**Техническая документация**

**Органайзер для расписания занятий стундентов**

**Версия документации:** 0.1  
**Дата:** 17 января 2025 года  
**Авторы:** Корчуганов Максим, Костюкевич Кирилл, Чикида Данила, Мицкевич Кирилл.

**1. Введение**

**Цель документации**

Данная документация описывает функциональность приложения для управления расписанием. Приложение позволяет добавлять, редактировать и удалять занятия для каждого дня недели, а также сортировать занятия по порядку. Система поддерживает работу с двумя неделями — первой и второй, позволяя пользователю легко переключаться между ними и добавлять занятия в соответствующие дни недели.

**Общее описание приложения**

Приложение разработано с использованием **Electron**, который предоставляет кроссплатформенную среду для настольных приложений с использованием **Node.js**. С помощью **Node.js** реализована серверная логика для обработки данных расписания, взаимодействия с файловой системой и управления процессом выполнения приложения.

Приложение состоит из следующих частей:

**Frontend** - интерфейс, написанный с использованием HTML, CSS и JavaScript.

**Backend** - логика приложения, работающая с данными расписания, обрабатывающая действия пользователя (например, добавление занятия).

----------------------

**2. Описание системы**

**Основные функции приложения:**

**Добавление занятия.**  
Пользователь может добавить новое занятие, указав его название, день недели, порядок занятия и номер недели.

**Редактирование расписания.**  
Занятия могут быть добавлены только в определенные дни недели и недели (первая или вторая неделя).

**Удаление занятия.**  
Пользователь может удалить любое занятие, которое было добавлено в расписание. Удаление осуществляется нажатием кнопки с изображением корзины рядом с занятием.

**Отображение расписания по неделям.**  
Расписание для каждой недели (I и II) отображается на главной странице. Каждому дню недели соответствует отдельный блок с занятиями.

**Сортировка занятий.**  
Все занятия сортируются по порядку, который указывается пользователем

при добавлении занятия (например, номер пары). Это позволяет поддерживать четкую структуру и организованность расписания.

**Технологии и инструменты:**

**Electron** - для создания кроссплатформенных настольных приложений.

**Node.js** - для обработки данных, выполнения серверной логики и работы с файловой системой.

**HTML, CSS, JavaScript** - для разработки интерфейса.

**Bootstrap** - для создания адаптивного и удобного интерфейса.

**Bootstrap Icons** - для использования иконок в интерфейсе (например, для кнопки удаления).

**Роль Node.js в приложении:**

1. **Серверная логика.**  
   **Node.js** используется для создания основной логики работы приложения, включая:
   * Сохранение и чтение данных расписания (в будущем можно расширить, добавив сохранение в файлы или базу данных).
   * Взаимодействие с внешними API или файловой системой.
   * Обработка событий и выполнение операций, связанных с данными расписания.
2. **Интеграция с Electron.**  
   Взаимодействие с интерфейсом и рендерингом веб-страниц происходит через **Electron**, который использует **Node.js** для выполнения серверной части внутри кроссплатформенного окна приложения. Это дает возможность использовать весь потенциал Node.js, включая доступ к файловой системе и мощные возможности асинхронного выполнения.
3. **Подключение модулей Node.js.**  
   Приложение может подключать и использовать различные модули Node.js для выполнения специфических задач. Например, модули для работы с файловой системой, отправки HTTP-запросов или других сторонних библиотек.

----------------------

**3. Архитектура и компоненты**

Приложение состоит из нескольких ключевых компонентов, каждый из которых отвечает за выполнение определенных функций:

1. **Главный процесс Electron (main.js)**

Этот процесс отвечает за создание и управление главным окном приложения, а также за обработку событий жизненного цикла приложения. В рамках этого процесса выполняются все операции с данными и взаимодействие с файловой системой через **Node.js**.

**Применение Node.js:**  
В главном процессе используется **Node.js** для:

Управления состоянием приложения.

Чтения/записи данных, работы с внешними API или локальной файловой системой.

Обработки пользовательских событий (например, добавление, удаление или изменение занятий).

2. **Интерфейс пользователя (HTML, CSS и JavaScript)**

HTML-файл (index.html) описывает структуру страницы, включая блоки для отображения расписания. CSS (включая библиотеки Bootstrap) используется для стилизации страницы, чтобы она была адаптивной и удобной для использования. JavaScript содержит логику для взаимодействия с пользователем, обработки данных и управления интерфейсом.

3. **Логика работы с расписанием (main.js)**

Взаимодействие с данными расписания происходит с помощью объекта scheduleData, который содержит расписания для двух недель и для каждого дня недели. Каждый день недели может содержать несколько занятий, которые сортируются по порядку.

4. **Модальное окно для добавления занятия**

Когда пользователь нажимает кнопку "Добавить расписание", открывается модальное окно, в котором он может ввести информацию о занятии. Это окно собирает данные с формы и передает их в основной процесс, который обновляет расписание.

----------------------

**4. Пример использования**

**Добавление нового занятия:**

Нажмите на кнопку **"Добавить расписание"** в нижней части экрана.

В открывшемся модальном окне выберите:

* + **Название занятия.** Укажите название предмета или занятия.
  + **День недели.** Выберите день недели, в который будет проходить занятие (например, Понедельник).
  + **Порядок занятия.** Укажите порядок занятия (например, номер пары).
  + **Номер недели.** Выберите, для какой недели (1-я или 2-я) добавляется занятие.

После ввода данных нажмите кнопку **"Добавить"**, чтобы сохранить изменения.

**Удаление занятия:**

Перейдите в соответствующий день недели (например, Понедельник).

Найдите занятие, которое нужно удалить.

Нажмите на кнопку **с иконкой корзины** рядом с занятием. Занятие будет удалено из расписания.

**Просмотр расписания:**

В главном окне приложения отображаются карточки для каждого дня недели. Каждая карточка содержит занятия, добавленные для этого дня, отсортированные по порядку.

----------------------

**5. Ошибки и решение проблем**

**Ошибка 1: Модальное окно не появляется при нажатии на кнопку "Добавить расписание".**

Проверьте, что правильно подключены все скрипты и стили. Убедитесь, что файл main.js загружается и не содержит ошибок.

Убедитесь, что библиотека **Bootstrap** подключена и доступна для

работы с модальными окнами.

**Ошибка 2: Расписание не обновляется после добавления нового занятия.**

Убедитесь, что объект scheduleData правильно обновляется, и функция рендеринга (renderSchedule) вызывается после добавления нового занятия.

Проверьте консоль на наличие ошибок в JavaScript.

**Ошибка 3: Кнопка удаления не работает.**

Проверьте, что в коде правильно привязана функция для удаления занятия (событие клика на кнопку). Убедитесь, что после удаления вызывается функция рендеринга расписания, чтобы интерфейс обновился.

----------------------

**6. Приложения**

**Приложение A: Структура данных расписания**

**Пример структуры данных для двух недель:**

|  |
| --- |
| const scheduleData = {  1: { // Первая неделя  monday: [],  tuesday: [],  wednesday: [],  thursday: [],  friday: [],  saturday: [],  },  2: { // Вторая неделя  monday: [],  tuesday: [],  wednesday: [],  thursday: [],  friday: [],  saturday: [],  },  }; |

**Приложение B: Код для добавления нового занятия через модальное окно**

|  |
| --- |
| document.getElementById('add-schedule').addEventListener('click', () => {  createModalDialog('Добавить занятие', [  { label: 'Название занятия', name: 'subject', required: true },  {  label: 'День недели', name: 'day', type: 'select', required: true, options: [  { value: 'monday', label: 'Понедельник' },  { value: 'tuesday', label: 'Вторник' },  { value: 'wednesday', label: 'Среда' },  { value: 'thursday', label: 'Четверг' },  { value: 'friday', label: 'Пятница' },  { value: 'saturday', label: 'Суббота' },  ]  },  { label: 'Порядок занятия (например, номер пары)', name: 'order', required: true, type: 'number' },  {  label: 'Номер недели', name: 'week', required: true, type: 'select', options: [  { value: '1', label: 'Первая неделя' },  { value: '2', label: 'Вторая неделя' },  ]  }  ], ({ subject, notes, day, order, week }) => {  If (scheduleData[week][day]) {  scheduleData[week][day].push({ subject, notes, order });  scheduleData[week][day].sort((a, b) => a.order - b.order); // Сортируем занятия по порядку  renderSchedule();  }  });  }); |

----------------------

**7. Заключение**

Приложение представляет собой удобный инструмент для управления расписанием, позволяя пользователям легко добавлять, редактировать и удалять занятия. Использование **Node.js** дает возможность более эффективно управлять данными, а также выполнять асинхронные операции и взаимодействие с файловой системой.