Курсовая работа

Студенты механико-математического факультета 2 группы второго курса Корбут Владислав и Дудко Дмитрий

БГУ

2017

Оглавление

[Введение 1](#_Toc482718227)

[Описание проекта 1](#_Toc482718228)

[Модель предметной области 2](#_Toc482718229)

[Средства реализации 2](#_Toc482718230)

[Клиентская часть 2](#_Toc482718231)

[Single Page Application (SPA) 2](#_Toc482718232)

[API. 3](#_Toc482718233)

[Формат обмена данными. 3](#_Toc482718234)

[Передача данных 3](#_Toc482718235)

[Серверная часть 3](#_Toc482718236)

[Реализация 3](#_Toc482718237)

[Клиентская часть 3](#_Toc482718238)

[Шаблон 4](#_Toc482718239)

[Представление 5](#_Toc482718240)

[Маршрутизатор 5](#_Toc482718241)

[Контроллер 6](#_Toc482718242)

[Сервис 8](#_Toc482718243)

[Модель 12](#_Toc482718244)

[Общее представление веб-приложения 12](#_Toc482718245)

[ПРОЕКТИРОВАНИЕ 12](#_Toc482718246)

[Актеры 12](#_Toc482718247)

[Варианты использования: 13](#_Toc482718248)

[Диаграмма вариантов использования 13](#_Toc482718249)

[Диаграмма классов сущностей 13](#_Toc482718250)

[Диаграмма классов логики 13](#_Toc482718251)

[Диаграмма последовательности 13](#_Toc482718252)

[Диаграмма деятельности 14](#_Toc482718253)

[Диаграмма базы данных 14](#_Toc482718254)

# Введение

## Описание проекта

Расписание для факультета мехмат является неудобным и чересчур загромождено. Оно дает избыточную информацию обо всем, при этом многие пункты можно было убрать.

Целью написания нашей курсовой работы является оптимизация расписания. Каждый студент сможет выбрать свою группу и свой курс, при этом расписание будет разбито на дни.

Цель:

* Сделать раписание более удобным. Уменьшить время на поиск нужной информации

Целевая аудитория:

* Студенты факультета мехмат

# Модель предметной области

Веб-приложение предоставляет возможность пользователю просмотреть расписание для заданной группы и заданного курса.

Пользователь может выбрать свой курс и группу и приложение сохранит данные. При следующем визите расписание отобразится с учетом того, что данные были введены ранее. При необходимости смотреть расписание для другого курса/группы пользователь может изменить их в настройках приложения.

# Средства реализации

## Клиентская часть

Клиентская часть предполагает работу веб-приложение в браузере пользователя. Для оптимального удовлетворения большинства требований было принято решение разработать веб-приложение с использованием технологии Single Page Application.

### Single Page Application (SPA)

SPA - концепция одностраничного веб-приложения, предполагающая использование архитектурного решения “Толстый клиент” (предоставляет собой функциональный интерфейс пользователя с выполнением бизнес-логики) и API серверной части для oбмена данными.

Особенность “Толстого клиента” в данном случае заключается в том, что практически все промежуточные действия для получения документа происходят на клиенте и только данные принципиальных действий отправляются на сервер, где в качестве ответа приходят необходимые данные для достижения пользовательской цели.

Таким образом, это уменьшает зависимость от качества связи между клиентом и сервером. Без обращения клиента на сервер, смена представлений ПИ меняется практически мгновенно согласно UX. В случае обращения, обмен данными происходит в асинхронном режиме, что позволяет продолжать работу пользователя без блокирования клиента в ожидании ответа сервера.

За счет вынесения части бизнес-логики и обработки представлений на клиента, а также отделение представления от данных, - достигается слабая связанность клиентской и серверной части ВП. Таким образом, в сравнении с традиционным подходом организации ВП, работа 27 клиентской части ВП характеризуется высоким быстродействием (однако имеет зависимость от аппаратных характеристик клиента) и возможностью корректной работы при некачественном соединении.

Клиентское приложение.

Приложение, которое выполняется в браузере, изменяя или расширяя его возможности и организуя уровень абстракции для работы с бизнес-логикой, логикой представления, сервисом предоставляющим данные. Для такой организации работы клиента зачастую используют некоторые браузерные технологии, например, Java-апплеты, Silverlight, Flash, JavaScript.

В данном случае, в качестве средства реализации клиентского приложения, было принято решение выбрать язык JavaScript. Это связано с тем, что JavaScript:

* Интегрирован в браузер.
* Не требует значительных вычислительных ресурсов по сравнению с другими технологиям.
* Распространен.

Для достижения гибкой и богатой функциональности клиентского приложения на JavaScript, целесообразно использовать фреймворк. В нашем случае, было принято решение выбрать JavaScript-фреймворк AngularJS, так как изначально согласуется с принципами SPA и корректно работает в браузерах, которые перечислены в требованиях.

### API.

В качестве способа организации обмена данными с API, оптимальным решением было выбрать шаблон проектирования Representational State Transfer (REST), так как:

* Обладает универсальным интерфейсом.
* Позволяет использовать различные HTTP-клиенты.

### Формат обмена данными.

В качестве формата данных для обмена между клиентом и сервером, было решено использовать JSON, так как:

* Имеет встроенную поддержку в популярных браузерах, вышедших после 2009 года.
* Простой.
* Распространенный.
* Нативный формат JavaScript.

### Передача данных

Для обмена данными между клиентом и сервером, было решено использовать протокол HTTP, как общепринятый и доступный для запросов к API. А также объект XmlHttpRequest для использования технологии AJAX, которая имеет поддержку во множествах клиентских библиотек, в том числе и в Angular.

## Источник данных

Данные ВП необходимо систематизировано хранить и обрабатывать, в качестве чего используется база данных. Прежде чем выбрать систему управления базой данных (СУБД), следует рассмотреть предпосылки для выбора таковой.

Учитывая бизнес-правила, например, что при статусе расписания “утверждено”, его запрещено редактировать и особенности ВП, самая часто используемая операция при использовании СУБД - чтение данных, затем – добавление и изменение.

Также база данных должна просто настраиваться и легко поддерживаться администратором. Выбирая между различными СУБД, нужно учитывать то факт что наш проект не отличается особой на негo нагрузкой и сложностью решений.

На основании предпосылок и требований, было сделано заключение, что необходима СУБД, которая предлагает:

* Поддержку SQL.
* Простоту в использовании, что не нужно было обладать высокой квалификацией и опытом в работе с различными СУБД.

На основании сказанного, в качестве оптимального решения была выбрана **СУБД MySQL**.

## Серверная часть

**Язык программирования**

В качестве языка программирования для разработки серверной части был выбран Java по следующим причинам:

* Распространенность и документация.
* Языком программирования, Java, владеют большое количество программистов в компании-разработчика, что позволит развивать и поддерживать ВП.
* Java содержит множество удобных фрейворков для разработки серверной части приложения.

Java – это строго-типизированный объектно-ориентированный язык программирования, который был разработан компанией [Sun Microsystems](https://ru.wikipedia.org/wiki/Sun_Microsystems" \o "Sun Microsystems), после, был преобретен компанией Oracle. Приложения написанные на Java обычно транслируются в специальный байт-код, благодаря этому она могут работать на любой архитектуре компьютера с помощью виртуальной машины Java.

Для создания гибкой настройки и быстрой разработки серверной части ВП, целесообразно выбрать следующий фреймворк. В качестве такого решения был выбран Java-фреймворк Spring, так как имеет отличительные особенности:

* Высокая скорость обработки запросов, что повысит скорость работы ВП в целом на серверной части.
* Популярность и документация.
* Готовый набор функций для разработки SPA

# Реализация

## Клиентская часть

В качестве основы для клиентской части был выбран фреймворк AngularJS. AngularJS гибкий фреймворк и позволяет разработать клиентское приложение, основанное на типовых решениях проектирования. В результате клиентское приложение легко расширяется. Множество решений предоставляется “из коробки”, например, AngularJS позволяет автоматически проверять на корректность ввод данных в поля формы. Не конфликтует с другими библиотеками на JavaScript.

Основными объектами и компонентом клиентского приложения являются:

* Шаблон - HTML со специальными маркерами, - выражениями, для интерпретации кода JavaScript. Выражение - доступные переменные и функции из области видимости. Является компонентом.
* Представление - DOM, то, что видит пользователь в окне браузера.
* Маршрутизатор - обеспечивает работу с адресной строкой клиента.
* Контролер - частная бизнес-логика для представления.
* Сервис - общая бизнес-логика, независимая от представления.
* Модель - данные отраженные в представлении.
* Область видимости – контекст, в котором хранятся данные модели, доступные для контроллера и др. компонентов. Иногда именуется ViewModel (модель представления).

### Шаблон

Предоставляет разметку HTML с директивами, выражениями, фильтрами, формами. За счет директив, после обработки контроллером и моделью, добавляется поведение для представления. Такой механизм позволяет создать динамический DOM

Для визуального оформления был выбран фреймворк Angular Material. Это оказалось очень удобным, во-первых потому что он имеет нативную поддержку Angualar, во-вторых потому что соответствует всем стандартам Material Design.

#### Шаблон меню

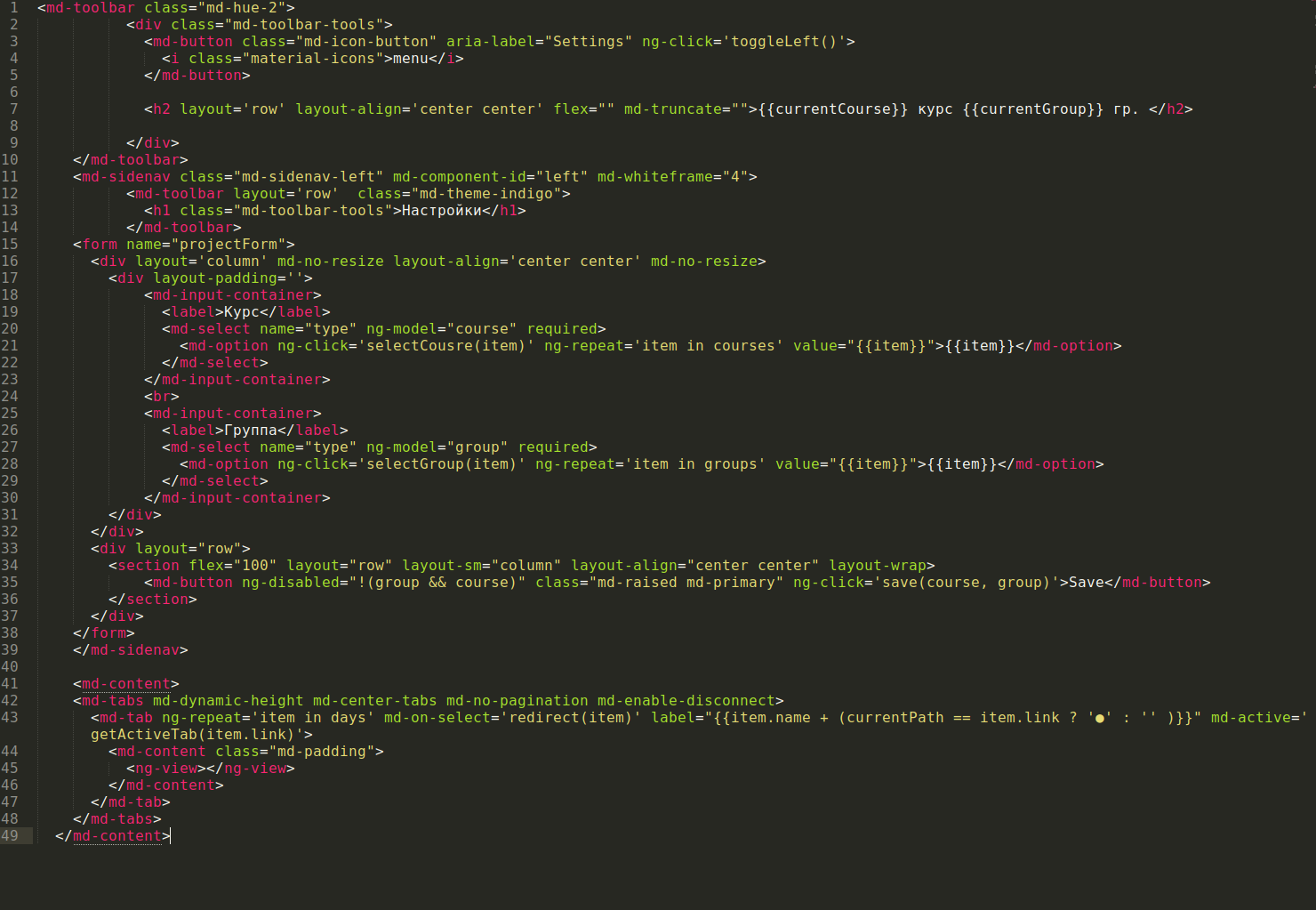


Рисунок 1

Данный шаблон представлен боковым меню, основным тулбаром и вкладками переключения дней.

#### Шаблон расписания

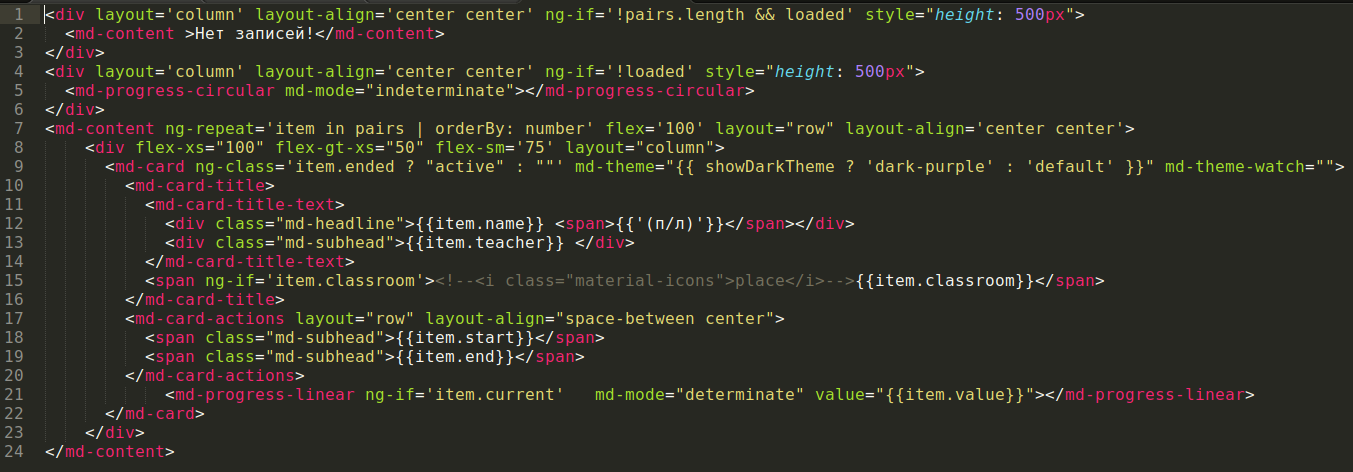


Рисунок 2

Данный шаблон представлен карточками расписания на конкретный день. В случае, если день на вкладке совпадает с календарным днем, на паре которая соответствует текущему времени отображается полоса прогресса, показывающая сколько времени прошло с начала пары. В случае, если пара уже закончилась она подсвечивается серым цветом. На остальные дни таких расширений нет.

### Представление

Отвечает за изменение HTML через DOM. Angular использует двухстороннее связывание(two-way data binding) в связи с чем изменение в модели происходят как в контроллере, так и на экране пользователя.

### Маршрутизатор

Наблюдает за состоянием адресной строки и загружает необходимые данные в зависимости от этого.

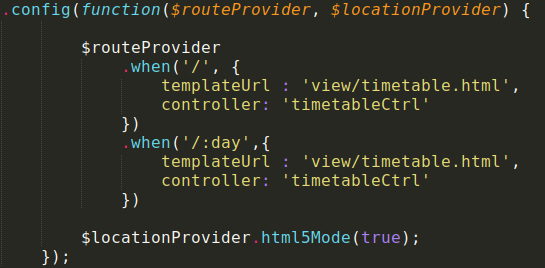


Рисунок 3

Маршрутизация представлена параметризировано, то есть в зависимости от пути отображается нужный день. Маршрутизация осуществляется при помощи директивы ngRoute.

### Контроллер

Отвечает за поведение представления или его отдельной части. Представление определяет контроллер. Изменяет состояние модели.

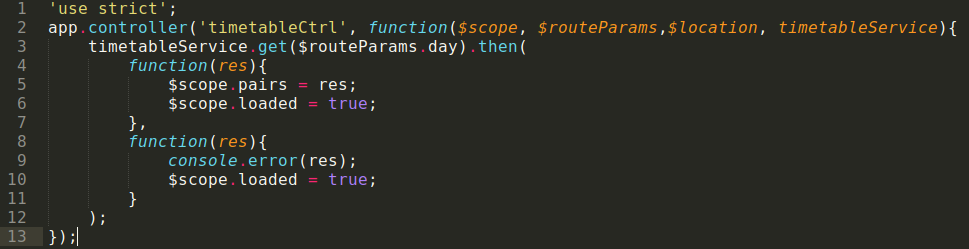
Контроллер расписания 

Рисунок 4

Контроллер расписания получает данные с сервера.

#### Контроллер меню

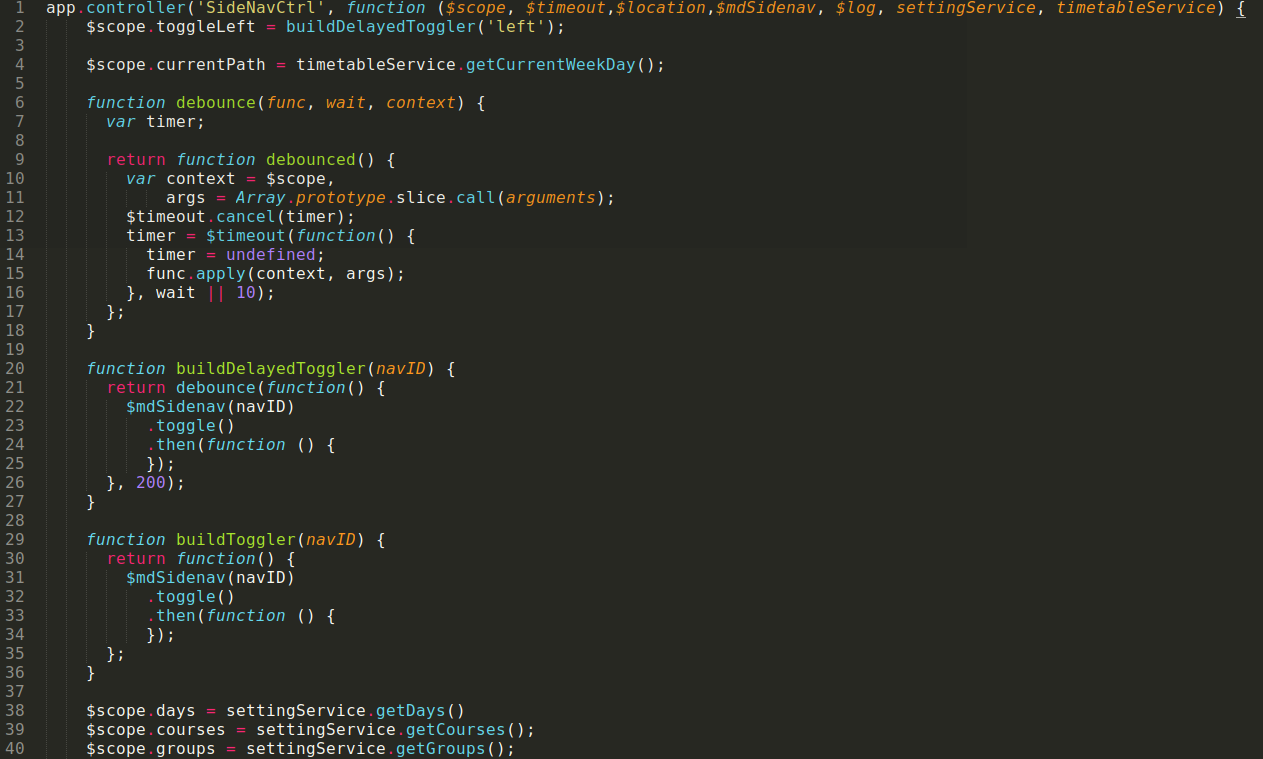


Рисунок 5

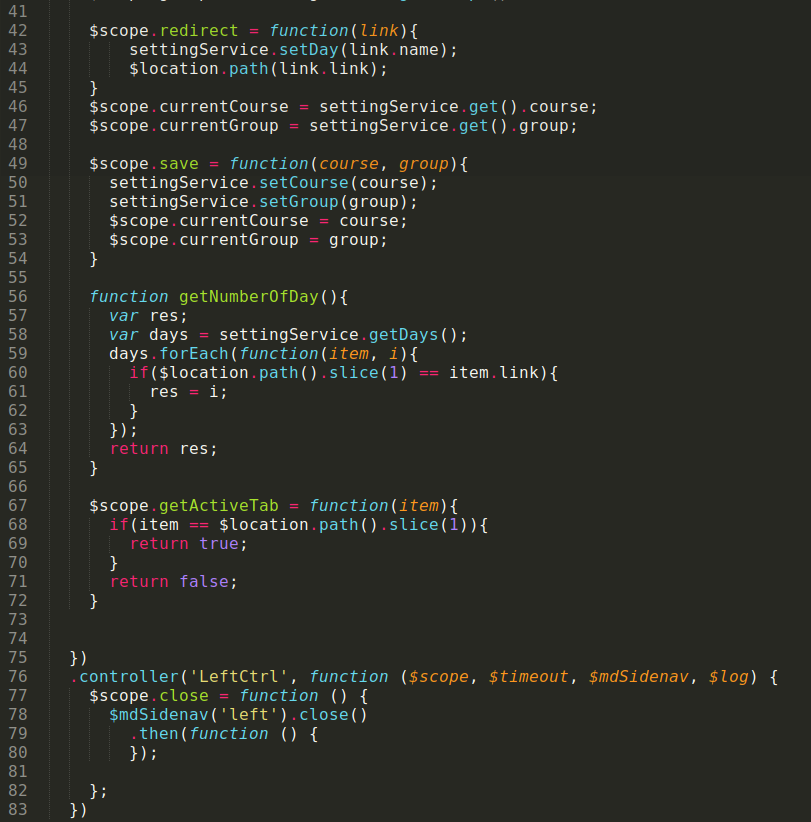


Рисунок 6

Контроллер бокового меню реализовывает более сложную логику, он управляет вкладками, открывает/закрывает меню, сохраняет данные о курсе/группе.

### Сервис

Отвечает за коммуникацию с сервером, в частности за обмен данными с API. Сервис работает с высокоуровневыми объектами браузера, например, такими как XmlHttpRequest для работы с AJAX. Основная логика взаимодействия с сервером и первичная обработка данных входит в задачу сервиса. Так же сервисы являются связующим звеном между контроллерами, сохраняя необходимую информацию.

#### Сервис настроек

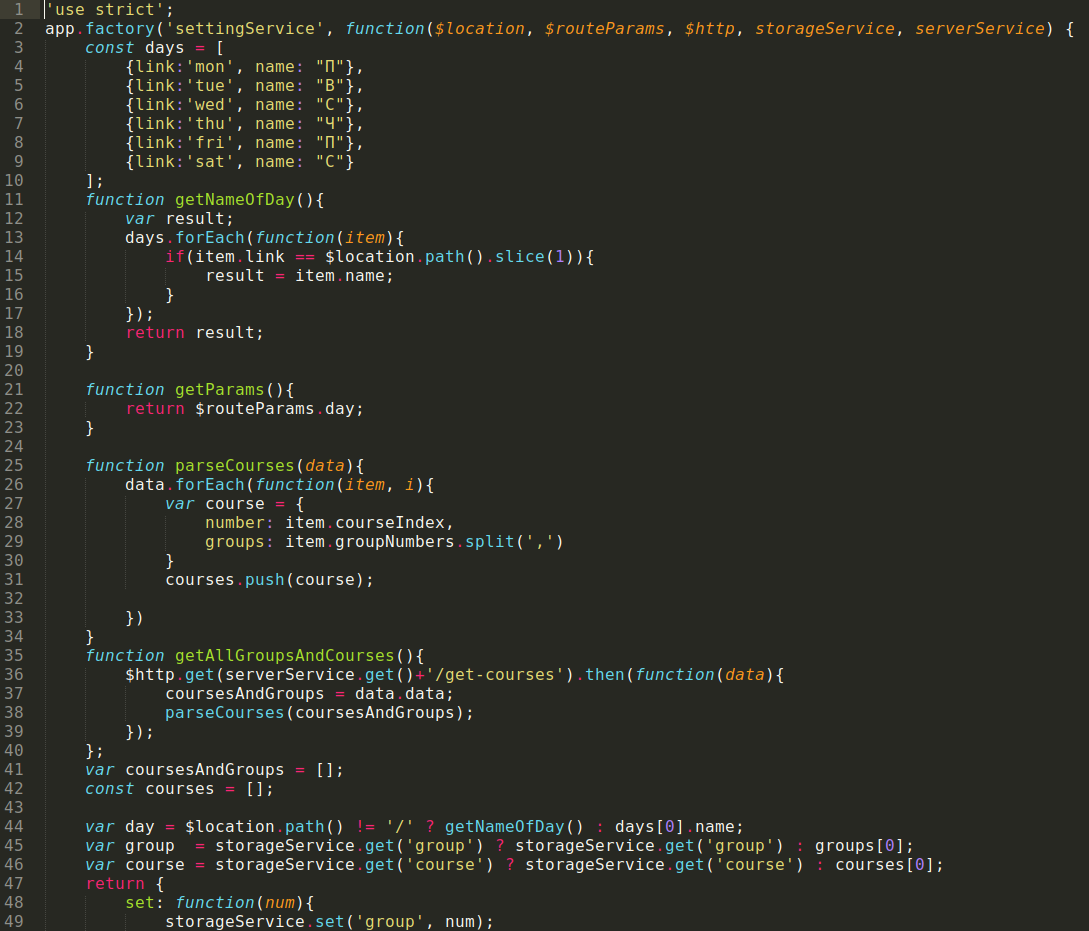


Рисунок 7



Рисунок 8

Сервис выполняет такие задачи, как:

* Сохраняет курс и группу
* Получает список курсов и групп с сервера
* Отдает список групп и курсов в контроллер
* Отдает текущую группу и курс в контроллер

#### Сервис времени пар

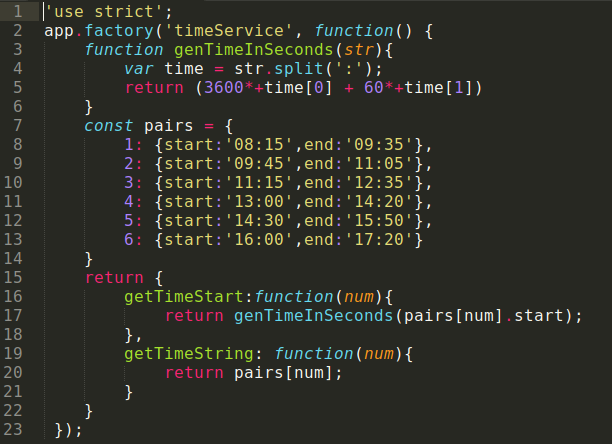


Рисунок 9

Этот сервис хранит информацию о номере пары и ее времени(так как эта информация неизменна, ее нет необходимости хранить на сервере)

Так же этот сервис парсит строку с временем в объект Дата, тем самым дает возможность вычислять разницу между началом пары и текущим временем.

#### Сервис работы с кэшем



Рисунок 10

Данный сервис предназначен для взаимодействия с кэшем браузера, чтобы хранить информацию о курсе и группе.

#### Сервис расписания

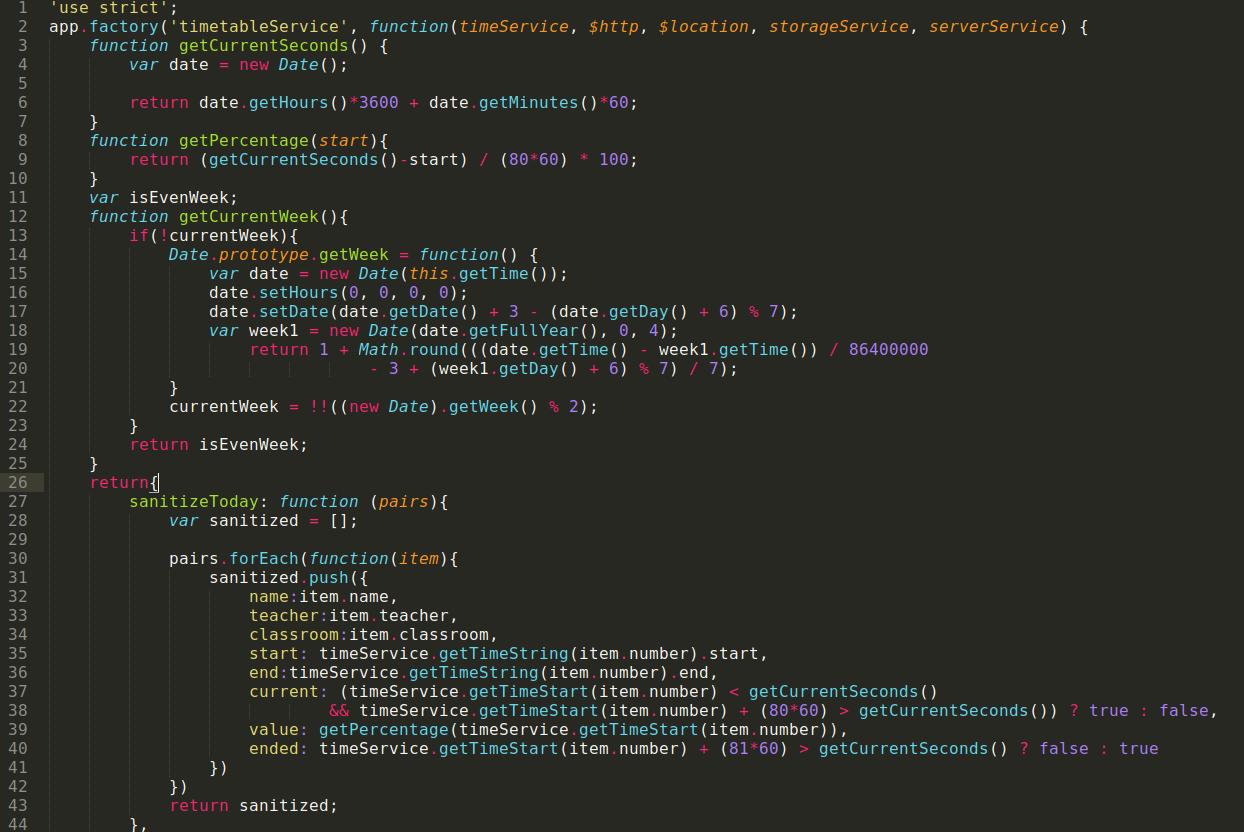


Рисунок 11

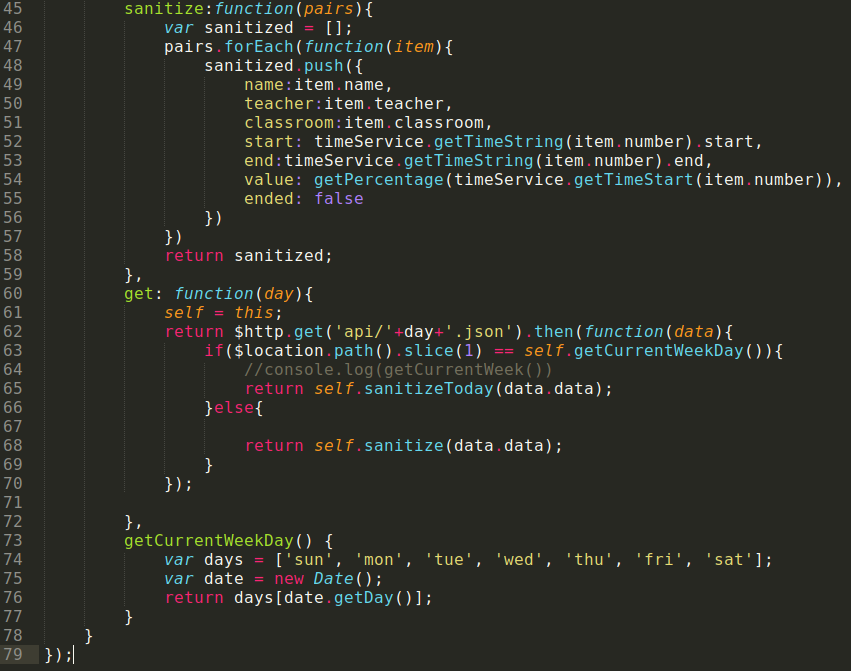


Рисунок 12

И последний, основной сервис. Он получает данные с сервера и обрабатывает их в соответствии с сегодняшней датой.

Так же он служит для вычисления номера текущей недели, для того, чтобы приложение могло отображать информацию о парах именно на данную неделю.

### Модель

Данные модели. Представенны в виде примитивных типов/объектов.

# Общее представление веб-приложения

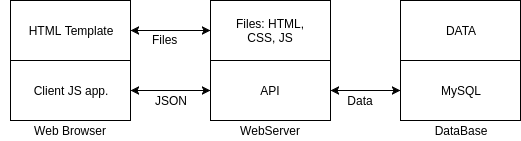


Рисунок 13

# ПРОЕКТИРОВАНИЕ

## Актеры

|  |  |  |
| --- | --- | --- |
| № | Актер | Описание |
| 1 | Гость | Пользователь может просмотреть расписание на текущий день, на другие дни для своей группы и курса. Изменять группу и курс. |

Таблица 1

## Варианты использования:

Гость. Первый визит.

|  |  |  |
| --- | --- | --- |
| Шаг |  | Гость |
| 1 | Цель | Посмотреть расписание на сегодня для своей группы |
| 2 | Инициирующее событие | Нажать кнопку "меню" |
| 3 | Предварительное состояние | Находиться в меню |
| 4 | Успешное конечное состояние | Выбрать свой курс и группу. Просмотреть расписание |

Таблица 2

Гость. Последующие визиты

|  |  |  |
| --- | --- | --- |
| Шаг |  | Гость |
| 1 | Цель | Посмотреть расписание на другой день(не текущий) для своей группы |
| 2 | Инициирующее событие | Переключить вкладку с текущим днем |
| 3 | Предварительное состояние | Находиться на главном экране веб-прилоежния |
| 4 | Успешное конечное состояние | Выбрать свой курс и группу. Просмотреть расписание |

Таблица 3

## Диаграмма вариантов использования

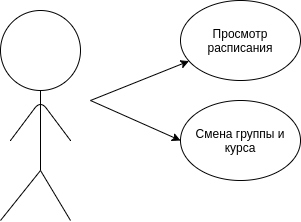
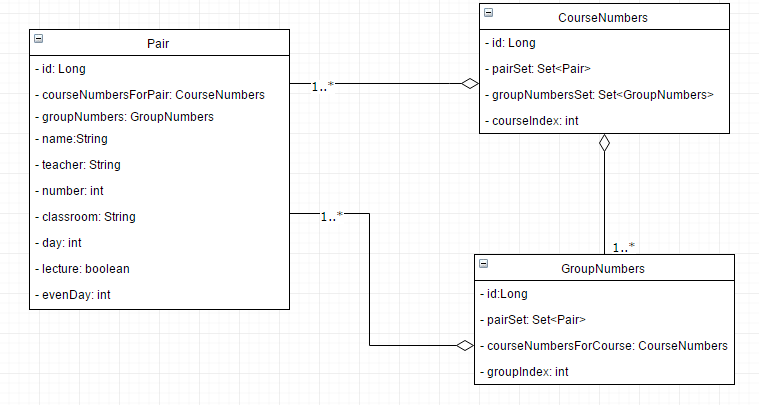


Рисунок 14

## Диаграмма классов сущностей



## Диаграмма классов логики

## Диаграмма последовательности

## Диаграмма деятельности

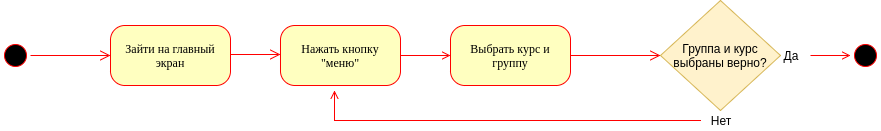


Рисунок 15

Просмотр расписания для заданной группы и курса

## Диаграмма базы данных

