Министерство транспорта Российской Федерации

Федеральное государственное автономное

образовательное учреждение высшего образования

«Российский университет транспорта (МИИТ)»

Институт транспортной техники и систем управления

Кафедра «Управление и защита информации»

Лабораторная работа №4

по дисциплине:

«Web-программирование»

на тему:  
«Последовательный и разветвляющийся вычислительные процессы»

Выполнил: ст. гр. ТКИ-541

Ковров А.И. Панаргин В.М.

Вариант №9

Проверил: к.т.н., доцент Сафронов А.И.

Москва – 2025 г.

# **Цель работы**

Изучение основных директив Vue.js (v-bind, v-on, v-model, v-if, v-else) и их практическое применение для создания реактивных пользовательских интерфейсов.

# **2. Формулировка задачи**

Разработать четыре простых локальных одностраничных *web*-приложение (*LSPWA*) под управлением фреймворка *Vue.js* на языке *JavaScript* в соответствии с указаниями вариантов индивидуального задания (покрываются три раздела последовательного вычислительного процесса и один раздел разветвляющегося вычислительного процесса). Допустимо выполнение в синтаксисе как *Vue.js 2.x*, так и *Vue.js 3.x*.

*Базовая структура отчёта по работе:*

1. Цель работы.

2. Формулировка задачи.

3. Технология **локального** (**LSPWA**) подключения фреймворка *Vue.js.*

4. Таблица соответствия переменных и методов, используемых в *web*-приложениях (для случаев, когда переменные названы не значимо).

5. Содержательная часть по **параметрической вставке** (*v-bind;* код *web*-приложения + отображение в браузере (с учётом изменяющихся состояний) + схема ТП согласно нотации, определённой рабочим коллективом).

6. Содержательная часть по **параметрической связке** (*v-model*; код *web*-приложения + отображение в браузере (с учётом изменяющихся состояний) + схема ТП согласно нотации, определённой рабочим коллективом).

7. Содержательная часть по **методам и событиям** (*v-on*; код *web*-приложения + отображение в браузере (с учётом изменяющихся состояний) + схема ТП согласно нотации, определённой рабочим коллективом).

8. Содержательная часть по **условной отрисовке** (*v-if*; код *web*-приложения + отображение в браузере (с учётом изменяющихся состояний) + схема ТП согласно нотации, определённой рабочим коллективом).

9. Вывод.

В каждом разделе учесть ограничения, прописанные в жёлтых клетках таблиц.

**Реализовать:**

– локально (*LSPWA*), не прибегая к инструментарию *Node.js* и *npm* (*Node Package Manager*);

– в файле с именем *index01\_Familiya\_I\_O1\_Familiya\_I\_O2.html* параметрическую вставку;

– в файле с именем *index02\_Familiya\_I\_O1\_Familiya\_I\_O2.html* параметрическую связку;

– в файле с именем *index03\_Familiya\_I\_O1\_Familiya\_I\_O2.html* работу с методами и событиями;

– в файле с именем *index04\_Familiya\_I\_O1\_Familiya\_I\_O2.html* условную отрисовку.

**Комплектность отчёта:**

– *Word*-документ согласно «Базовой структуре отчёта»;

– перечень *web*-страниц и приложений к ним согласно разделу «Реализовать».

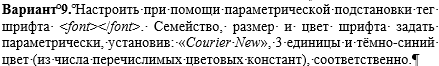


Рисунок 1 - Вариант задания с параметрической вставкой

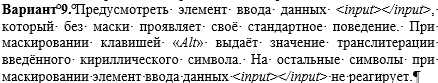


Рисунок 2 - Вариант задания с параметрической связкой

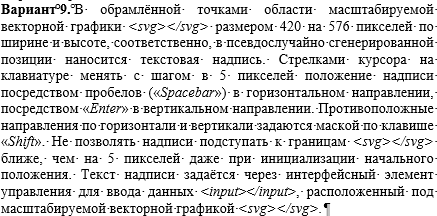


Рисунок 3 - Вариант задания с работой с методами и событиями

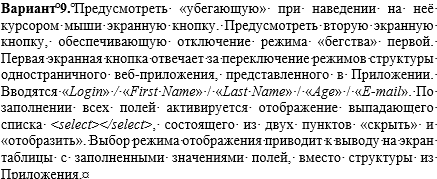


Рисунок 4 - Вариант задания с условной отрисовкой

# **3. Технология локального подключения фреймворка Vue.js**

Подключение Vue.js локально (LSPWA) выполняется следующим образом:

1. Перейти по ссылке <https://unpkg.com/vue@3/dist/vue.global.js> (или на любой другой CDN
2. Скачать открывшийся файл в папку проекта
3. В теге script в параметре src указывать вместо ссылки на веб-страницу с скриптом путь к локальному файлу:

<script src="./vue.global.js"></script>

**4. Содержательная часть по параметрической вставке**

Код веб-приложения:

<!DOCTYPE html>

<html>

    <head>

    </head>

    <body>

        <script src="./vue.global.js"></script>

        <div id="app">

            <p>

                <font   :face="fnt\_face"

                        :size="fnt\_size"

                        :color="fnt\_color">

                    {{ message }}

                </font>

            </p>

        </div>

        <script>

            const {createApp} = Vue

            createApp({

                setup() {

                    const message = "Hello world!"

                    const fnt\_face = "Courier New"

                    const fnt\_size = "3"

                    const fnt\_color = "navy"

                    return {

                        message,

                        fnt\_face,

                        fnt\_size,

                        fnt\_color

                    }

                }

            }).mount('#app')

        </script>

    </body>

</html>

Отображение в веб-браузере:

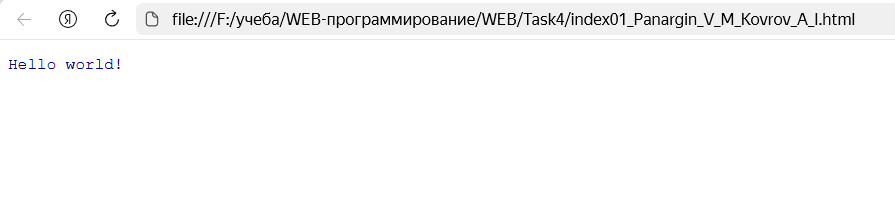


Рисунок 5 – Отображение страницы в браузере

Схема ТП согласно установленной нотации:

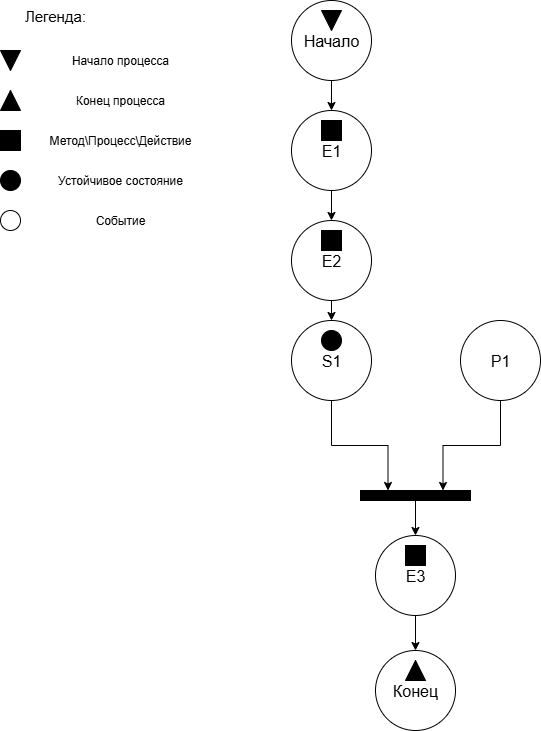


Рисунок 6 - Схема Петри задачи с параметрической вставкой

Описание сети Петри:

- состояния (states)

s1 – форма ожидает действий пользователя

- действия (effects)

e1 – подгрузка файла vue

e2 – вставка параметров

e3 – закрытие страницы

- события (prompts)

p1 – пользователь закрыл страницу

**5. Содержательная часть по методам и событиям**

Код веб-приложения:

<!DOCTYPE html>

<html>

    <head>

    </head>

    <body>

        <script src="./vue.global.js"></script>

        <div id="app">

            <input  @keydown.alt = "keydown\_event\_handler"

                    >

        </div>

        <script>

            const {createApp} = Vue

            createApp({

                setup() {

                    const transliterationMap = {

                        'а': 'a', 'б': 'b', 'в': 'v', 'г': 'g', 'д': 'd', 'е': 'e', 'ё': 'yo',

                        'ж': 'zh', 'з': 'z', 'и': 'i', 'й': 'y', 'к': 'k', 'л': 'l', 'м': 'm',

                        'н': 'n', 'о': 'o', 'п': 'p', 'р': 'r', 'с': 's', 'т': 't', 'у': 'u',

                        'ф': 'f', 'х': 'kh', 'ц': 'ts', 'ч': 'ch', 'ш': 'sh', 'щ': 'shch',

                        'ъ': '', 'ы': 'y', 'ь': '', 'э': 'e', 'ю': 'yu', 'я': 'ya',

                        'А': 'A', 'Б': 'B', 'В': 'V', 'Г': 'G', 'Д': 'D', 'Е': 'E', 'Ё': 'Yo',

                        'Ж': 'Zh', 'З': 'Z', 'И': 'I', 'Й': 'Y', 'К': 'K', 'Л': 'L', 'М': 'M',

                        'Н': 'N', 'О': 'O', 'П': 'P', 'Р': 'R', 'С': 'S', 'Т': 'T', 'У': 'U',

                        'Ф': 'F', 'Х': 'Kh', 'Ц': 'Ts', 'Ч': 'Ch', 'Ш': 'Sh', 'Щ': 'Shch',

                        'Ъ': '', 'Ы': 'Y', 'Ь': '', 'Э': 'E', 'Ю': 'Yu', 'Я': 'Ya'

                    };

                    function keydown\_event\_handler(event) {

                        console.log(event)

                        input = event.target

                        key = event.key

                        if (/[а-яёА-ЯЁ]/.test(key)) {

                            input.value += transliterationMap[key]

                        }

                        else {

                            event.preventDefault()

                        }

                    }

                    function keyup\_event\_handler(event) {

                        console.log(event)

                    }

                    function input\_handler(event) {

                        console.log(event)

                        event.preventDefault()

                    }

                    return {

                        keydown\_event\_handler,

                        keyup\_event\_handler,

                        input\_handler

                    }

                }

            }).mount('#app')

        </script>

    </body>

</html>

Отображение в браузере:

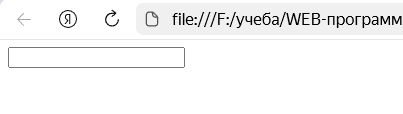


Рисунок 7 - Изначальное состояние страницы



Рисунок 8 - Введен текст без зажатия кнопки Alt

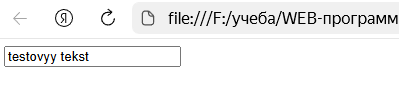


Рисунок 9 - Введен аналогичный текст с зажатой кнопкой Alt

Схема ТП согласно установленной нотации:

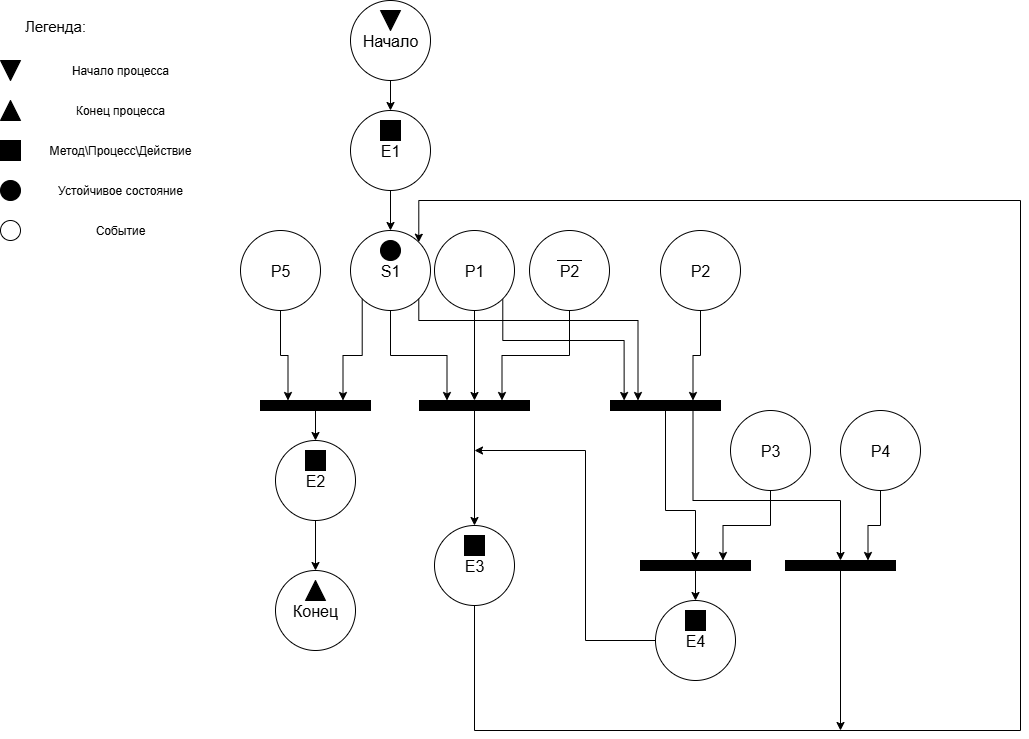
****

Рисунок 10 - Схема Петри задачи по методам и событиям

Описание сети Петри:

- состояния (states)

s1 – форма ожидает действий пользователя

- действия (effects)

e1 – подгрузка файла vue

e2 – закрытие страницы

e3 – добавление символа в input

e4 – транслитерация символа

- события (prompts)

p1 – пользователь вводит символы в input

p2 – зажат Alt

p3 – введен кириллический символ

p4 – введен не кириллический символ

p5 – пользователь закрыл страницу

**6. Содержательная часть по параметрической связке**

Код веб-приложения:

<!DOCTYPE html>

<html>

    <head>

    </head>

    <body>

        <script src="./vue.global.js"></script>

        <div id="app">

             <svg width="420" height="576" xmlns="http://www.w3.org/2000/svg">

                <rect width="100%" height="100%" fill="#f0f8ff" stroke="black" stroke-width="2" stroke-dasharray="2,2"/>

                <text

                    :x="textPosition.x"

                    :y="textPosition.y"

                    text-anchor="middle"

                    dominant-baseline="middle"

                    font-size="20"

                    fill="#333"

                >{{ textContent }}</text>

            </svg>

            <br>

            <input

                type="text"

                v-model="textContent"

                placeholder="Введите текст для отображения в SVG"

            >

            <p>Текущая позиция: X={{ textPosition.x }}, Y={{ textPosition.y }}</p>

        </div>

        <script>

            const {createApp, ref, onMounted, onUnmounted} = Vue

            createApp({

                setup() {

                    const textContent = ref('Пример текста')

                    const textPosition = ref({ x: 210, y: 288 })

                    var shiftPressed = false

                    const generateRandomPosition = () => {

                        const minX = 5

                        const maxX = 415

                        const minY = 5

                        const maxY = 571

                        return {

                            x: Math.floor(Math.random() \* (maxX - minX + 1)) + minX,

                            y: Math.floor(Math.random() \* (maxY - minY + 1)) + minY

                        };

                    };

                    textPosition.value = generateRandomPosition()

                    const handleKeyDown = (event) => {

                        if (event.key === 'Shift') {

                            shiftPressed = true

                            return

                        }

                        console.log(shiftPressed)

                        const direction = shiftPressed ? -1 : 1

                        const step = 5

                        if (event.key === ' ') {

                            event.preventDefault()

                            const newX = textPosition.value.x + (step \* direction)

                            if (newX >= 5 && newX <= 415) {

                                textPosition.value.x = newX

                            }

                        } else if (event.key === 'Enter') {

                            event.preventDefault()

                            const newY = textPosition.value.y + (step \* direction)

                            if (newY >= 5 && newY <= 571) {

                                textPosition.value.y = newY

                            }

                        }

                    };

                    function handleKeyUp(event) {

                        if (event.key === 'Shift') {

                            shiftPressed = false;

                        }

                    }

                    onMounted(() => {

                        window.addEventListener('keydown', handleKeyDown);

                        window.addEventListener('keyup', handleKeyUp);

                    });

                    // Удаление обработчиков событий

                    onUnmounted(() => {

                        window.removeEventListener('keydown', handleKeyDown);

                        window.removeEventListener('keyup', handleKeyUp);

                    });

                    return {

                        textContent,

                        textPosition

                    }

                }

            }).mount('#app')

        </script>

    </body>

</html>

Отображение в браузере:

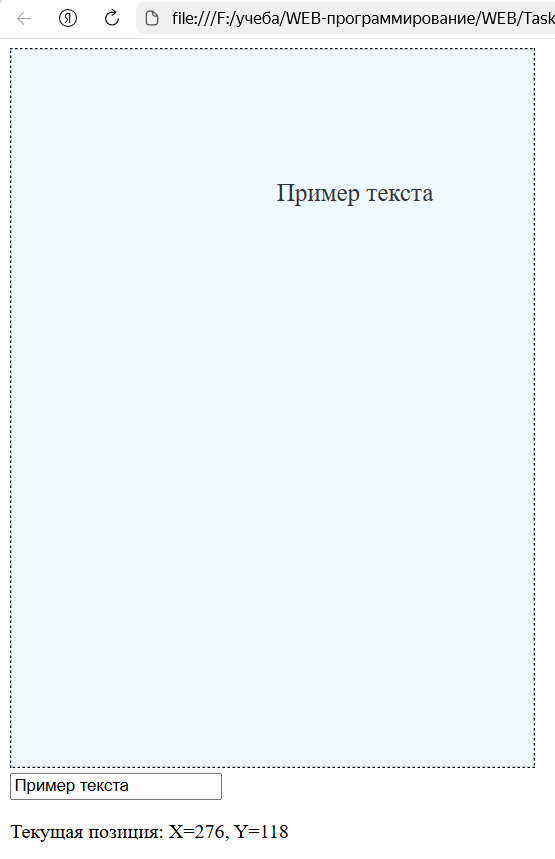


Рисунок 11 - Отображение страницы в браузере

Схема ТП согласно установленной нотации:

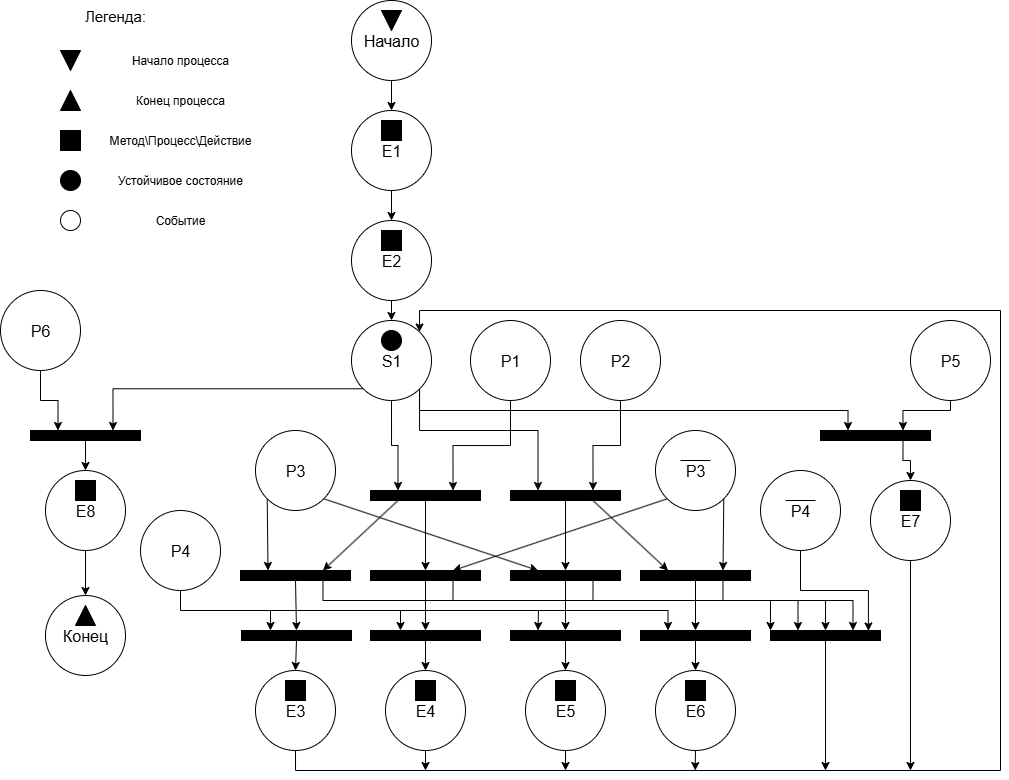
****

Рисунок 12 - Схема по параметрической связке

Описание сети Петри:

- состояния (states)

s1 – форма ожидает действий пользователя

- действия (effects)

e1 – подгрузка файла vue

e2 – вставка параметров

e3 – перемещение текста влево

e4 – перемещение текста вправо

e5 – перемещение текста вниз

e6 – перемещение текста вверх

e7 – изменение параметра текста

e8 – закрытие страницы

- события (prompts)

p1 – пользователь нажимает кнопку Space

p2 – пользователь нажимает кнопку Enter

p3 – зажата клавиша Shift

p4 – смещение вписывается в рамки поля svg

p5 – пользователь изменил содержание input

p6 – пользователь закрыл страницу

**7. Содержательная часть по условной отрисовке**

Код веб-приложения

<!DOCTYPE html>

<html>

    <head>

    </head>

    <body>

        <script src="./vue.global.js"></script>

        <div id="app">

            <div v-if="formState === 'input'">

                <table width="300px">

                    <tr>

                        <td><p>{{ fields[fieldIndex] }}: </p></td><td><input v-model="inputData"></td>

                    </tr>

                </table>

                <button @click="nextField" @mouseover="moveButton" :style="escapeMode ? escapeButtonStyle : ''">Переключение</button>

                <button @click="stopRun">Отключить убегание</button>

            </div>

            <div v-if="formState === 'select'">

                <select @change="handleSelect">

                    <option>cкрыть</option>

                    <option>отобразить</option>

                </select>

            </div>

            <div v-if="formState === 'table'">

                <table>

                    <tr>

                        <td>Логин</td>

                        <td>{{ formData[0] }}</td>

                    </tr>

                    <tr>

                        <td>Имя</td>

                        <td>{{ formData[1] }}</td>

                    </tr>

                    <tr>

                        <td>Фамилия</td>

                        <td>{{ formData[2] }}</td>

                    </tr>

                    <tr>

                        <td>Возраст</td>

                        <td>{{ formData[3] }}</td>

                    </tr>

                    <tr>

                        <td>E-mail</td>

                        <td>{{ formData[4] }}</td>

                    </tr>

                </table>

            </div>

        </div>

        <script>

            const {createApp, ref, onMounted, onUnmounted} = Vue

            createApp({

                setup() {

                    formState = ref("input")

                    fields = ["Логин", "Имя", "Фамилия", "Возраст", "Электронная почта"]

                    const fieldIndex = ref(0)

                    escapeMode = true

                    const escapeButtonStyle = ref({

                        position: 'absolute',

                        left: '0px',

                        top: '0px'

                    });

                    formData = ref(['', '', '', '', ''])

                    inputData = ref('')

                    function stopRun() {

                        escapeMode=false

                        escapeButtonStyle.value = {

                            position: 'absolute',

                            left: "0px",

                            top: "0px"

                        };

                    }

                    function nextField() {

                        console.log(fields.length)

                        formData.value[fieldIndex.value] = inputData.value

                        if (fieldIndex.value < fields.length - 1) {

                            fieldIndex.value = fieldIndex.value + 1

                        }

                        else {

                            formState.value="select"

                        }

                    }

                    const moveButton = (event) => {

                        if (!escapeMode) return;

                        const button = event.target;

                        const container = button.parentElement;

                        const containerRect = container.getBoundingClientRect();

                        const buttonRect = button.getBoundingClientRect();

                        // Случайное смещение

                        const maxX = containerRect.width - buttonRect.width - 10;

                        const maxY = containerRect.height - buttonRect.height - 10;

                        const newX = Math.floor(Math.random() \* maxX);

                        const newY = Math.floor(Math.random() \* maxY);

                        escapeButtonStyle.value = {

                            position: 'absolute',

                            left: `${newX}px`,

                            top: `${newY}px`

                        };

                    };

                    function handleSelect(event) {

                        formState.value = "table"

                    }

                    return {

                        formState,

                        fields, fieldIndex,

                        formData,

                        inputData,

                        escapeMode,

                        escapeButtonStyle,

                        stopRun,

                        moveButton,

                        nextField,

                        handleSelect

                    }

                }

            }).mount('#app')

        </script>

    </body>

</html>

Отображение в браузере:

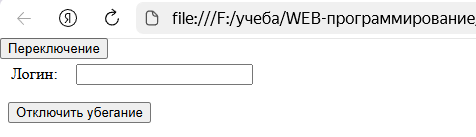


Рисунок 13 - Изначальный вид страницы



Рисунок 14 - "Убегание" кнопки



Рисунок 15 - Элемент select

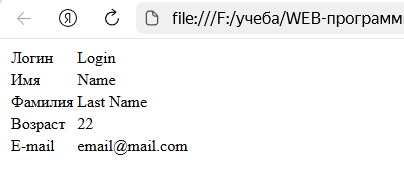


Рисунок 16 - Таблица после заполнения данных

Схема ТП согласно установленной нотации:

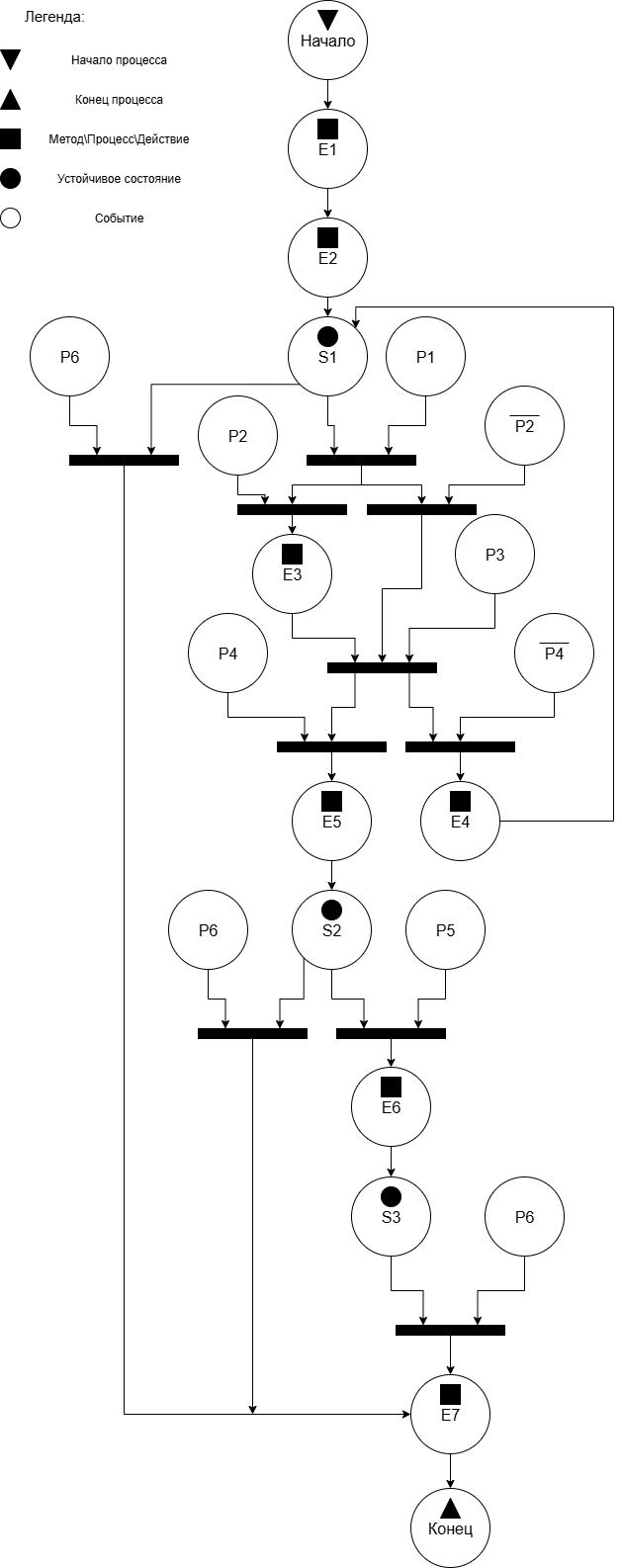


Рисунок 17 - Схема по условной отрисовке

Описание сети Петри:

- состояния (states)

s1 – форма ожидает действий пользователя

s2 – форма ожидает изменения select

s3 – демонстрируется таблица с введенными данными

- действия (effects)

e1 – подгрузка файла vue

e2 – вставка параметров

e3 – перемещение кнопки переключения

e4 – переход к следующему полю ввода данных

e5 – переход к странице с select

e6 – переход к странице с таблицей

e7 – закрытие страницы

- события (prompts)

p1 – пользователь наводит курсор на кнопку переключения

p2 – кнопка остановки убегания нажималась

p3 – пользователь нажимает кнопку переключения

p4 – введены все необходимые данные

p5 – пользователь выбрал отобразить в select

p6 – пользователь закрыл страницу

# **7. Формулировку вывода о проделанной работе**

В результате работы были успешно изучены и применены на практике ключевые директивы Vue.js, что подтвердило их эффективность для создания динамических веб-интерфейсов.