизм событий с помощью callback-функций или std::function.

1. **Какой встроенный механизм используется в Node.js для генерации и обработки событий. Поясните принцип его работы.**

В Node.js используется EventEmitter – встроенный класс для работы с событиями.

**Принцип работы:**

* Создаём объект EventEmitter.
* Добавляем обработчик с помощью .on().
* Генерируем событие с помощью .emit().