Schedule2calendar - Документация

Владислав Полетаев

May 9, 2017

Schedule2calendar - Документация
by Владислав Полетаев

Contents

List of Figures

Abstract

Данный документ предназначен для использования разработчиками программного обеспечения. Документ содержит общие сведения о программном продукте, технические характеристики: структуру, описание компонентов, примеры использования определнных в проекте публичных интерфейсов.

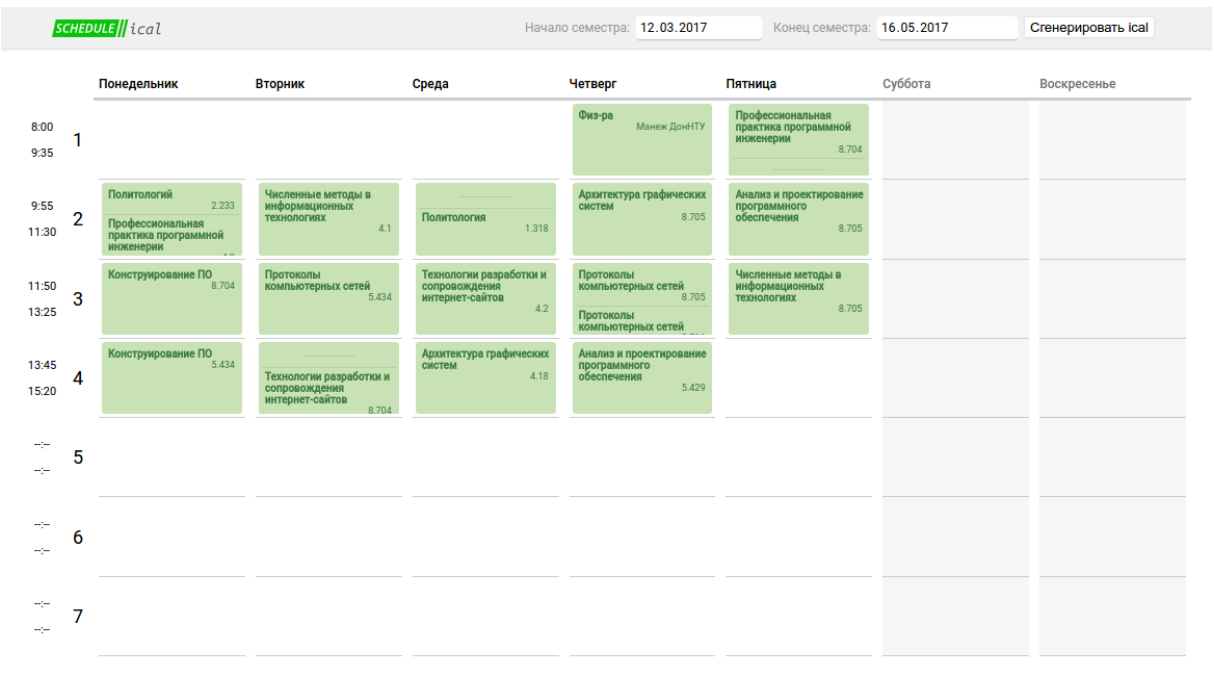
Chapter 1

Описание программного продукта

Проект schedule2calendar предназначен для упрощения создания файлов iCalendar при добавлении расписаний занятий учащимися средних и высших учебных учереждений.

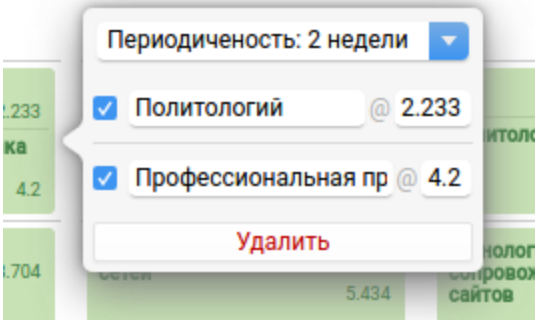
На снимке экрана приведен внешний вид приложения на момент написания настоящего документа (Figure 1.1).

Figure 1.1 Снимок работы программы



Редактирование данных производится с использованием элемента управления Popover (Figure 1.2).

Figure 1.2 Редактор данных приложения



Приложение реализует графический интерфейс, который позволяет пользователю:

- редактировать время начала и конца до десяти занятий;
- добалять удалять проводимые занятия для каждого дня недели;
- для каждого занятия выбрать количество чередующихся вариантов занятий: одинаковое занятие каждую неделю, два чередующегося варианта занятия или четыре варианта занятия;
- для каждого варианта занятия ввести название и место проведения занятия;
- деактивировать один или несколько вариантов занятия, в случае если оно проводится не каждую неделю;
- по сформированному расписанию генерировать файл в формате .ics, по стандарту, определенному в RFC 5545

Chapter 2

Внешние зависимости

2.1 npm

Для управления зависимостями проекта используется пакетный менеджер npm.

Все зависимости указаны в файле `package.json`. Для установки всех зависимостей необходимо выполнить:

```
npm install
```

2.2 Grunt

Для построения проекта в форму, доступную для использования в production используется система автоматизированной сборки Grunt. Конфигурация Grunt содержится в файле `Gruntfile.js`

Файлы, сформированные в результате сборки помещаются в директорию `build`.

В проекте определены 6 задач:

2.2.1 less

Преобразует файлы .less в файлы .css. В результате работы задачи
формируется файл app.css и elements.css. На вход подаются файлы директоии `src/less`.

2.2.2 browserify

Преобразует JavaScript файлы, написанные с использованием EcmaScript 6 с использованием диалекта JSX к виду, поддерживаемому большинством браузеров.

2.2.3 copy

Копирует файл `index.html` из директории `src` в директорию `build`

2.2.4 watch

Выполняет задачи copy, less или browserify при изменении определенных исходных файлов

2.2.5 default

Выполняет задачи copy, less и browserify.

2.2.6 dev

Выполняет задачи default и watch.

2.3 Зависимости этапа компиляции

- *assemble-less*: используется для сборки less-файлов
- *grunt-contrib-watch*: используется для отслеживания изменений файлов
- *grunt-browserify*: используется для управления сборкой JS-файлов
- *grunt-copy*: используется для копирования файлов
- *babelify*: используется для сборки JS файлов
- *babel-preset-es2015*: используется для преобразования стандарта EcmaScript 2015
- *babel-preset-react*: используется для преобразования JSX-файлов

2.4 moment

Библиотека для манипуляции датами. Использоуется во время генерации файла .ics

2.5 ical-generator

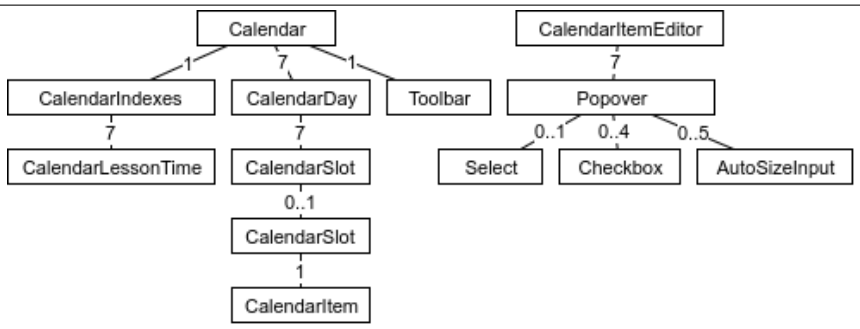
Библиотека для формирования файла iCalendar.

Chapter 3

Структура приложения

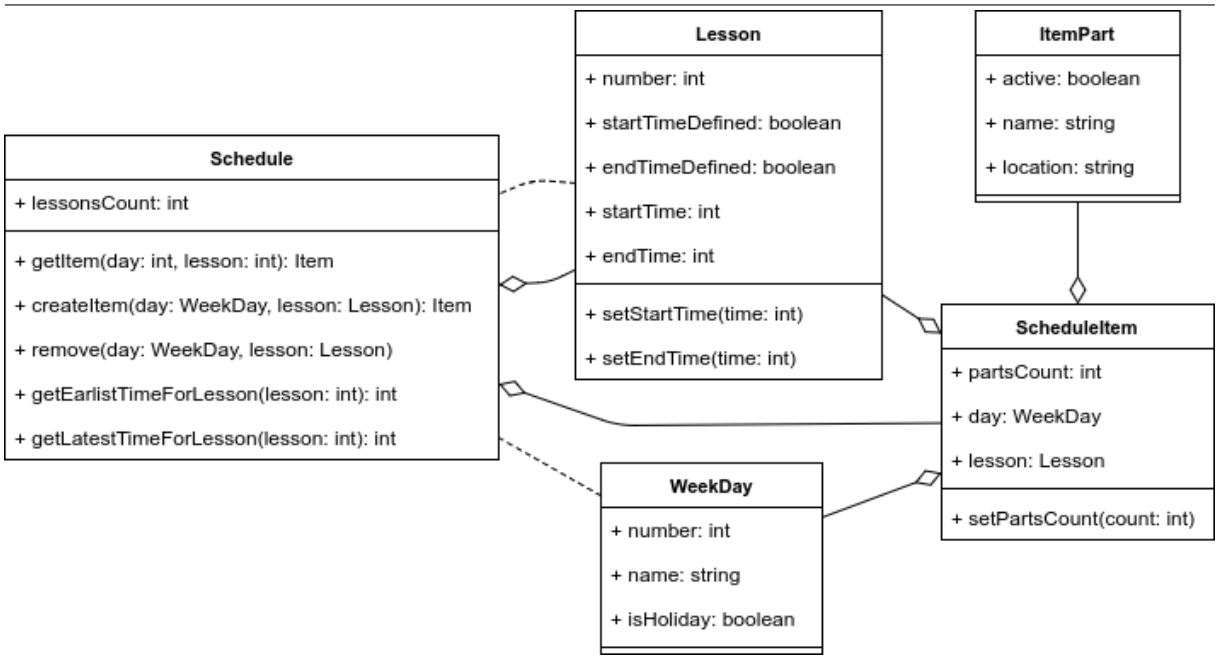
Приложение построено в виде иерархии React-компонентов (Figure 3.1).

Figure 3.1 Иерархия компонентов



Для работы компонентов используется модель данных, описанная на диаграмме (Figure 3.2).

Figure 3.2 Классы модели данных



Chapter 4

Примеры использования публичного интерфейса модулей системы

Example 4.1 Создание событий

В приведенном ниже коде представлен пример создания нового календаря и добавления нового периодического события.

```
let weekDays = require("./weekdays.js");
let schedule = require("./schedule.js");

let scheduleObject = new Schedule();
let weekDay = weekDays[2]; // Среда
let lesson = scheduleObject.lessons[1]; // Второе занятие
let item = scheduleObject.createItem(weekDay, lesson);
item.setPartsCount(2);
item.parts[0].name = "Занятие 1";
item.parts[0].location = "Аудитория 1";
item.parts[1].name = "Занятие 2";
item.parts[1].location = "Аудитория 2";
```

Example 4.2 Изменение веремени проведения занятия

В приведенном ниже коде представлен пример модификации времени начала и завершения занятия.

```
let schedule = require("./schedule.js");

let scheduleObject = new Schedule();
let lesson = scheduleObject.lessons[1]; // Второе занятие
lesson.setStartTime(9 * 60 + 55); // 9 часов 55 минут
lesson.setEndTime(10 * 60 + 30); // 10 часов 3 минут
```

Example 4.3 Генерация файла календаря

Генерация календаря включает в себя вызов функции `generate` модуля `./generator.js`

```
let generate = require("./generator.js").generate;
let canGenerate = require("./generator.js").canGenerate;

let schedule = new Schedule();
let startDate = "2017-03-12";
let endDate = "2017-05-16";
...
// Проверка: для любого события, добавленного в календарь, должно быть задано ↩
    время начала и конца
if (!canGenerate(schedule)){
    alert("Необходимо ввести даты конца и начала занятий для всех событий");

// Проверка: в календарь должно быть добавлено хотя-бы одно событие
```

```
}else if (schedule.items.length == 0){
    alert(”Необходимо добавить хотя бы одно событие”);
}else{
    // Генерация. Созданный файл будет загружен на компьютер пользвателя
    generate(schedule, startDate, endState);
}
```