**Техническая документация приложения "Student Organizer"**

**Версия:** 0.2  
**Дата создания:** 17 января 2025 года  
**Авторы:** Корчуганов Максим, Костюкевич Кирилл, Чикида Данила, Мицкевич Кирилл

**1. Общие положения**

**1.1 Назначение приложения**

Приложение **"Student Organizer"** предназначено для управления расписанием занятий студентов. Оно позволяет:

Создавать и редактировать расписания для двух недель (I и II).

Организовывать занятия по дням недели и порядку.

Обеспечивать удобный интерфейс для просмотра, редактирования и удаления данных.

**1.2 Область применения**

Программа может быть использована студентами, преподавателями и администрацией учебных заведений для упрощения планирования учебного процесса.

**2. Функциональные возможности**

**Добавление занятий:**  
Пользователь вводит название занятия, день недели, номер недели (I или II) и порядок занятия (например, номер пары).

**Редактирование расписания:**  
Возможность обновлять данные существующего расписания.

**Удаление занятий:**  
Удаление добавленных занятий через интерфейс приложения.

**Просмотр расписания:**  
Расписание отображается в формате карточек с разделением по неделям и дням недели.

**Сортировка занятий:**  
Упорядочивание занятий по порядковому номеру, указанному пользователем.

**Управление двумя неделями:**  
Возможность переключения между неделями для редактирования и просмотра данных.

**3. Технические характеристики**

**3.1 Архитектура приложения**

Приложение построено на основе технологии **Electron** и разделено на следующие компоненты:

1. **Главный процесс (Electron):**
   * Управляет созданием окон приложения.
   * Взаимодействует с файловой системой и внешними API через Node.js.
2. **Интерфейс пользователя (Frontend):**
   * Разработан с использованием HTML, CSS и JavaScript.
   * Использует библиотеку Bootstrap для стилизации.
3. **Серверная логика (Backend):**
   * Обработка пользовательских действий.
   * Управление данными расписания с использованием объектов и массивов.
   * Работа с модулями Node.js.

**3.2 Основные технологии**

**Electron** — создание настольного приложения.

**Node.js** — обработка данных и файловая логика.

**Bootstrap** — создание адаптивного пользовательского интерфейса.

**Bootstrap Icons** — использование графических элементов.

**3.3 Требования к системе**

* **Аппаратное обеспечение:**
  + Процессор: двухъядерный, 2 ГГц.
  + Оперативная память: 2 ГБ и более.
  + Свободное место на диске: 100 МБ.
* **Программное обеспечение:**
  + Node.js версии 14.0 или выше.
  + Electron версии 18.0 или выше.
  + Операционная система: Windows, macOS, Linux.

**4. Установка и запуск**

**4.1 Установка приложения**

Скачайте архив с исходным кодом приложения или установочный файл.

Распакуйте архив в желаемую директорию.

Убедитесь, что на вашем компьютере установлен Node.js и npm.

**4.2 Запуск приложения**

Перейдите в папку с проектом.

Выполните команду для установки зависимостей:

npm install

Запустите приложение:

npm start

**5. Пользовательский интерфейс**

**5.1 Главный экран**

На главном экране отображаются:

Карточки для каждого дня недели.

Переключатели между I и II неделями.

Кнопка для добавления нового занятия.

**5.2 Модальное окно добавления/редактирования занятия**

Окно содержит:

Поле для ввода названия занятия.

Списки для выбора дня недели и недели (I или II).

Поле для указания номера пары.

**5.3 Отображение расписания**

Занятия сортируются по дням недели и порядковому номеру.

**6. Пример использования**

**6.1 Добавление занятия**

1. Нажмите кнопку **"Добавить расписание"**.

2. Введите данные:

* + Название занятия.
  + День недели.
  + Номер недели.
  + Порядок занятия.

3. Нажмите кнопку **"Добавить"**.

**6.2 Удаление занятия**

Найдите нужное занятие на карточке соответствующего дня недели.

Нажмите на кнопку **"Корзина"** рядом с занятием.

**7. Взаимодействие компонентов**

**Frontend:** Отправляет запросы на Backend через события и обработчики.

**Backend:** Получает данные от пользователя, обрабатывает их и обновляет структуру расписания.

**База данных:** Расписание хранится в памяти приложения (или в JSON-файле).

**8. Приложения**

**Приложение A. Пример структуры данных расписания**

|  |
| --- |
| 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 занятия**

|  |
| --- |
| 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();  }  });  }); |

**8. Заключение**

Приложение **"Student Organizer"** предоставляет интуитивно понятный инструмент для управления расписанием. Современные технологии позволяют обеспечить высокую производительность, кроссплатформенность и удобство для пользователей.