министерство образования и науки РФ
московский государственный индустриальный университет
кафедра информационных систем и технологий

Руководители работы:
доцент, к.т.н. Куприянов Д.Ю.
ассистент Александров А.И.

Мазов Иван Александрович

«Разработка системы составления графика проведения консультаций и приёма задолженностей преподавателей МГИУ»

Курсовая работа по дисциплине
«Проектирование и разработка корпоративных информационных систем»
4-й курс, 7-й семестр

Москва 2011

Аннотация

Курсовая работа посвящена описанию дипломного проекта «Разработка системы составления графика проведения консультаций и приёма задолженностей преподавателей МГИУ». В данной работе описываются причины создания системы, актуальность. Описывается предполагаемая архитектура системы, обосновывается ее выбор. Так же показаны компоненты программного кода и примеры работы в виде скриншотов.

Оглавление

1. Введение

В современном мире сложно представить себе развитие нашего общества без использования интернета. Каждая коммерческая фирма или же научная организация старается создать свою страницу в интернете для того, чтобы пользователям было удобнее просматривать информацию об их деятельности. Наш ВУЗ не стал исключением из данного правила. В МГИУ большая он-лайн система, которая обеспечивает информационную поддержку различных факультетов. В нее включены такие модули, как: различные расписания, личный кабинет студента сдающего тестирования, электронная почта и пр. Но существует одна актуальная проблема — отсутствие удобной электронной версии расписания приема задолженностей. В данный момент расписание выкладывается в виде большого количества текстовых документов, что сильно затрудняет навигацию в них и нахождение необходимой информации. Всвязи с данными обстоятельствами было принято решение, о необходимости создания удобной, легко читаемой версии расписания, в которой несложно будет найти всю необходимую информацию, связаную с приемом задолженностей.

Цель создания системы — автоматизация процесса составления графика приема задолженностей, а так же обеспечения возможности удаленного просмотра расписания с помощью Web.
Назначение системы:

- Получение единого электронного хранилища с расписанием
- Повышение скорости создания расписания
- Возможность получения графика приема задолженностей в различных форматах

Представления пользовательских интерфейсов должны легко модифицироваться в рамках html страниц, система должна предусматривать интеграцию с уже работающими модулями системы управления ВУЗом. Так же, она должна сохранять работоспособность при пиковых нагрузках в 4 тысячи человек. Необходима реализация автоматического создания контрольных точек сохранения баз данных раз в день, что обусловлено актуальностью вносимой в систему информации.

В системе будет реализован следующий функционал:

- Изменение графика приема задолженностей с помощью визуального интерфейса
- Добавление различных мероприятий в сетку расписания приема задолженностей
- Выгрузка данных в xml для составления отчетности

Для того, чтобы описать структуру информационной системы и ее пользовательских интерфейсов необходимо:

- выбрать архитектуру;

- провести анализ функциональных особенностей информационной системы;
- спроектировать модели данных;
- разработать программное обеспечение;

2. Проектирование системы

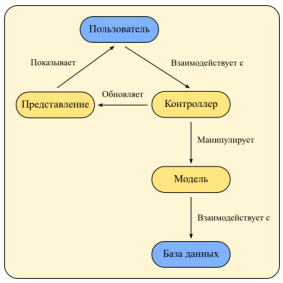
Выбор архитектуры

Одной из важнейших задач, встающих перед разработчиком при создании новой информационной системы — выбор ее архитектуры. Он обуславливается многими факторами. Основные из них: возможность легкой интеграции с уже существующими модулями, скорость работы и удобство использования, возможность легкой модификации и динамического изменения проекта, экономическая выгода.

В проекте будет использована концепция MVC — **Model-View-Controller**

- Models (Модель) — получает необходимые данные.
- Views (Представление) — показывает пользователю данные.
- Controllers (Контроллер) — управляет моделью и представлением.

Пример: допустим, пользователь заходит на нашу страницу. В этот момент Контроллер вызывает Модель, которая возвращает последние 10 записей. Далее данные передаются из Контроллера в Представление, которое выводит страницу пользователю.



Выбор архитектуры

Так как большая часть модулей системы управления ВУЗом написана на языке *Ruby*, выберем его как наш основной язык разработки. Из этого следует что веб интерфейсы мы будем разрабатывать с помощью *Ruby on Rails*(RoR).

RoR — веб ориентированная среда разработки, являющаяся фреймворком для построение приложений использующих базы данных, основанный на архитектуре

MVC, которая была описана выше. Она в нее входят динамичные AJAX-интерфейсы, обработка запросов и выдача данных в контроллерах. В последнее время Ruby on Rails все больше получает распространение, по причине удобства и скорости написания сайтов, использования современных технологий, коммерческой выгоды и открытых кодов. Rails отлично работает со многими веб-серверами и СУБД. В качестве веб-сервера рекомендуется использовать Apache или nginx с модулем Phusion Passenger. Rails также можно разворачивать используя Unicorn, Thin, Mongrel или FastCGI. В качестве СУБД можно использовать MySQL, PostgreSQL, SQLite, Oracle, SQL Server, DB2 или Firebird. Использовать Rails можно на практически любой операционной системе, однако для развертывания мы рекомендуем системы семейства *nix.

Так же в нашем Rails-приложении будет активно использоваться пакет языка *Javascript — Jquery*. Это одна из самощных мощных библиотек Javascript, которая фокусируется на взаимодействии HTML и Javascript. Библиотека jQuery снабжена удобным API по работе с *Ajax*.

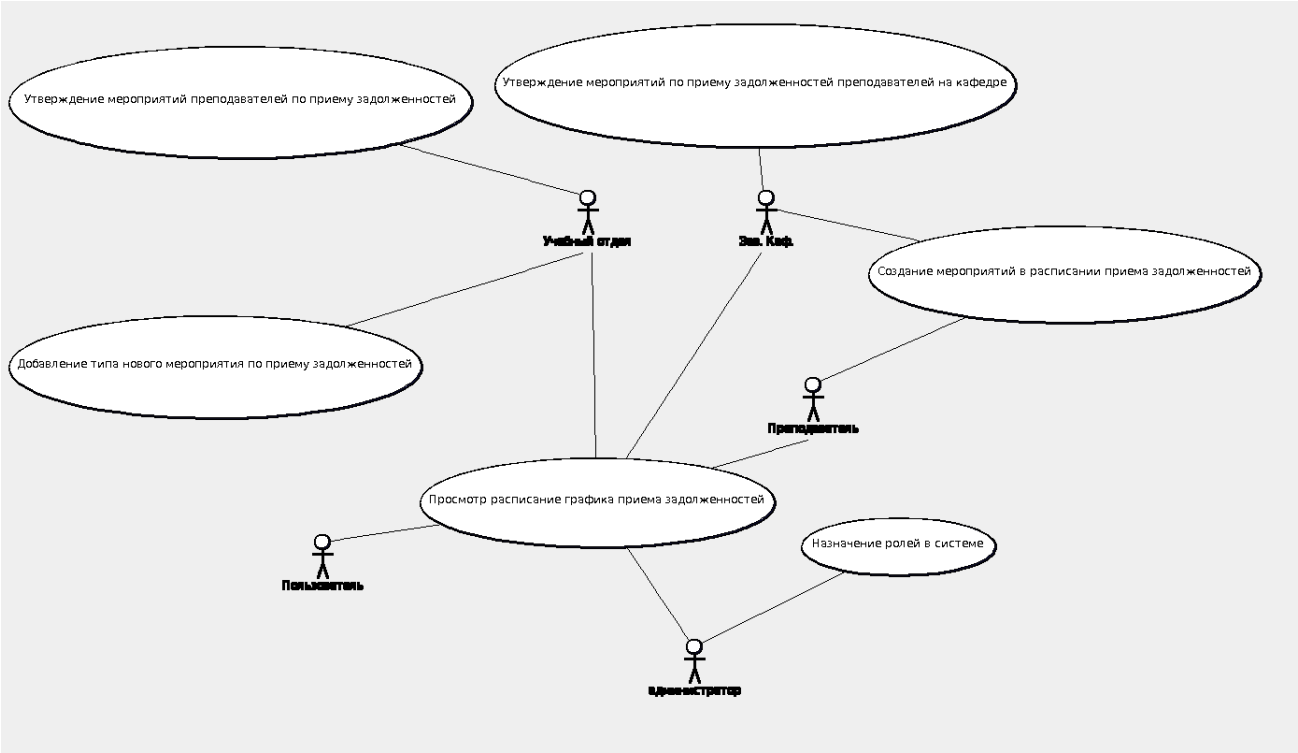
Анализ функциональных особенностей информационной системы

Приблизительная численность пользователей 7-9 тысяч человек, среди них существуют такие роли как:

- пользователи: 5-7 тысяч человек;
- преподаватели: 1,5 тысячи человек;
- заведующие кафедрой 150 человек;
- учебный отдел: 30 человек;
- администраторы: 1-3 человека;

Разберем более подробно ответственность ролей, для того чтобы лучше понимать требуемый от создаваемой системы функционал.

- Пользователь — роль которая позволяет просматривать сайт, совершать поиск расписания по группам, кафедрам факультетам и преподавателям.
- Преподаватель — составляет график приема задолженностей, добавляя новые мероприятия в сетку расписания. У преподавателя есть возможность добавления, удаления или редактирования мероприятия для одной или нескольких групп, по различным дисциплинам.
- Заведующий кафедрой - имеет те же возможности, что и преподаватель, плюс утверждает график приема задолженностей на своей кафедре.
- Учебный отдел — утверждает график приема задолженностей, составленный кафедрами или же отправляет его на доработку.
- Администратор — поддерживает работоспособность системы, назначает роли пользователям.



Проектирование моделей данных

Для того, чтобы грамотно организовать систему нужно четко себе представлять какие классы в ней будут. Так как система составления графика приема задолженностей проектируется для всего ВУЗа, нам потребуется разбиение на факультеты, кафедры, группы и предметы.

Ниже приведено описание классов.

- Факультет — отвечает за хранение, создание, удаление факультетов и их связи с кафедрами. Поля: Имя.

- Кафедра — отвечает за создание, удаление и редактирование кафедры, ее связь с преподавателями и дисциплинами. У каждой кафедры может быть много преподавателей и дисциплин. Поля: Имя, Номер.

- Преподаватель — отвечает за хранение преподавателей, их связи с кафедрами и предметами, которые они ведут. Поля,: Имя, Отчество, Фамилия и Ид_пользователя, который отвечает за связь с профилем в системе.

- Дисциплина — связывает дисциплины с кафедрами, на которых они предусмотрены. Поля: Имя, Сокращенное название, Ид_кафедры

- Предмет — дисциплина, которую конкретный преподаватель ведет у определенной группы. Имеет связь с кафедрой, дисциплиной, преподавателем и мероприятием. Поля: Ид_кафедры, Ид_дисциплины, Ид_преподавателя, Ид_группы.

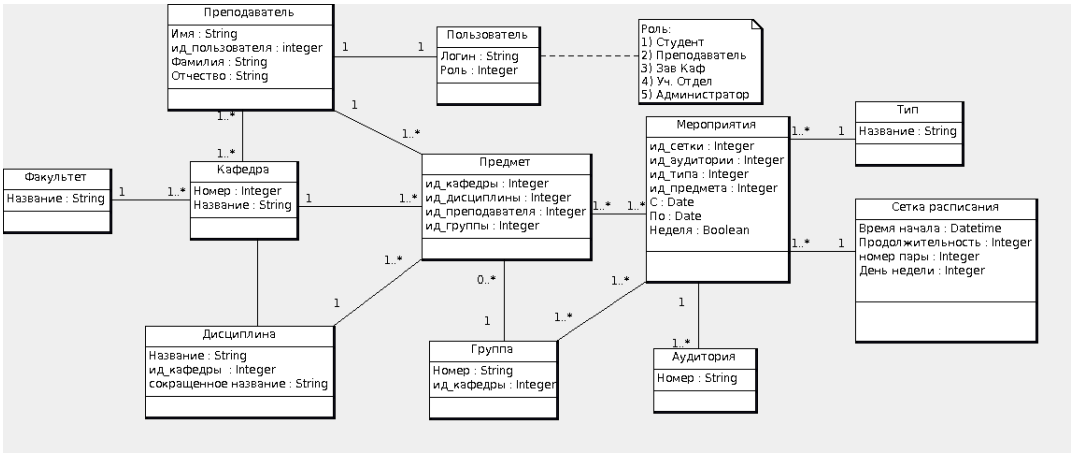
- Мероприятие — отвечает за создание нового события, которое будет проведено в определенную дату, неделю (числитель/знаменатель) и может длиться некоторый период. Поля: Ид_сетки_расписания, ид_аудитории, ид_типа, ид_предмета, С, По, Неделя (числитель/знаменатель).

- Аудитория — место проведения различных мероприятий. Поля: Номер

- Тип — тип мероприятия, например прием задолженностей или консультация. Поля: Название

- Сетка расписания отображает в таблице время начала мероприятия, его продолжительность, номер пары и день недели в который проводиться. Поля: Время_начала, Продолжительность, Номер_пары, День_недели.

Ниже приведена диаграмма классов:



3. Описание структуры информационной системы пользовательских интерфейсов

Использование генератора scaffold

Теперь перейдем к реализации разработанных классов Чтобы создать модель, вью и контролер воспользуемся генератором *scaffold*, и сгенерируем все части с его помощью

```
rails g scaffold Faculty name:string

rails g scaffold Chair name:string number:string

rails g scaffold Teacher name:string pathname:string
                         surname:string user_id:integer

rails g scaffold Discipline name:string reductname:string
                            chair_id:integer

rails g scaffold Subject chair_id:integer discipline_id:integer
                         teacher_id:integer group_id:integer

rails g scaffold Group number:string chair_id:integer

rails g scaffold Action web_id:integer lectroom_id:integer type_id:integer
                        subject_id:integer from:date to:date week:boolean

rails g scaffold Lectroom number:string

grails g scaffold Type name:string

rails g scaffold Web begintime:datetime lenght:integer
                     pair:integer day:integer
```

Теперь реализуем связи и отношения во всех моделях.

Вот примеры нескольких моделей со связями:
Предмет:

```
class Subject < ActiveRecord::Base
belongs_to :teacher
belongs_to :chair
belongs_to :discipline
belongs_to :group
has_and_belongs_to_many :actions

end
```

Мероприятие:

```
class Action < ActiveRecord::Base
belongs_to :type
```

```
belongs_to :web
has_many :lectrooms
has_and_belongs_to_many :subjects
has_and_belongs_to_many :groups

validates :from, :presence => true
end
```

А так же создадим миграции для добавления промежуточных таблиц, через которые делается отношение “многие ко многим”

```
rails g migration create_actions_subjects


class CreateActionsSubjects < ActiveRecord::Migration
  def self.up
    create_table :actions_subjects, :id=>false do |g|
      g.integer :action_id
      g.integer :subject_id
    end
  end

  def self.down
   drop_table :actions_subjects
  end
end


по аналогии

rails g migration create_chairs_disciplines
rails g migration create_chairs_teachers
rails g migration create_actions_groups
```

Представление

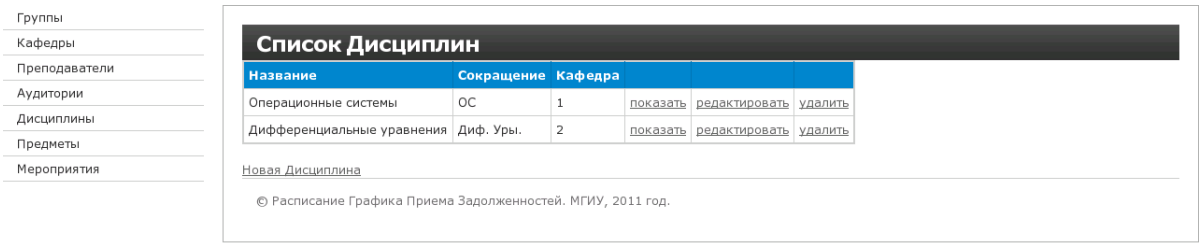
Добавим русификацию с помощью файла ru.yml; изменим представление, для корректного отображения информации. Для примера, возьмем представление дисциплин

изменим _form.html.erb

```
<%= form_for(@subject) do |f| %>
  <% if @subject.errors.any? %>
    <div id="error_explanation">
      <h2><%= pluralize(@subject.errors.count, "error") %>
            prohibited this subject from being saved:</h2>

      <ul>
      <% @subject.errors.full_messages.each do |msg| %>
        <li><%= msg %></li>
      <% end %>
      </ul>
    </div>
  <% end %>

  <div class="field">
    <%= f.label :chair_id %><br />
    <%= f.select :chair_id, Chair.all.map {|c| [c.name, c.id]} %>
  </div>
  <div class="field">
    <%= f.label :discipline_id %><br />
    <%= f.select :discipline_id, Discipline.all.map {|d| [d.name, d.id]} %>
  </div>
  <div class="field">
    <%= f.label :teacher_id %><br />
    <%= f.select :teacher_id, Teacher.all.map
    {|t| [t.name.to_s + ' '+t.surname.to_s, t.id]} %>
  </div>
  <div class="field">
    <%= f.label :group_id %><br />
    <%= f.select :group_id, Group.all.map {|g| [g.number, g.id]} %>
  </div>
  <div class="actions">
    <%= f.submit %>
  </div>
<% end %>
```


Группы
Кафедры
Преподаватели
Аудитории
Дисциплины
Предметы
Мероприятия
Список Дисциплин
Название
Сокращение
Кафедра
Операционные системы
ОС
1
показать
редактировать
удалить
Дифференциальные уравнения
Диф. Уры.
2
показать
редактировать
удалить
Новая Дисциплина
© Расписание Графика Приема Задолженностей. МГИУ, 2011 год.

4. Заключение

В данной работе была выбрана архитектура системы составления графика проведения консультаций и приёма задолженностей преподавателей МГИУ, был определен необходимый функционал, что облегчает дальнейшее написание проекта. Так же были частично реализованы модели.

Список литературы и интернет-ресурсов

[1] С.М. Львовский. *Набор и вёрстка в системе LaTeX, 3-е изд., испр. и доп.* — М., МЦНМО, 2003. Доступны исходные тексты этой книги.

[2] `http://ru.wikipedia.org/wiki/LaTeX` — Википедия (свободная энциклопедия) о системе LaTeX.

[3] `http://www.sbras.ru/win/docs/TeX/LaTex2e/docs_koi.html` — Различная документация по системе LaTeX.

[4] `http://edgeguides.rubyonrails.org/` — Официальный сайт Ruby On Rails.

[5] `http://railscasts.com/` Видео уроки работы с Ruby On Rails.

[6] `http://apidock.com/rails` Электронная документация по Ruby On Rails