

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ
ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ им. А. И. ГЕРЦЕНА»



Основная профессиональная образовательная программа
Направление подготовки 09.03.01 Информатика и вычислительная техника
Направленность (профиль) «Технологии разработки программного обеспечения»
форма обучения – очная

Выпускная квалификационная работа

Разработка распределённой системы представления информации для
образовательной организации

Обучающегося 4 курса
Исайчева Данилы Олеговича

Научный руководитель:
Аксютин Павел Александрович

Санкт-Петербург
2023

СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	3
ГЛАВА 1. ПОДГОТОВКА К РАЗРАБОТКЕ.....	5
1.1 Анализ существующих решений.....	5
1.2 Выбор средств реализации.....	16
ГЛАВА 2. РАЗРАБОТКА.....	27
2.1 Жизненный цикл.....	27
2.2 Схема взаимодействия компонентов системы.....	28
2.3 Создание базы данных.....	28
2.4 Разработка сервера для взаимодействия с базой данных.....	30
2.5 Разработка сервера для предоставления пользовательский интерфейс.....	30
2.6 Разработка сервера для генерации файлов календарей.....	32
2.7 Разработка сервера для доставки файлов календарей.....	33
ВЫВОД.....	34
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ.....	35
ПРИЛОЖЕНИЕ 1.....	38
ПРИЛОЖЕНИЕ 2.....	39
ПРИЛОЖЕНИЕ 3.....	43

ВВЕДЕНИЕ

С вступлением человечества в информационную эру значительно увеличилось количество потребляемой каждым человеком информации. В связи с этим возросли требования к качеству её представления и скорости её доставки.

Некачественно представленная информация тяжело воспринимается, а иногда и просто игнорируется, что недопустимо, когда речь заходит об организационной информации.

В современных образовательных учреждениях представление организационной информации зачастую выполняется вручную. Это увеличивает нагрузку на исполнителя и увеличивает время, необходимое для представления информации.

В связи с этим, актуальной является задача автоматизации процессов представления организационной информации в образовательном учреждении.

Предметом работы является распределённая система представления информации для образовательной организации.

Целью выпускной квалификационной работы является разработка распределённой системы, которая позволит автоматизировать представление организационной информации в образовательном учреждении.

Для достижения поставленной цели необходимо решить ряд **задач**:

1. Проанализировать требования заказчика и существующие решения, на основе которых спроектировать модель базы данных.
2. Разработать модель распределённой системы.
3. Реализовать серверную и клиентскую составляющие веб-приложения.
4. Осуществить контейнеризацию приложения с помощью Docker.
5. Составить сопроводительную документацию: руководство для администратора и пользователей.
6. Провести внедрение и апробацию разработанной системы в образовательной организации.

Практическая значимость заключается в разработке распределённой системы, которая позволит автоматизировать процесс представления организационной информации в образовательном учреждении.

ГЛАВА 1. ПОДГОТОВКА К РАЗРАБОТКЕ

На момент начала выполнения проекта задача представления информации выполнялась по средствам программы подготовки презентаций Microsoft PowerPoint. Доставка информации осуществлялась с помощью:

- отправки изображений с расписанием в канал в Telegram;
- отправки изображений с расписанием в чат в Яндекс Мессенджере;
- отображения изображений расписанием на экранах телевизоров, расположенных на территории школы.

Основными недостатками существующего способа представления организационной информации являлись:

- высокая трудоёмкость составления расписания;
- трудности с оперативной доставкой информации о расписаниях и изменениях в них

В результате анализа потребностей заказчика, связанных с упрощением процесса составления и увеличением оперативности доставки организационной информации о расписании в образовательной организации было принято решение использовать электронные календари.

1.1 Анализ существующих решений

Существует множество решений, которые позволяют составлять электронные календари и делиться ими. Проанализируем такие решения для того, чтобы выявить основные преимущества и недостатки и определить целесообразность их использования в образовательной организации.

Яндекс Календарь

Яндекс Календарь – это веб-сервис от компании Яндекс, позволяющий создавать, обновлять, редактировать и отслеживать электронные календари.

Для определения преимуществ и недостатков данного сервиса проанализируем его со стороны двух категорий пользователей: тех, кто создаёт календари, и тех, кто просматривает их.

Для пользователя, отвечающего за создание календарей доступен интуитивно понятный интерфейс создания и редактирования календарей.

При создании календарей можно выбрать: название и оформление, установить: стандартные уведомления для событий, видимость событий в календаре, определить: влияние событий в календаре на занятость и является ли календарь основным.

В соответствии с рисунком 1, для оформления календаря доступен ограниченный набор цветов.

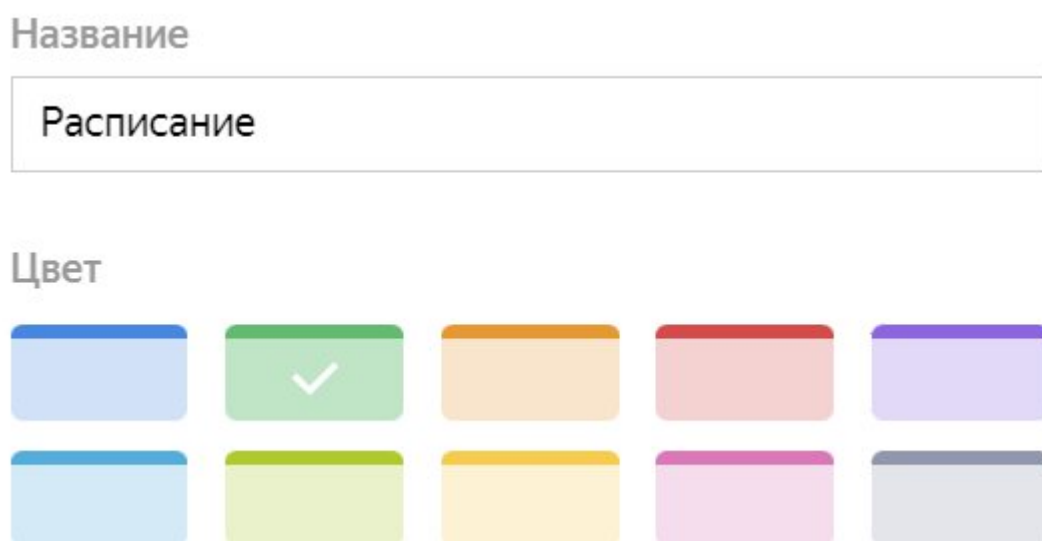


Рисунок 1 – Интерфейс выбора цвета календаря

В соответствии с рисунком 2, для уведомления пользователя о событии доступно три вида уведомлений: по почте, в СМС и через CalDav.

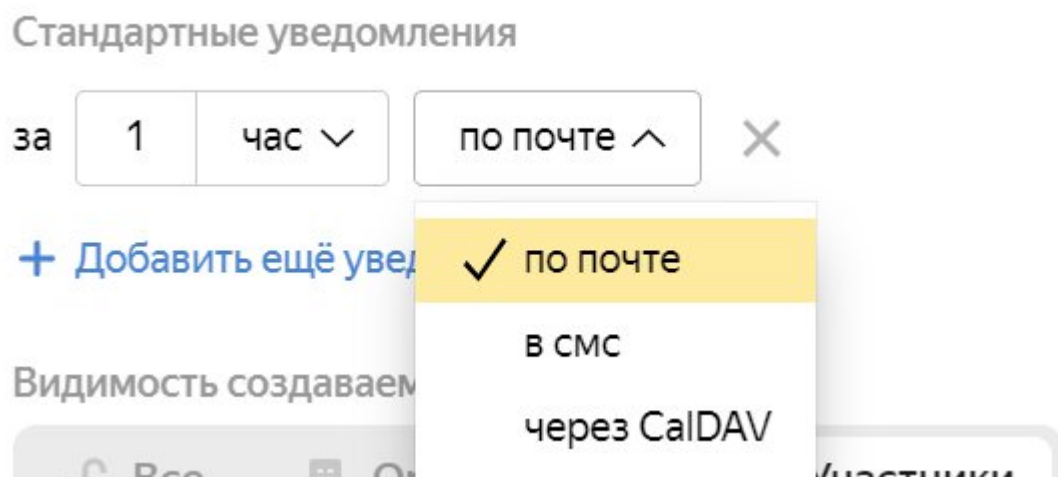


Рисунок 2 – Интерфейс настройки уведомления

Уведомление о событии приходит, как на рисунке 3 – в виде письма на электронную почту, или, как на рисунке 4 – в виде СМС сообщения на телефон, или, как на рисунке 5 – в виде уведомления.

Сегодня, 24-го апреля, с 13:25 до ... < Пред. След. >
13:55 (Europe/Moscow,
GMT+03:00), «Событие»



Яндекс.Календарь info@calendar.yandex.ru Сегодня в 13:23
Я >

**Напоминание: событие «[Событие](#)» состоится
сегодня, 24-го апреля, с 13:25 до 13:55
(Europe/Moscow, GMT+03:00)**

Описание Описание события

Рисунок 3 – Письмо с уведомлением о событии

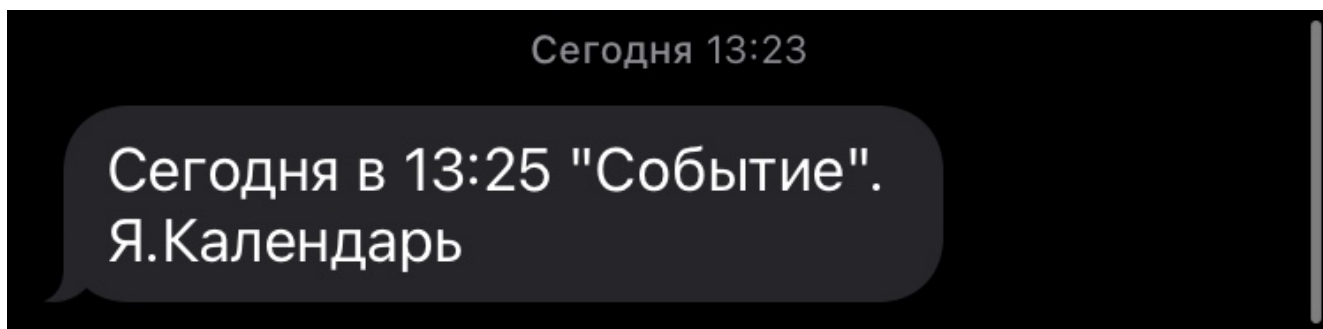


Рисунок 4 – СМС сообщение с уведомлением о событии



Рисунок 5 – Интерфейс уведомления о событии

Для видимости событий в календаре доступно три опции: «Все» – события в календаре видят все, «Организация» – события видят пользователи, входящие в одну организацию, «Участники» – события видят пользователи, которые были добавлены к ним, как участники.

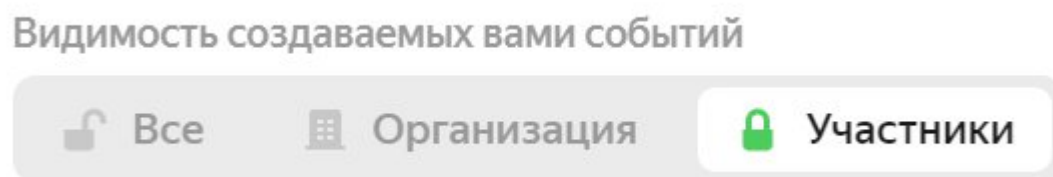


Рисунок 6 – Интерфейс настройки видимости события

Для влияния на занятость есть три варианта: события не влияют на занятость; влияют те, где я участник; влияют все.

Влияние событий на занятость

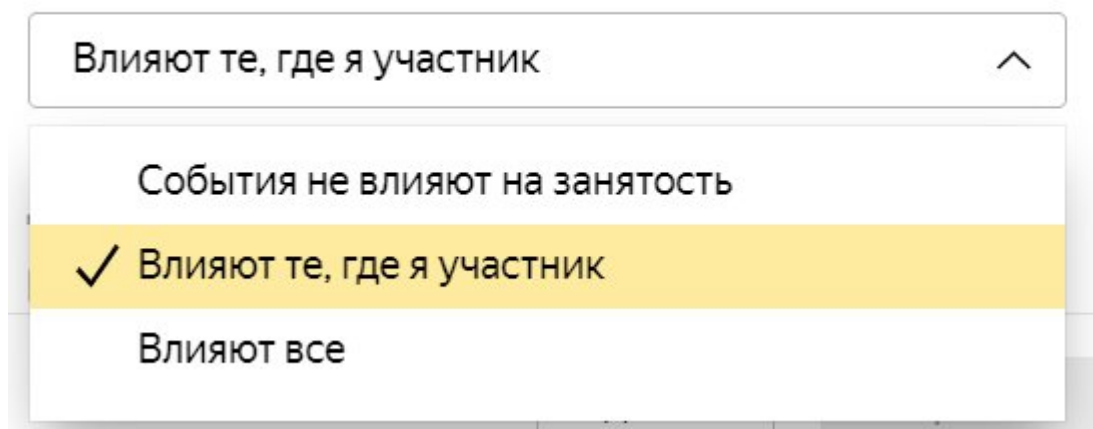


Рисунок 7 – Интерфейс настройки влияния на занятость

К достоинствам с точки зрения составителя календарей можно отнести то, что сервис обладает интуитивно понятным, наглядным интерфейсом.

Главными недостатками данного сервиса для составителя будут то, что расписание каждого учителя и класса нужно составлять индивидуально, проставляя каждое событие вручную, а также то, что для добавления календаря с расписаниями в личный календарь ученика составитель должен знать адрес электронной почты каждого ученика.

Со стороны пользователей, просматривающих календарей можно отметить, как недостатки то, что нужно подтверждать добавление каждого события в календарь (составителю на электронную почту придёт уведомление о принятии или отклонении добавления событий), то, что события добавляются в один календарь (не смотря на то, что составляющий календари пользователь может распределять их в разные), а также то, что уведомления о предстоящих событиях не всегда приходят вовремя.

Параграф 3

Как сказано в учебно-методическом пособии «Информационная структура комплекса «Параграф-3» и основные приемы работы с приложениями» за авторством И. П. Невзоровой и М. И. Скалецкой:

«Программный комплекс «ПараГраф-3» используется в образовательных учреждениях Санкт-Петербурга с 2006 года и является основой информационной среды образовательного учреждения.» [25, с. 2].

Интерфейс данного комплекса позволяет работнику просмотреть список своих задач, пример которого виден на рисунке 8, и своё расписание, пример которого можно увидеть на рисунке 9.

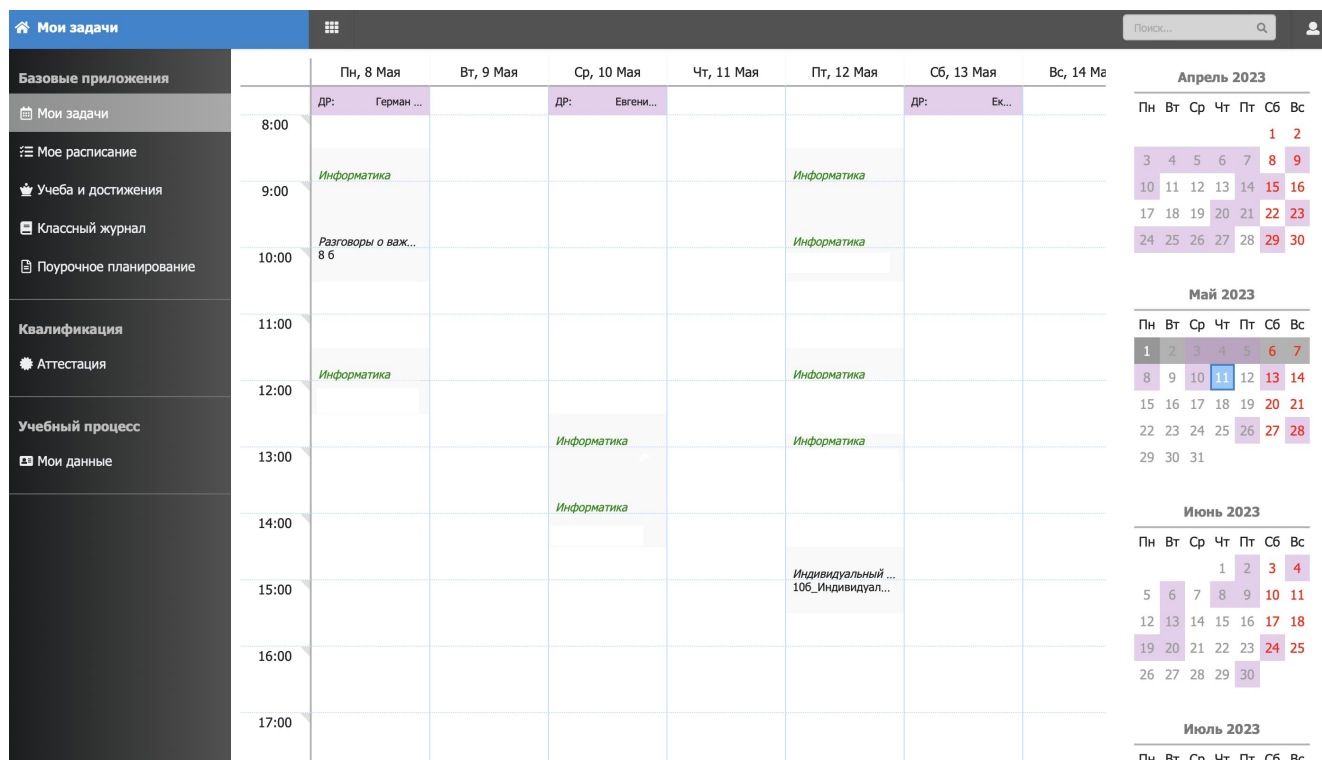


Рисунок 8 – Интерфейс просмотра задач

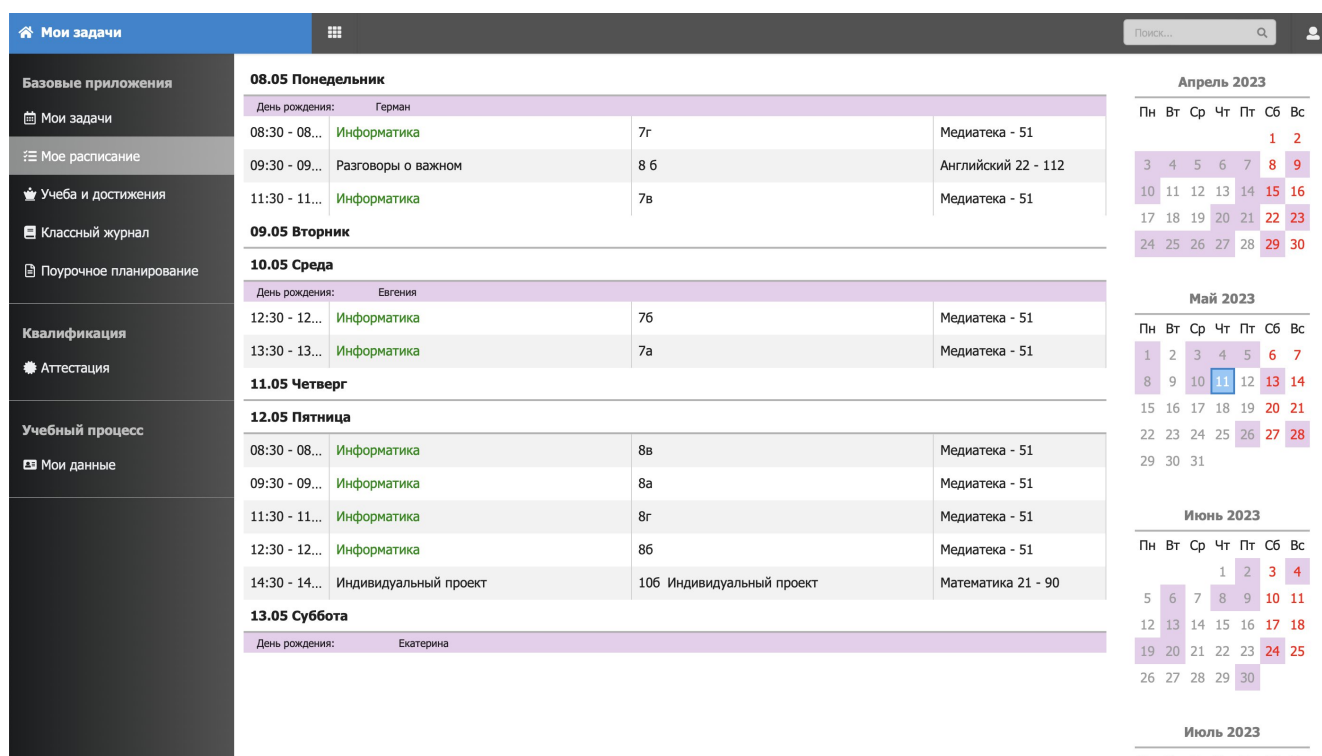


Рисунок 9 – Интерфейс просмотра расписаний

Преимущества:

- календарь автоматически составляется по расписанию.

Недостатки:

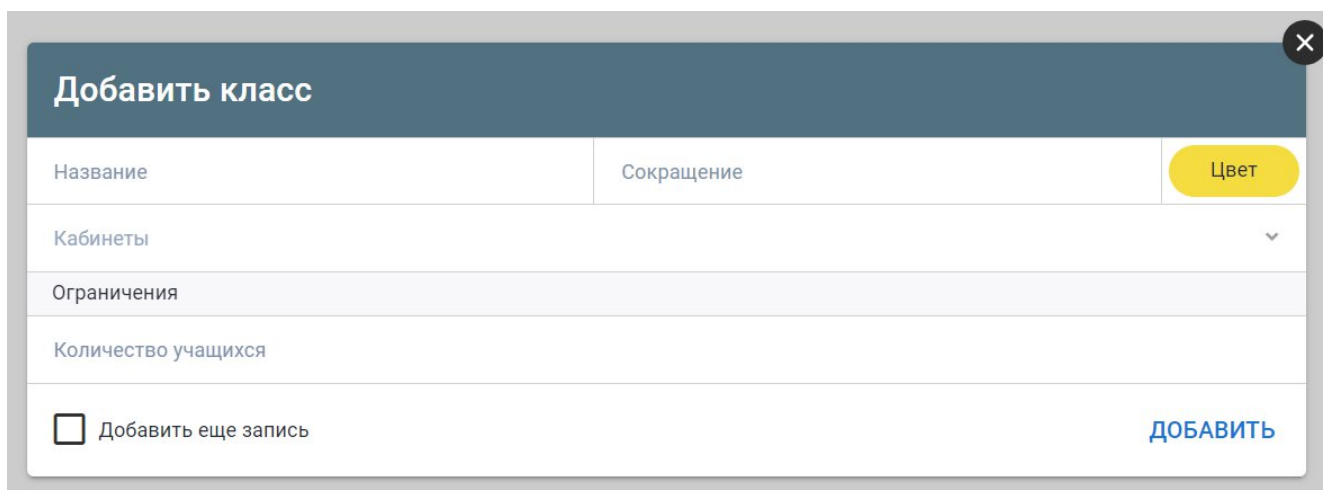
- отсутствие возможности добавления событий, не привязанных к урокам в расписании;
- отсутствие возможности просмотра расписания вне школы, так как сервис развёрнут в локальной сети.

Timetable

Timetable (mstimetables.ru) – веб-сервис для составления расписаний с возможностью загрузки отчётов.

При составлении расписания учитываются пять основных единиц: классы, предметы, учителя, кабинеты и виды занятий; справочники с которыми нужно предварительно заполнить.

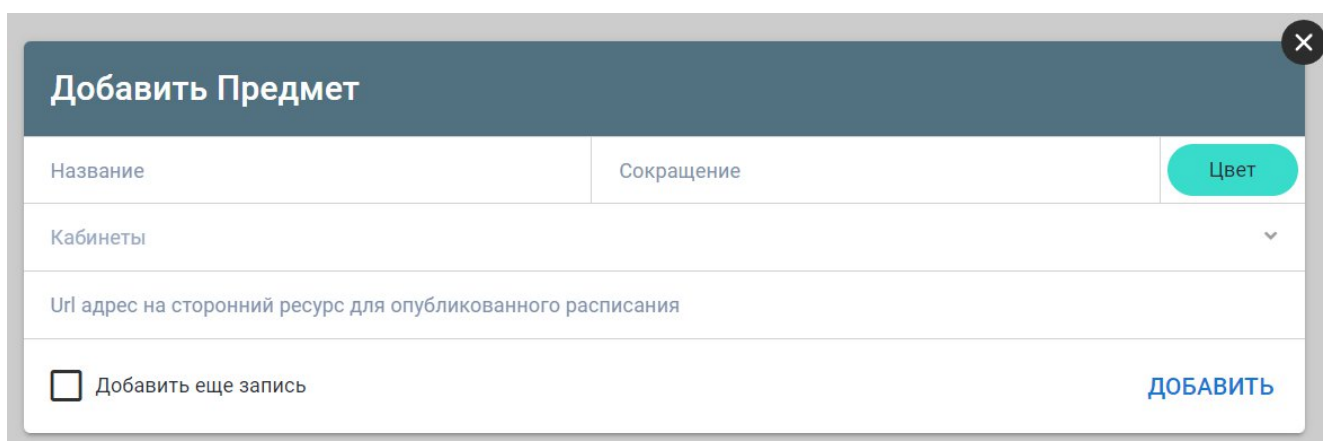
В соответствии с рисунком 10, при добавлении класса вводятся: название, сокращение названия, кабинеты класса, количество учащихся в классе; выбирается цвет из набора цветов.



Добавить класс		
Название	Сокращение	Цвет
Кабинеты		
Ограничения		
Количество учащихся		
<input type="checkbox"/> Добавить еще запись		ДОБАВИТЬ

Рисунок 10 – Интерфейс добавления класса

В соответствии с рисунком 11, при добавлении предмета вводятся: название, сокращение названия, кабинеты предмета, URL на сторонний ресурс; выбирается цвет из набора цветов.

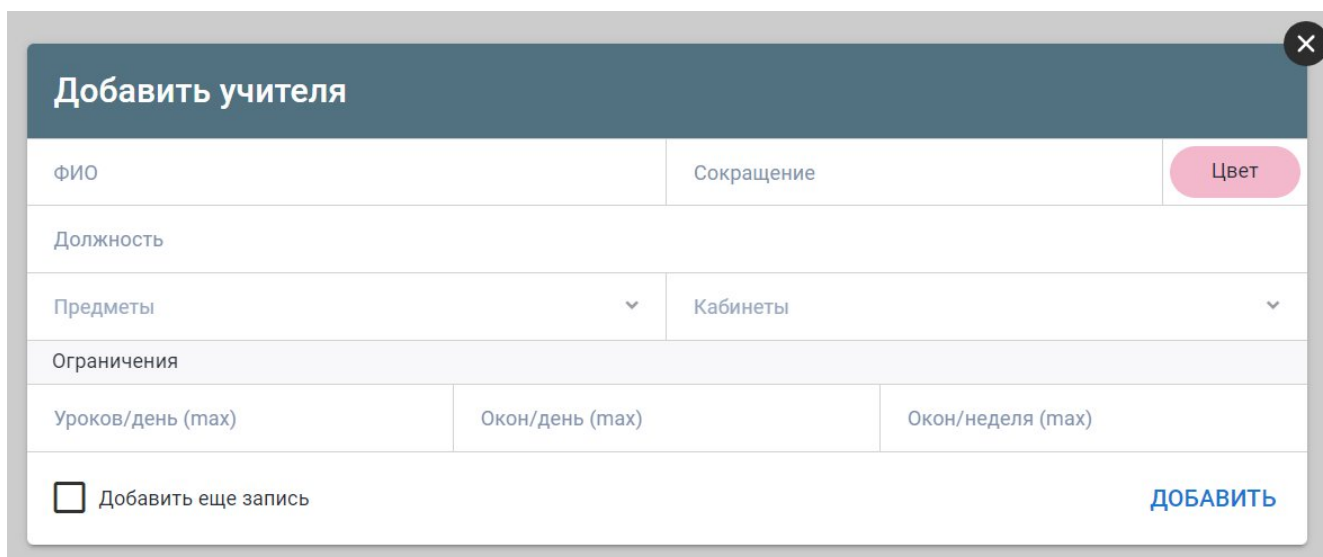


Добавить Предмет		
Название	Сокращение	Цвет
Кабинеты		
Url адрес на сторонний ресурс для опубликованного расписания		
<input type="checkbox"/> Добавить еще запись		ДОБАВИТЬ

Рисунок 11 – Интерфейс добавления предмета

В соответствии с рисунком 12, при добавлении учителя вводятся: ФИО, сокращение ФИО, должность, предметы учителя, кабинеты учителя,

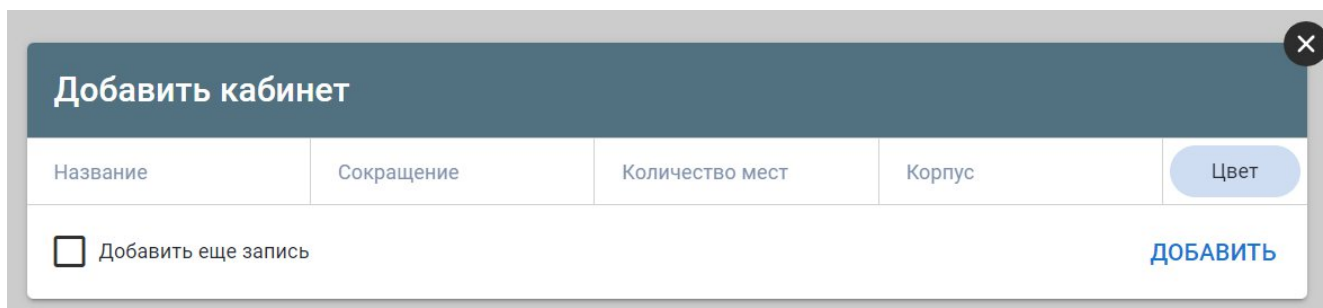
ограничения по максимальному числу уроков и окон в день и окон в неделю;
выбирается цвет из набора цветов.



Добавить учителя		
ФИО	Сокращение	Цвет
Должность		
Предметы	Кабинеты	
Ограничения		
Уроков/день (max)	Окн/день (max)	Окн/неделя (max)
<input type="checkbox"/> Добавить еще запись		ДОБАВИТЬ

Рисунок 12 – Интерфейс добавления учителя

В соответствии с рисунком 13, при добавлении кабинета вводятся:
название, сокращение названия, количество мест в кабинете, корпус;
выбирается цвет из набора цветов.



Добавить кабинет				
Название	Сокращение	Количество мест	Корпус	Цвет
<input type="checkbox"/> Добавить еще запись				ДОБАВИТЬ

Рисунок 13 – Интерфейс добавления кабинета

В соответствии с рисунком 14, при добавлении вида занятий вводятся:
название, сокращение названия для сетки уроков, сокращение названия для
печати.

Добавить вид занятия			
Название	Сокращение на сетке уроков	На печать	Цвет
<input type="checkbox"/> Добавить еще запись			ДОБАВИТЬ

Рисунок 14 – Интерфейс добавления вида занятия

После заполнения этих основных единиц добавляется расписание. В соответствии с рисунком 15 вводится название расписания, определяется тип расписания: шаблон или период; выбираются: тип недели: одна неделя или четная / нечетная; учебные дни в неделе, количество уроков в день, необходимость использовать несколько сеток звонков.

Добавить расписание			
Название	Тип	Шаблон	
Тип недели	Дней в неделю	Уроков в день	
Одна неделя	пн, вт, ср, чт, пт, сб, вс	7	
<input type="checkbox"/> Использовать несколько сеток звонков			
			ДОБАВИТЬ

Рисунок 15 – Интерфейс добавления расписания

После добавления расписания добавляется нагрузка. В соответствии с рисунком 16, при добавлении нагрузки выбираются: классы, необходимость деления на группы, вид занятия, учитель, предмет, кабинеты, используемые для урока, сложность в соответствии с СанПИН, вводится количество уроков.

- возможность подробно настроить каждую составляющую расписания (класс, кабинет, урок, учителя, определить для учителя его кабинет и при составлении расписания подставлять кабинет учителя);
- наличие документации.

Недостатки:

- отсутствует возможность создания расписания типа «Период», начинающегося не с понедельника (например, когда 1-ое сентября не в понедельник, первая неделя неактивна);
- расписание не копируется на следующую неделю автоматически, нужно это делать вручную;
- нет функции генерации личных электронных календарей.

Вывод

Каждое из рассмотренных средств обладает определёнными преимуществами, так например Яндекс календарь позволяет удобно создавать электронные календари, Параграф 3 составляет расписания учителей и классов исходя из общего расписания, Timetable обладает удобным интерфейсом и функциями, которые позволяют удобно создавать, обновлять и редактировать расписание, но при этом ни один из сервисов не обладает функционалом для выполнения поставленной задачи представления информации в образовательной организации с учетом требований заказчика. В связи с этим следует разработать программный продукт, учтя все преимущества и недостатки рассмотренных сервисов.

1.2 Выбор средств реализации

Исходя из имеющихся потребностей необходимо выбрать следующие средства реализации программного продукта:

- язык программирования для серверной составляющей;
- средство реализации клиентской составляющей;

- СУБД;
- система управления версиями;
- средство контейнеризации.

Язык программирования серверной составляющей

Для выбора наиболее оптимального языка программирования серверной составляющей рассмотрим несколько наиболее известных и подходящих из них.

Python

Python – это высокоуровневый язык программирования общего назначения с динамической типизацией, выпущенный 20 февраля 1991 года Гвидо ван Россумом.

К преимуществам языка Python относятся:

- простой синтаксис и хорошая читаемость кода; так Марк Лутц в своей книге «Изучаем Python» пишет, что «Большинство людей находят Python легким в изучении и увлекательным в использовании, особенно в сравнении с его ровесниками наподобие Java, C# и C++» [22, с. 46], а также то, что «Код Python по замыслу должен быть читабельным ... в гораздо большей степени, чем традиционные языки написания сценариев» [22, с. 40] и «обычно занимает от одной трети до одной пятой части размера эквивалентного кода C++ или Java» [22, с. 41];
- широкая стандартная библиотека и возможность расширения кода сторонними библиотеками и прикладным ПО; Марк Лутц в своей книге «Изучаем Python» отмечает, что «Область стороннего ПО для Python предлагает инструменты, предназначенные для конструирования веб-сайтов, численного программирования, доступа к последовательным портам, разработки игр и многого другого» [22, с. 41].

К недостаткам языка Python относятся:

- низкая производительность, Марк Лутц в книге «Изучаем Python» отмечает, что «значительный недостаток текущей реализации Python

связан с тем, что скорость выполнения может не всегда быть такой же, как у полностью компилируемых и низкоуровневых языков, подобных С и С++» [22, с. 44];

- специфичность синтаксиса.

Java

Java – это язык программирования общего назначения со статической типизацией, выпущенный в 1995 году компанией Sun Microsystems.

Преимущества языка Java:

- безопасность; Герберт Шилдт в книге «Java. Полное руководство, 10-е издание» пишет: «Тот факт, что программа на Java выполняется виртуальной машиной JVM, способствует также повышению ее безопасности. Виртуальная машина JVM управляет выполнением программы, поэтому она может изолировать программу, чтобы создать ограниченную исполняющую среду, которая называется «песочницей» и содержит программу, препятствующую неограниченному доступу к машине.» [32, с. 44];
- производительность в сравнении с Python; по результатам исследования "A comparison of common programming languages used in bioinformatics" [5], проведенного Матье Фурманом и Майклом Джиллингсом Java показал лучшие результаты в производительности в сравнении с Python.

Недостатки языка Java:

- большая сложность и меньшая читаемость кода в сравнении с Python.

Go

Go – это язык программирования общего назначения со статической типизацией, выпущенный компанией Google 10 ноября 2009 года.

Преимущества языка Go:

- простой и понятный си-подобный синтаксис; Марк Саммерфильд в книге «Программирование на Go. Разработка приложений XXI века»

отмечает: «Язык Go имеет очень простой и понятный синтаксис, в котором отсутствуют сложные и замысловатые конструкции, характерные для более старых языков, таких как C++ (появившегося в 1983 году) или Java (появившегося в 1995 году).» [20, с. 13];

- большое количество встроенных утилит и библиотек;
- большая производительность в сравнении с Python [7] и Java [6] по данным сайта Benchmarks Game при тестировании моделирования орбит планет-гигантов.

Недостатки языка Go:

- направленность преимущественно на веб-сервисы;
- малая распространённость.

Средства реализации клиентской составляющей

Для реализации клиентской составляющей используются фреймворки и библиотеки. Наиболее популярными фреймворками для реализации клиентской составляющей на данный момент являются:

- Vue;
- Angular.

Помимо фреймворков существуют библиотеки, например:

- React;
- JSRender/JSView.

React

React – это открытая JavaScript библиотека для написания клиентской части веб-сервисов, разработанная и выпущенная Джорданом Валке 29 мая 2013 года.

Преимущества:

- высокая популярность;
- меньший размер проектов в сравнении с Angular и Vue.

Недостатки:

- сложность изучения.

Vue

Vue - JavaScript-фреймворк с открытым исходным кодом для написания клиентской составляющей, выпущенный Эваном Ю в феврале 2014 года.

Преимущества:

- меньший размер проекта в сравнении с Angular [29];
- проще в освоении в сравнении с Angular [29].

Недостатки:

- большой размер проектов в сравнении с JSRender/JSView.

Angular

Angular – фреймворк с открытым исходным кодом для написания клиентской составляющей, выпущенный компанией Google 15 сентября 2016 года.

Преимущества:

- долгосрочная поддержка и документация от Google;
- лучшая организация и предсказуемость кода, связанная с использованием TypeScript.

Недостатки:

- большой размер проектов в сравнении с Vue и JSRender/JSView.

JSRender/JSView

JSRender/JSView – это открытая JavaScript библиотека для написания клиентской части веб-сервисов с использованием шаблонов.

Преимущества:

- простота;
- малый размер проекта; для него не требуется никаких дополнительных директорий и файлов;
- наличие полной документации и песочницы.

Недостатки:

- меньший функционал в сравнении с остальными фреймворками.

Сравнение СУБД

Проведём сравнение и выберем наиболее оптимальный способ реализации базы данных.

SQLite

SQLite – свободная встраиваемая СУБД, выпущенная Ричардом Хиппом в 2000 году.

Преимущества:

- все данные хранятся в одном файле, что упрощает миграцию.

Недостатки:

- малое количество поддерживаемых типов данных.

MySQL

MySQL – свободная СУБД, разрабатываемая компанией Oracle и вышедшая в 2000 году.

Преимущества:

- более высокая скорость чтения в сравнении с PostgreSQL.

Недостатки:

- более низкая скорость обработки сложных запросов в сравнении с PostgreSQL.

PostgreSQL

PostgreSQL – свободная СУБД, выпущенная Майклом Стоунбрейкером в 1996 году.

Преимущества:

- «наличие великолепной документации и активного сообщества позволяет быстро находить ответы на возникающие вопросы.

Руководство по PostgreSQL насчитывает свыше 2500 страниц» [16, с. 48];

- поддержка NoSQL и разнообразие типов данных.

Недостатки:

- более низкая скорость чтения в сравнении с MySQL.

Сравнение систем управления версиями

Проведём сравнение и выберем наиболее оптимальный вариант системы управления версиями.

Git

Git – распределённая система управления версиями, выпущенная 7 апреля 2005 года Линусом Торвальдсом для управления разработкой ядра Linux. Разрабатывается Software Freedom Conservancy.

Преимущества:

- распределённость;
- быстроедействие;
- кроссплатформенность;
- простота отслеживания изменений в коде;
- широкая распространённость, по данным Open Hub, представленным на рисунке 18.

Недостатки:

- большая сложность в освоении;
- при увеличении размера проекта отслеживать историю изменений становится затруднительно.

Mercurial

Mercurial – распределённая система управления версиями, выпущенная 19 апреля 2005 года Мэтом Мэколом.

Преимущества:

- распределённость;

- простота в освоении.

Недостатки:

- малая распространённость, по данным Open Hub, представленным на рисунке 18.

SVN

SVN – централизованная система управления версиями, выпущенная 20 октября 2000 года компанией CollabNet. С 2010 года разрабатывается Apache Software Foundation.

Преимущества:

- широкое распространение в среде более старых и проприетарных проектов.

Недостатки:

- централизованность;
- малая распространённость, по данным Open Hub, представленным на рисунке 18;
- меньшая скорость.

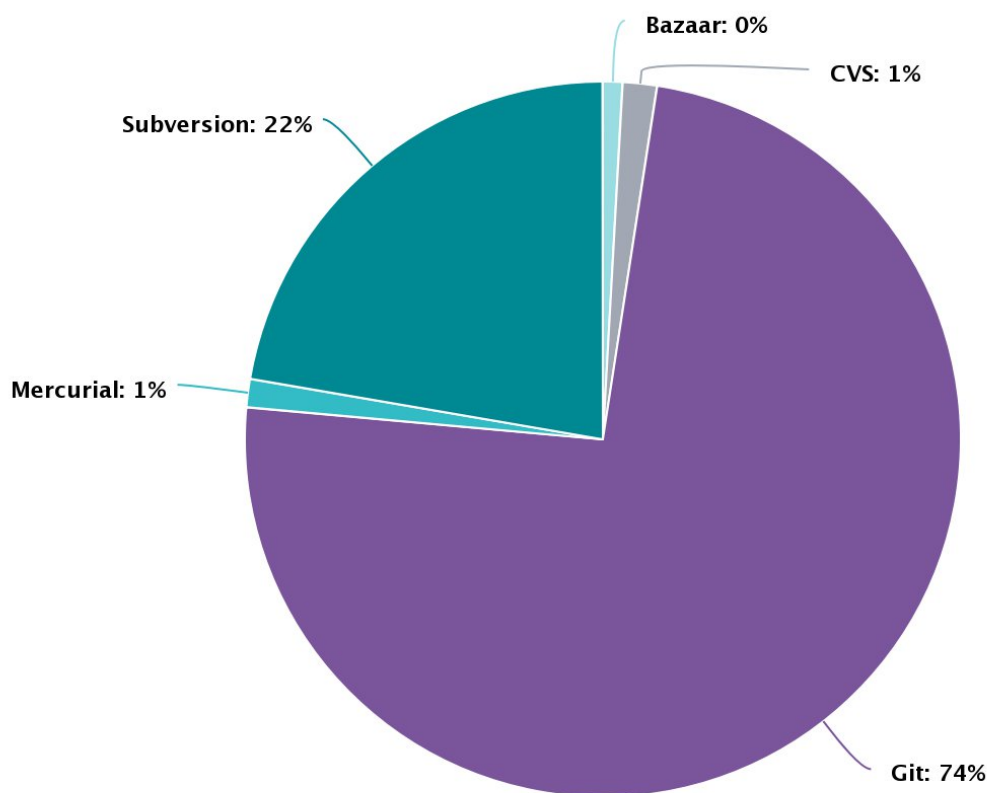


Рисунок 18 – Диаграмма распространённости систем контроля версий по данным Open Hub на начало-середину мая 2023 [3]

Сравнение средств контейнеризации

Проведём сравнение и выберем наиболее оптимальный вариант средства контейнеризации.

Docker

Docker – средство контейнеризации, выпущенное Соломоном Хайксом 13 марта 2013 года. Разрабатывается компанией Docker.

Преимущества:

- большая распространённость [2];
- поддерживает как ОС Linux, так и ОС Windows.

Недостатки:

- ограничение контейнера одним приложением.

LXC

LXC – средство контейнеризации, выпущенное Даниэлем Лескано, Сержем Айюном, Стефаном Грабе 6 августа 2008 года.

Преимущества:

- легковесность и производительность контейнеров.

Недостатки:

- не поддерживает ОС Windows.

Вывод

В связи с необходимостью разработки небольшого, но производительного веб-приложения для представления информации об образовательной организации наиболее оптимальным выбором является язык программирования Go, так как он сочетает в себе направленность на веб-сервисы, высокую производительность и простой си-подобный синтаксис.

Наиболее оптимальным решением для клиентской составляющей является библиотека JSRender/JSView, так как она способна предоставить весь необходимый для разрабатываемого сервиса функционал с меньшими требованиями к размеру проекта и большей простотой сопровождения.

В качестве СУБД для разрабатываемого проекта лучше всего подходит PostgreSQL, так как она поддерживает большое множество типов данных и возможность реализации NoSQL.

Наиболее подходящей системой контроля версий является GIT, так как она является распределённой и кроссплатформенной системой с высоким быстродействием, а её широкая распространённость позволяет без затруднений справиться с любой возникшей проблемой при помощи поиска соответствующей информации на форумах и в документации.

Предпочтительным средством контейнеризации для проекта является Docker, так как он поддерживает как ОС Linux, так и ОС Windows, а благодаря

большей распространённость информацию о популярных решения возникших проблем можно будет без затруднений найти на форумах и связанных ресурсах.

ГЛАВА 2. РАЗРАБОТКА

После выявления потребностей заказчика и определения оптимальных средств реализации проекта должна быть определена модель жизненного цикла для разрабатываемого программного продукта.

2.1 Жизненный цикл

В качестве модели жизненного цикла ПО была выбрана спиральная модель, схема которой представлена на рисунке 19. Спиральная модель позволяет изменять и улучшать продукт в соответствии с меняющимися требованиями заказчика.

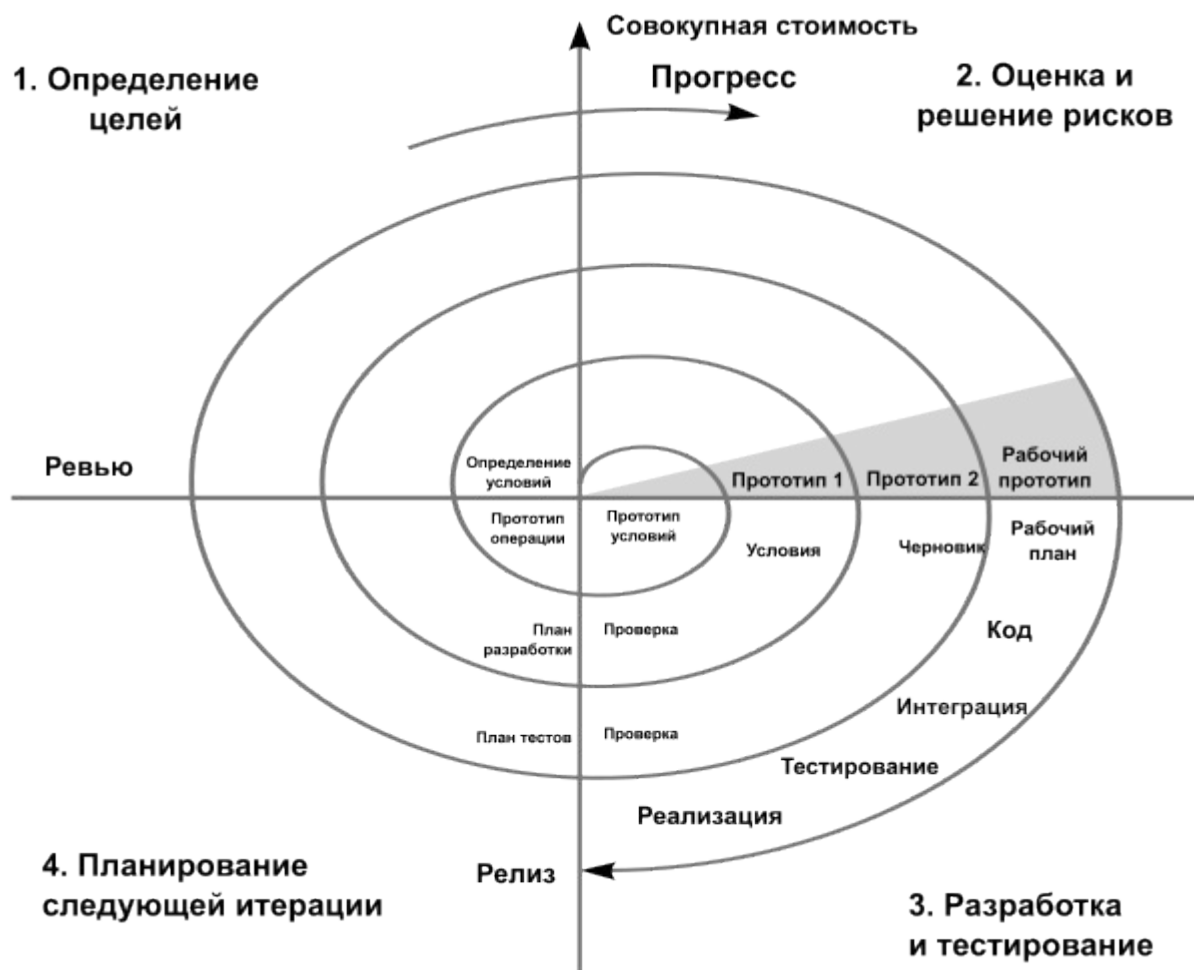


Рисунок 19 – Схема спиральной модели [28]

2.2 Схема взаимодействия компонентов системы

Взаимодействие внутри системы происходит в соответствии со схемой, представленной на рисунке 20.

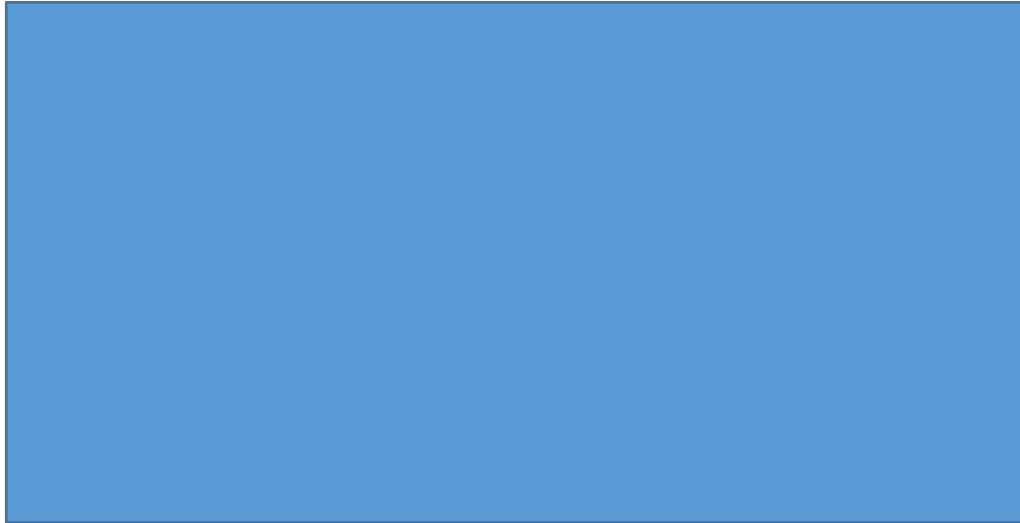


Рисунок 20 – Схема взаимодействия компонентов распределённой системы

2.3 Создание базы данных

В условиях постоянных изменений требований к информатизации в области образования, нужно использовать наиболее подходящий для этого вид базы данных. NoSQL за счёт отсутствия необходимости соответствия реляционной модели упрощает процессы масштабирования и доступа к данным. В связи с этим было принято решение придерживаться NoSQL модели.

Для обеспечения работы системы была создана модель реализации базы данных, включающая в себя пять таблиц:

- class – таблица, содержащая информацию о классах;
- teacher – таблица, содержащая информацию об учителях;
- room – таблица, содержащая информацию о кабинетах;
- subject – таблица, содержащая информацию о предметах;
- schedule – таблица, содержащая информацию о расписаниях;

и двух пользователей:

- getter – пользователь, ответственный за получение информации из таблиц;
- setter – пользователь, ответственный за изменение информации в таблицах.

К NoSQL составляющей базы данных относится таблица schedule, в которой поле data отвечает за хранение информации о расписании в формате JSON. На рисунке 21 показан список полей таблицы schedule. Рисунок 22 показывает, как заполняется поле data в таблице schedule.

```
id serial PRIMARY KEY,  
start varchar(10),  
year int NOT NULL,  
parallel int NOT NULL,  
is_base boolean NOT NULL,  
data jsonb[] NOT NULL,  
UNIQUE(start, year, parallel, is_base)
```

Рисунок 21 – Поля таблицы schedule



Рисунок 22 – Схема заполнения поля data таблицы schedule

Для реализации базы данных в системе было принято решение использовать Docker-контейнер PostgreSQL последней версии.

2.4 Разработка сервера для взаимодействия с базой данных

Для работы с базой данных предусмотрен сервер, принимающий запросы от клиентов, выполняющий необходимые запросы у базы данных, формирующий данные в оптимальный для клиента формат и отправляющий данные клиенту.

Сервер обрабатывает пять маршрутов получения данных:

- маршрут получения данных о классах;
- маршрут получения данных об учителях;
- маршрут получения данных о кабинетах;
- маршрут получения данных о предметах;
- маршрут получения данных о расписаниях;

и пять маршрутов изменения данных:

- маршрут изменения данных о классах;
- маршрут изменения данных об учителях;
- маршрут изменения данных о кабинетах;
- маршрут изменения данных о предметах;
- маршрут изменения данных о расписаниях.

Для обработки данных в формате JSON был разработан отдельный модуль. А сам сервер был упакован в Docker контейнер.

2.5 Разработка сервера для предоставления пользовательский интерфейс

Для разработки пользовательского интерфейса была использована библиотека JSRender/JSView. При открытии пользователем соответствующей страницы интерфейса происходит запрос данных от сервера:

- для страницы изменения данных о классах, пример которой расположен на рисунке 23, запрашиваются данные о всех классах;



Рисунок 23 – Страница изменения данных о классах

- для страницы изменения данных об учителях, пример которой расположен на рисунке 24, запрашиваются данные о всех учителях;



Рисунок 24 – Страница изменения данных об учителях

- для страницы изменения данных о кабинетах, пример которой расположен на рисунке 25, запрашиваются данные о всех кабинетах;



Рисунок 25 – Страница изменения данных о кабинетах

- для страницы изменения данных о предметах, пример которой расположен на рисунке 26, запрашиваются данные о всех предметах;

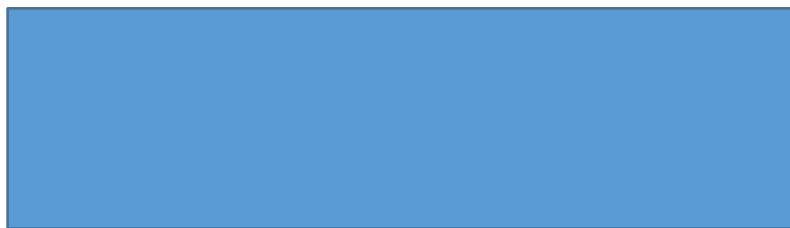


Рисунок 26 – Страница изменения данных о предметах

- для страницы изменения данных о расписании, пример которой расположен на рисунке 27, запрашиваются данные о:
 - всех классах;
 - всех учителях;
 - всех кабинетах;
 - всех предметах;
 - существующем расписании на эту неделю.



Рисунок 26 – Страница изменения данных о расписании

Сервер, предоставляющий пользовательский интерфейс был упакован в отдельный Docker-контейнер.

2.6 Разработка сервера для генерации файлов календарей

Для генерации файлов электронных календарей был разработан отдельный сервер. Этот сервер раз в сутки запрашивает данные о расписании

на актуальную и следующую неделю, после чего, используя специально разработанный для этого модуль, генерирует файлы электронных календарей, пример содержимого которых расположен на рисунке 27.



Рисунок 27 – Пример содержимого файла электронного календаря с расписанием учителя

Сервер для генерации электронных календарей был упакован в отдельный Docker-контейнер.

2.7 Разработка сервера для доставки файлов календарей

Для доставки файлов календарей было принято решение развернуть отдельный сервер, который будет работать в связке с сервером генерации электронных календарей и предоставлять файлы расписаний для учителей и учеников по соответствующим маршрутам по примеру рисунка 82.



Рисунок 28 – пример пути до файла календаря 8-го А класса

Сервер, предоставляющий файлы расписаний был упакован в отдельный Docker контейнер.

ВЫВОД

В ходе выполнения работ по проекту был проведён анализ требований заказчика и существующих решений, которые могли бы удовлетворять эти требования. На основании результатов анализа были выбраны наиболее подходящие средства реализации проекта.

Для реализации проекта была выбрана спиральная модель жизненного цикла. Была спроектирована и реализована модель NoSQL базы данных. Была спроектирована модель распределённой системы, в соответствии с которой были разработаны серверная и клиентская составляющие веб-приложения. Для всех составляющих распределённой системы была выполнена контейнеризация с помощью Docker.

Для возможности внедрения и использования разработанной системы была составлена сопроводительная документация, а именно: руководство администратора и руководство пользователя.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Adrian M. Using Docker - Sebastopol: O'Reilly Media, Inc, - 2015.
2. Compare Projects - Open Hub [Электронный ресурс].
https://www.openhub.net/p/_compare?project_0=docker&project_1=Linux+Containers&submit_1=Go.
3. Compare Repositories - Open Hub [Электронный ресурс].
<https://www.openhub.net/repositories/compare>.
4. Donovan A. A. The Go Programming Language. / A. A. Donovan, B. W. Kernighan. — Addison-Wesley Professional. 2015.
5. Fourment, M. A comparison of common programming languages used in bioinformatics / M. Fourment, M. R. Gillings // BMC Bioinformatics. – 2008. – Vol. 9. – P. 82. – DOI 10.1186/1471-2105-9-82. – EDN XWUTLM.
6. Go vs Java - Which programs are fastest? [Электронный ресурс]. Режим доступа: <https://benchmarksgame-team.pages.debian.net/benchmarksgame/fastest/go.html>.
7. Go vs Python 3 - Which programs are fastest? [Электронный ресурс]. Режим доступа: <https://benchmarksgame-team.pages.debian.net/benchmarksgame/fastest/go-python3.html>.
8. LernerR. M. At the forge: PostgreSQL, the NoSQL database // Linux Journal. 2014. № 247.
9. PostgreSQL 9.6: Documentation: 9.6: JSON Types [Электронный ресурс]. Режим доступа: <https://www.postgresql.org/docs/9.6/static/datatype-json.html/>
10. Sarah Mei Why You Should Never Use MongoDB [Электронный ресурс]: 2013. Режим доступа: <http://www.sarahmei.com/blog/2013/11/11/why-you-should-never-use-mongodb/>.
11. Адрова Л.С., Полежаев П.Н. Сравнительный анализ существующих технологий контейнеризации. - [Электронный ресурс]. - Режим доступа: <http://elib.osu.ru/bitstream/123456789/1955/1/2473-2477.pdf>

- 12.Власов К.П., Трамов И.Б., Кирпиченко М.С., Саадиев А.С. ОЦЕНКА ВЛИЯНИЯ ФУНКЦИОНАЛЬНОЙ ПОЛНОТЫ СИСТЕМ РАЗВЕРТЫВАНИЯ НА СКОРОСТЬ ЗАПУСКА КОНТЕЙНЕРОВ // Международный журнал гуманитарных и естественных наук. 2022. №3-2.
- 13.ГОСТ Р ИСО/МЭК12207-99 Процессы жизненного цикла программных средств.
- 14.Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. Приемы объектно-ориентированного проектирования. Паттерны проектирования. СПб.: Питер, 2001.
- 15.Дейт К. Дж. Введение в системы баз данных. -8-е изд.: Пер. с англ. - М.: «Вильямс», 2005.
- 16.Джуба С., Волков А. Изучаем PostgreSQL 10 / пер. с англ. А. А. Слинкина. – М.: ДМК Пресс, 2019. – С. 48.
- 17.Емельянова Н.З., Партыка Т.Л., Попов И.И. Проектирование информационных систем: учебное пособие. Москва: Форум: НИЦ ИНФРА-М, 2014.
- 18.Конноли Т. Базы данных: проектирование, реализация, сопровождение / Т. Конноли, К. Бегг, А. Страчан. - М.: Вильямс, 2003
- 19.Коптева Анна Витальевна, Князев Илья Вадимович СОВРЕМЕННЫЙ ПОДХОД К УПРАВЛЕНИЮ ТАЙМ-АУТОМ БАЗЫ ДАННЫХ И ОТМЕНОЙ ЗАПРОСОВ В GOLANG // Наука, техника и образование. 2021. №8 (83).
- 20.Саммерфильд М. Программирование на Go. Разработка приложений XXI века: пер. с англ.: Киселёв А. Н. – М.: ДМК Пресс, 2013. – С. 13.
- 21.Ларман К. Применение UML и шаблонов проектирования. Вильямс, 2002.
- 22.Лутц М. Изучаем Python, том 1, 5-е изд.: Пер. с англ. — СПб.: ООО “Диалектика”, 2019.— С. 40-41, 44, 46.

23. Мартин Фаулер, Прамодкумар Дж. Садаладж. NoSQL: новая методология разработки нереляционных баз данных: Пер. с англ.- М.: «И. Д. Вильямс», 2013.
24. Михаэл Стоунбрейкер. Объектно-реляционные системы баз данных // «Открытые системы», N 04, 1994.
25. Невзорова И.П., Скалецкая М.И. Информационная структура комплекса «Параграф-3» и основные приемы работы с приложениями: Учебно-методическое пособие. – 2-е изд., перераб. и доп.– СПб., ГБУ ДПО «СПбЦОКОиИТ», 2016. – С. 2.
26. Новиков Борис Асенович СРАВНИТЕЛЬНЫЙ анализ производительности SQL И NOSQL СУБД // КИО. 2017. №4.
27. Пуцин М.Н. Проектирование информационных систем. М.: Изд-во МИЭТ, 2008.
28. Спиральная модель — Википедия [Электронный ресурс]. https://ru.wikipedia.org/wiki/Спиральная_модель
29. Сравнение с другими фреймворками — Vue.js [Электронный ресурс]. Режим доступа: <https://ru.vuejs.org/v2/guide/comparison.html>.
30. Цебренок К.Н. РАЗРАБОТКА РАСПРЕДЕЛЕННОЙ СИСТЕМЫ ОБМЕНА УВЕДОМЛЕНИЯМИ НА ОСНОВЕ МИКРОСЕРВИСНОЙ АРХИТЕКТУРЫ // Международный журнал гуманитарных и естественных наук. 2021. №8-1.
31. Цукалос М. Golang для профи: работа с сетью, многопоточность, структуры данных и машинное обучение с Go. М.: Прогресс книга, 2021.
32. Шилдт Г. Java. Полное руководство, 10-е изд. : Пер. с англ. -СПб. ООО "Альфакнига"; 2018. — С. 44.

ПРИЛОЖЕНИЕ 1

Инструкция для администратора

Для развёртывания системы понадобится два сервера:

1. Сервер, на котором будет развёрнута часть системы, отвечающая за хранение данных и предоставление пользовательского интерфейса составителю расписания.
2. Сервер, на котором будет развёрнута часть системы, отвечающая за генерацию и предоставление файлов электронных календарей с расписанием.

На обоих серверах должна быть установлена операционная система, поддерживающая работу Docker

Перед развёртыванием распределённой системы необходимо установить на каждый сервер Docker в соответствии с инструкцией, представленной на официальном сайте.

Для развёртывания системы нужно:

1. Скопировать на сервер, на котором будет развёрнута часть системы, отвечающая за хранение данных и предоставление пользовательского интерфейса составителю расписания, соответствующий файл `docker-compose.yml`.
2. Выполнить команду `“docker compose up”`
3. Скопировать на сервер, на котором будет развёрнута часть системы, отвечающая за генерацию и предоставление файлов электронных календарей с расписанием, соответствующий файл `docker-compose.yml`.
4. Выполнить команду `“docker compose up”`

После этого система будет развёрнута и готова к работе.

Для остановки работы системы необходимо выполнить на каждом сервере команду `“docker compose down”`.

ПРИЛОЖЕНИЕ 2

Инструкция для пользователя

Для составления расписания с использованием системы представления информации нужно запустить браузер и перейти по адресу, на котором развёрнута система. Этот адрес следует узнать у Вашего системного администратора.

Перед тем, как перейти к составлению расписания нужно заполнить информацию о:

- классах, для которых будет составляться расписание;
- учителях, работающих в школе;
- кабинетах, в которых будут вестись занятия;
- предметах, которые ведутся в школе.

Для добавления учителя нужно:

1. Открыть страницу изменения информации об учителях.
2. Нажать кнопку добавить.
3. Заполнить поля. Обязательными из них являются:
 - ФИО;
 - логин.
4. Нажать кнопку готово.

Для изменения информации об учителе нужно:

1. Открыть страницу изменения информации об учителях.
2. Изменить информацию в полях. Обязательными из них являются:
 - ФИО;
 - логин.
3. Нажать кнопку готово.

Для удаления учителя нужно:

1. Открыть страницу изменения информации об учителях.
2. Нажать кнопку удалить.
3. Нажать кнопку готово.

Для добавления кабинета нужно:

1. Открыть страницу изменения информации о кабинетах.
2. Нажать кнопку добавить.
3. Заполнить поля. Обязательными из них являются:
 - название.
4. Нажать кнопку готово.

Для изменения информации о кабинете нужно:

1. Открыть страницу изменения информации о кабинетах.
2. Изменить информацию в полях. Обязательными из них являются:
 - название.
3. Нажать кнопку готово.

Для удаления кабинета нужно:

1. Открыть страницу изменения информации о кабинетах.
2. Нажать кнопку удалить.
3. Нажать кнопку готово.

Для добавления предмета нужно:

1. Открыть страницу изменения информации о предметах.
2. Нажать кнопку добавить.
3. Заполнить поля. Обязательными из них являются:
 - название.
4. Нажать кнопку готово.

Для изменения информации о предмете нужно:

1. Открыть страницу изменения информации о предметах.
2. Изменить информацию в полях. Обязательными из них являются:
 - название.
3. Нажать кнопку готово.

Для удаления предмета нужно:

1. Открыть страницу изменения информации о предметах.
2. Нажать кнопку удалить.
3. Нажать кнопку готово.

Для добавления класса нужно:

1. Открыть страницу изменения информации о классах.
2. Нажать кнопку добавить.
3. Заполнить поля. Обязательными из них являются:
 - номер;
 - буква.
4. Нажать кнопку готово.

Для изменения информации о классе нужно:

1. Открыть страницу изменения информации о классах.
2. Изменить информацию в полях. Обязательными из них являются:
 - номер;
 - буква.
3. Нажать кнопку готово.

Для удаления класса нужно:

1. Открыть страницу изменения информации о классах.
2. Нажать кнопку удалить.
3. Нажать кнопку готово.

Для изменения расписания нужно:

1. Открыть страницу изменения расписаний.
2. Выбрать является ли расписание расписанием на год или изменением на неделю.
3. Если создаётся изменение на неделю, выбрать начало недели.
4. Выбрать параллель.
5. Заполнить информацию на неделю:
6. Для каждого урока каждого дня выбрать:
 - предмет;
 - кабинет;
 - учителя.
7. Нажать кнопку готово.

ПРИЛОЖЕНИЕ 3

РАСПРЕДЕЛЁННАЯ СИСТЕМА ПРЕДСТАВЛЕНИЯ ИНФОРМАЦИИ ДЛЯ ОБРАЗОВАТЕЛЬНОЙ ОРГАНИЗАЦИИ

Техническое задание

Титульный лист

Листов 10

2022

1 Введение

Распределённая система представления информации для образовательной организации – это система, позволяющая редактировать и представлять различными способами информацию об образовательной организации.

2 Основания для разработки

Основанием является заказ на разработку распределённой системы представления информации для образовательной организации от Государственного бюджетного общеобразовательного учреждения средней общеобразовательной школы №80 с углубленным изучением английского языка Петроградского района Санкт-Петербурга.

3 Назначение разработки

3.1 Функциональное назначение

Система предоставляет возможности создания расписания занятий для учеников и работников образовательной организации и внесения изменений в созданные расписания. Помимо этого система визуализирует полученные расписания, а также предоставляет возможность добавлять расписания в личный календарь.

3.2 Эксплуатационное назначение

Система позволяет облегчить с помощью частичной автоматизации процессы составления, представления и доставки информации о расписаниях занятий работников и учеников образовательной организации.

4 Требования к программе или программному изделию

4.1 Требования к функциональным характеристикам

Система состоит из двух частей: клиентской и серверной, между которыми должно быть налажено взаимодействие.

4.1.1 Требования к серверной части

Серверная часть должна представлять собой несколько взаимодействующих между собой микросервисов, собранных в виде Docker-контейнеров.

Должно быть организовано хранение информации о расписаниях в базе данных.

Также должно быть организовано взаимодействие с базой данных для осуществления представления информации о расписаниях.

Должен быть реализован интерфейс для получения информации о расписаниях, представленной в виде календарей iCal.

Должна быть осуществлена возможность вывода представленной информации.

4.1.2 Требования к клиентской части

На клиентской части требуется реализовать два компонента: приложение для создания расписаний и внесения изменений в расписания и интерфейс для подключения персонифицированного календаря.

4.1.3 Требования к взаимодействию клиентской и серверной части

Взаимодействие между клиентской и серверной частями должно осуществляться с помощью HTTP-запросов. Обмен данными о расписаниях между клиентом и сервером осуществляется через обмен сообщениями в формате JSON.

4.2 Требования к надёжности

4.2.1 Требования к обеспечению устойчивого функционирования системы

Пользователю, взаимодействующему с системой через веб-браузер должен быть предоставлен непрерывный доступ к веб-сервису, расположенному по определённом адресу. Веб-сервис не должен непредвиденно прерывать свою работу.

4.2.2 Время восстановления после отказа

В случае отказа работы серверной части и последующей недоступности веб-сервиса, время восстановления не должно превышать одни сутки.

4.2.3 Отказы из-за некорректных действий пользователя

После запуска системы на сервере отказ вследствие некорректных действий пользователя должен быть исключён .

4.3 Условия эксплуатации

4.3.1 Климатические условия эксплуатация

Требования к климатическим условиям эксплуатации не предъявляются.

4.3.2 Требования к видам обслуживания

Обслуживание не требуется.

4.3.3 Требования к численности и квалификации персонала

Для использования системы достаточно одного человека, обладающего достаточными квалификациями для составления расписания занятий в образовательной организации.

4.4 Требования к составу и параметрам технических средств

4.4.1 Требования к серверу

Сервер должен быть совместим с любой из операционных систем, поддерживающей работу Docker.

4.4.2 Требования к клиенту

Компьютер клиента должен иметь доступ к сети, в которой работает сервер, а также на нём должен быть установлен любой веб-браузер.

4.5 Требования к информационной и программной совместимости

4.5.1 Требования к исходным кодам и языкам программирования

Исходные коды серверной составляющей должны быть написаны на языке программирования Go.

5 Требования к программной документации

Состав программной документации:

1. Руководство по установке и настройке системы.
2. Руководство пользователя.

6 Технико-экономические показатели

6.1 Ориентировочная экономическая эффективность

В рамках данной работы расчёт экономической эффективности не предусмотрен. Использование разрабатываемой системы сократит время, затрачиваемое на составление, представление и доставку информации о расписаниях в образовательной организации.

6.2 Предполагаемая потребность

Предполагаемая потребность обуславливается трудоёмкостью процессов составления, представления и доставки информации о расписаниях в образовательной организации.

7 Стадии и этапы разработки

1. Определение данных, необходимых для работы системы.
2. Проектирование базы данных.
3. Разработка модели и архитектуры распределённой системы.
4. Разработка микросервиса, отвечающего за создание расписаний и внесения изменений в расписания.
5. Разработка микросервиса, отвечающего за представление расписаний в виде календаря iCal.
6. Разработка микросервиса, отвечающего за представление расписаний в графическом виде.
7. Контейнеризация каждого микросервиса.
8. Организация микросервисов, представленных в виде Docker-контейнеров, в систему с помощью docker-compose.

8 Порядок контроля и приемки

8.1 Виды испытаний

Проводится проверка корректного выполнения системой заложенных в неё функций, функциональное тестирование системы.

Проводится визуальная проверка интерфейса веб-сервиса.

8.2 Общие требования к приёмке системы

Приём системы будет утверждён при: корректной работе системы при различных входных данных и предоставлении документации к продукту.