МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Московский государственный индустриальный университет»

(ФГБОУ ВПО «МГИУ»)

кафедра информационных систем и технологий

КУРСОВАЯ РАБОТА

по дисциплине «Базы данных»

на тему «Модификация эталонного проекта «Аэропорт»: реализация просмотра переходов по сайту и просмотра статистики блоков ссылок»

Группа	121132
Студент	Д.В. Седых
Руководитель работы к.т.н., доцент	В.Ю. Радыгин

Москва 2014

Аннотация

В курсовой работе требуется модифицировать эталонный проект «Аэропорт», который состоит из реализации веб-сайта, отражающего деятельность аэропорта, написанной с использованием фреймворка Ruby on Rails. В данном проекте отражается деятельность аэропорта. Модификация отражает функции представляемые пользователю "operator" для отслеживания переходов пользователей по сайту и организация собираемых данных для удобства отслеживания.

Содержание

1. Введение

В данной курсовой работе необходимо модифицировать эталонный проект «Аэропорт». Согласно условию, в модификации нуждаются файлы проекта, а так же добавление собственных файлов для получения требуемого результата. Для этого необходимы знания языков Ruby, PostgreSQL, Ruby on Rails и понимание архитектуры проекта «Аэропорт».
Эталонный проект «WebAirport» разработан с применением шаблона проектирования MVC. $Model–View–Controller$ — это схема для использования нескольких шаблонов проектирования, предполагающие разделение данных приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента: модель, представление и контроллер – таким образом, что модификация каждого компонента может быть осуществлена независимо.

Постановка задачи

Модифицируйте эталонный проект «Аэропорт» добавив новые модели «Переход по сайту», «Ссылка», «Блок ссылок». Модель «Ссылка» должна содержать поля: контроллер, акшион и описание ссылки. Модель «Переход по сайту» содержит информацию о всех, нажимаемых в проекте «Аэропорт» пользователями, ссылках. Она включает поля: параметры ссылки, пользователь, нажавший ссылку, время и дата создания. Модель «Переход по сайту» должна быть связана отношением «многие к одному» с пользователями и ссылками. Модель «Блок ссылок» объединяет несколько ссылок в блок.

Реализуйте для каждого блока ссылок статистику, показывающую информацию о количестве ссылок в данном блоке ссылок и информацию о количестве переходов по ссылкам, находящимся в данном блоке ссылок.

2. Разработка структуры базы данных

Рассмотрим концепцию базы данных и включенных в нее моделей.
На рисунке 1 показана диаграмма классов, отражающая виды взаимосвязей и кратность между моделями эталонного проекта.
Более подробная диаграмма классов, представленная на рисунке 2 поможет понять необходимые для добавления в модификацию модели.

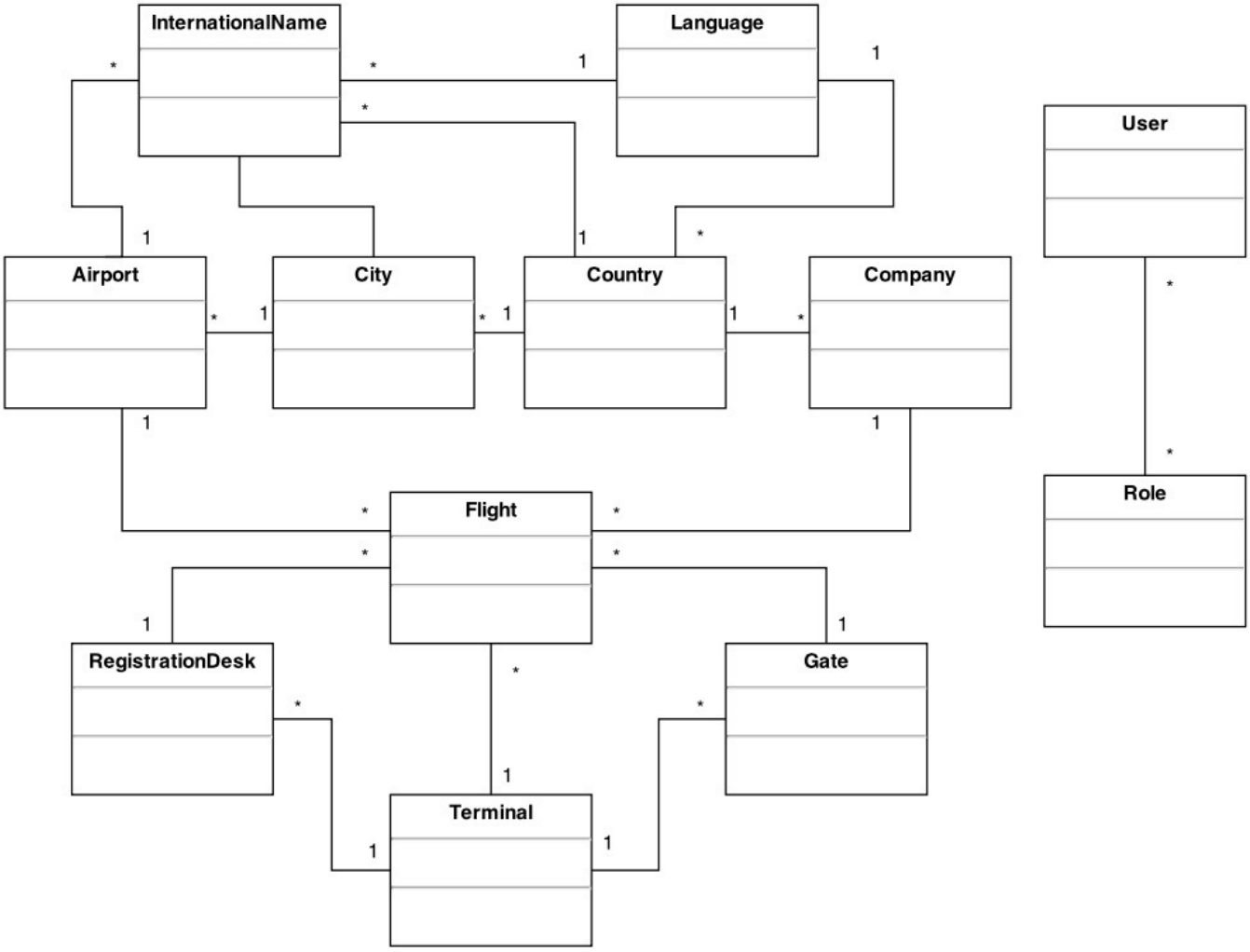

Рис. 1. Диаграмма классов эталонного проекта «Аэропорт»

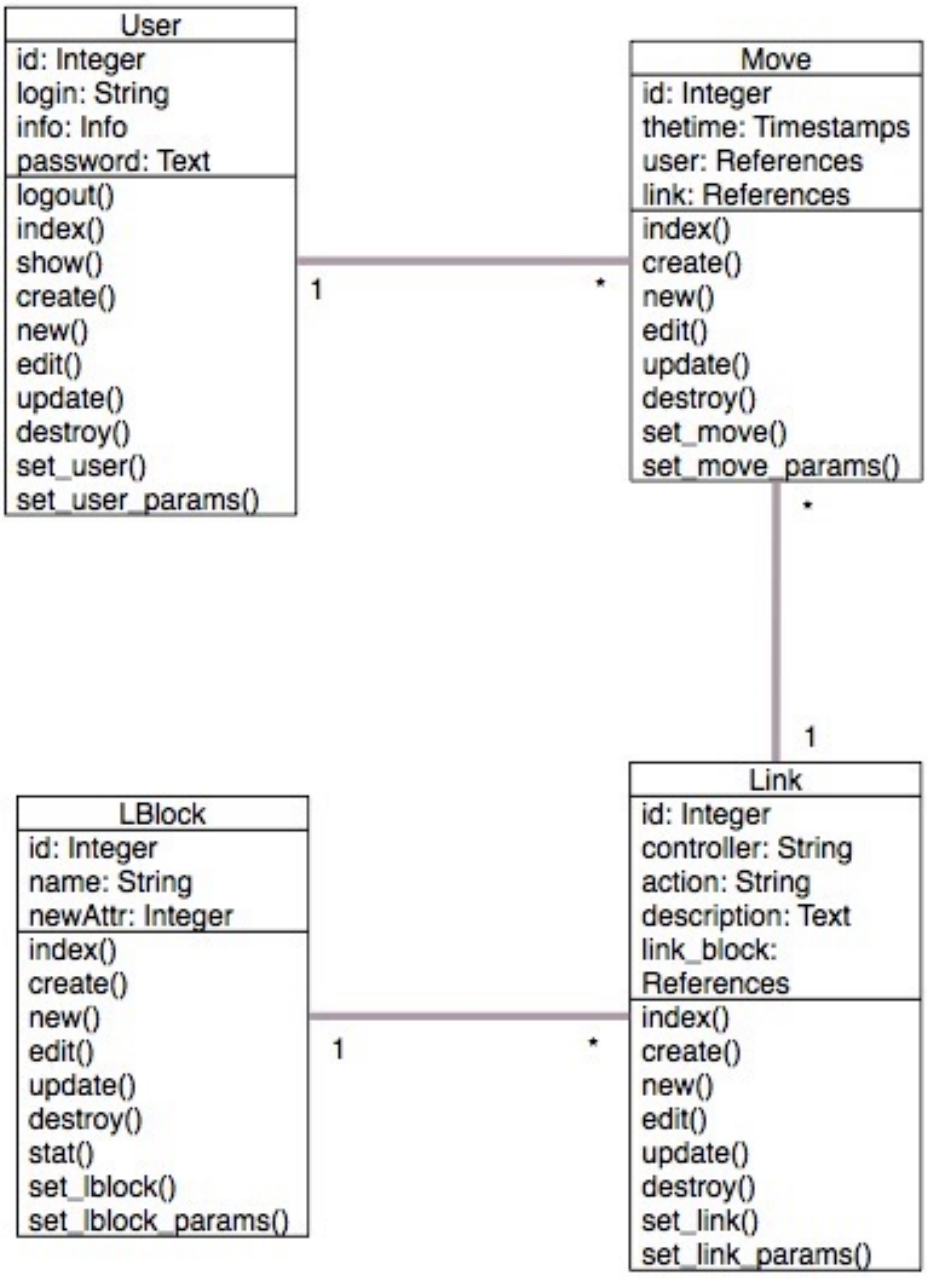

Рис. 2. Диаграмма классов, добавленных в эталонный проект «Аэропорт»

Модели

- «Flight» является главное моделью. В данной модели содержатся базовая информация о полётах, то есть дата и время вылета или прилета, так же направление, код рейса, так же статус рейса, то есть прилет или вылет. Эта модель связана с пятью другими моделями, то есть с «Company», «Airport», «Gate», «RegistrationalDesk», «Terminal».
- «Company» - это модель, которая используется для создания авиакомпаний. В данной модели содержатся информация об авиакомпаниях, то есть их названия и страны к которым эти авиакомпании принадлежат. Модель имеет связь с двумя моделями: «Country», «Flight».

- «Airport» - это модель, которая содержит информацию об аэропортах. Данная модель позволяет добавить новый аэропорт, соотнести определенный аэропорт с определенной страной и городом. Модель аэропорт имеет связь со следующими моделями: «City», «Flight», «Language».
- «Role» - это модель, которая описывает возможные роли, которые могут быть у того или иного пользователя.
- «Country» - модель, описывающая страну, то есть куда или откуда летит самолет по тому или иному рейсу. Данная модель имеет три поля: название, официальный язык и код. Связана данная модель с четырьмя другими: «City», «Language», «Company», «InternationalName».
- «City» - модель, содержащая в себе описание города (куда или откуда летит самолет по тому или иному рейсу. Данная модель связана с тремя другими моделями: «Country», «InternaionalName», «Airport».
- «Terminal» - модель описывающая терминалы аэропорта.
- «Language» - модель с помощью которой создаются иностранные языки, для отображения информации. Данная модель связана с двумя другими моделями: «Country», «InternationalName».
- «Gate» - модель, которая описывает выходы на посадку.
- «InternationalName» - модель, с помощью которой происходит перевод терминов на иностранные языки. Данная модель свзяна с четырьмя другими моделями: «Language», «Airport», «City», «Country».
- «User» - это модель, которая описывает пользователей данного проекта.

Миграции

В соответствии с заданием, были сгенерированны следующие шаблоны моделей. Примеры команд для генерации представлены ниже:

1. rails generate model Lblock name:string
Шаблон модели «Lblock» («Блок ссылок») с полем name (название).

2. rails generate model Link controller:string description:text lblock:references
Шфблон модели «Link» («Ссылка») с полями controller (контроллер), action (акшион), description (описание ссылки) и lblock (ссылка на модель «Lblock»).

3. rails generate model Move thetime:timestamps user:references link:references
Шаблон модели «Move» («Переход по сайту») с полями thetime (время и дата перехода), user (ссылка на модель «User») и «Link» (ссылка на модель «Lblock»).

Пример файла миграции для шаблона «Переход по сайту» с полями отвечающими за целостность (NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY, CHECK) приведен ниже:

```
class CreateMoves < ActiveRecord::Migration
  def up
    create_table :moves do |t|
      t.text :parametrs, :null => false

          t.timestamp :thetime, :null => false

      t.references :user, index: true
      t.references :link, :null => false, index: true

      t.timestamps
    end
        execute "
                ALTER TABLE moves ADD FOREIGN KEY(user_id) REFERENCES users(id)
                "
        execute "
                ALTER TABLE moves ADD FOREIGN KEY(link_id) REFERENCES links(id)
                "
  end

        def down
                drop_table :moves
        end

end
```

Добавленные модели

После проведения маграций мы получили три таблицы в базе данных и три файла с моделями, необходимыми нам для решения поставленной задачи.

Модель «Переход по сайту» содержит информацию о всех ссылках, нажимаемых в проекте «Аэропорт» пользователями.Она включает поля: ссылка (параметрынажимаемой ссылки) , пользователь, нажавший ссылку, время и дата создания.
Эта модель связана отношением «многие к одному» с пользователями и ссылками, поэтому были добавлены ассоциации, характеризующие это отношение:

- belongs_to :user
- belongs_to :link

Также были добавлены валидации, отвечающие за валидность данных сохраненных в нашей базе данных:

validates :user_id, :presence => true
validates :link_id, :presence => true

Модель «Ссылка» описывает конкретную ссылку, по которой возможен переход в рамках проекта. Эта модель содержит поля: контроллер ссылки, акшион ссылки и описание ссылки.
«Ссылка» связана отношением «один ко многим» с моделью «Переход по сайту» и отношением «многие к одному» с блоками ссылок.

Валидации позволяют узнать, что заполнение ссылки не корректно без атрибутов контроллер, акшион, описание.

```
class Link < ActiveRecord::Base

        belongs_to :lblock
        has_many :moves, :dependent => :destroy

        validates :controller, :presence => true, :uniqueness => {:scope => [:action]}
        validates :action, :presence => true
        validates :description, :presence => true
    #validates :lblock#, :presence => true
end
```

Модель «Блок ссылок» объединяет несколько ссылок в блок.
Одному экземпляру класса «Блок ссылок» соответствует ноль или более экземпляров класса «Ссылка» .

3. Реализация интерфейса
Описание контроллеров

Стандратное веб приложение, как правило, сосоит из контроллеров, представлений и моделей.Контроллер может быть устроен как основной, который получает все запросы и вызывает другие контроллеры для выполнения действий в зависимости от ситуаций.

Контроллер выступает посредником между моделями и представленийми(отображениями данных на сайте). Благодаря контроллеру существует возможность отображать данные пользователю и сохранять или обновлять данные, которые вводил или введет пользователь в те или иные модели.
Ниже представлены примеры некоторых методов из контроллера Link. Метод create вызывается при создании нового обьекта:

```
def create
  @link = Link.new(link_params)
  lb = Lblock.where(:id => link_params['lblock_id'].to_i).first
  @link.lblock = lb

  respond_to do |format|
    if @link.save
      format.html { redirect_to :action => :show, :id => @link.id, notice: 'Ссылка успешно создана' }
    else
      format.html { render action: 'new' }
    end
  end
end
```

Метод update вызывается при редактировании объекта:

```
  def update
    @link.attributes=link_params
    lb = Lblock.where(:id => link_params['lblock_id'].to_i).first
    @link.lblock = lb
    respond_to do |format|
      if @link.save
        format.html { redirect_to :action => :show, :id => @link.id, notice: 'Link успешно отредактирован.' }
      else
        format.html { render action: 'edit' }
      end
    end
  end
```

Код в представлении(View) должен отображать требуемую информацию. В папке */app/ views* для моделей «Links» и «Lblocks» и «Moves» были созданы соответствующие файлы: edit.html.erb, index.html.erb, new.html.erb, show.html.erb. В каждом из перечисленных файлов содержится определенный код, отвечающий за отображение данных при определенном выполнении действий пользователя. Собственно, в этих файлах и указывается, какие поля отображать и их названия.
Также был добавлен контроллер Stats для предоставления соответствующих данных в форму представления View.

```
class StatsController < ApplicationController

  # GET /lblocks/index
  def index

    @lblocks = Lblock.includes(links: [:moves]).references(:links, :moves).load

  end
end
```

В форме View, в свою очередь, представлена обрабатывающая функция, которая выводит корректное представление статистики.

Описание представлений

Контроллер выполняет обслуживание запросов в Rails. View обрабатывет данные для представление их пользователю.
Для каждой модели в папке /app/views существует папка с шаблонами, которые используются непосредственно контроллером. Для новых моделей были созданы аналогичные папки. В этих папках содержатся представленные далее файлы:

- edit.html.erb
- _form.html.erb
- index.html.erb
- new.html.erb
- show.html.erb

В edit.html.erb осуществляется редактирование нужного обьекта.

В _form.html.erb представлена типовая форма (шаблог) для заполнения полей обьекта. Для каждой модели прописаны свои поля.

Файл index.html.erb используется в контроллерах по умолчанию. С его помощью выводятся таблицы в интерфейсе.

Добавить новый обьект можно благодаря new.html.erb, который тоже, в свою очередь, ссылается на _form.html.erb.

Когда требуется посмотреть подробную информацию об обьекте, используется show.html.erb. В этом файле прописаны поля, отображение которых будет происходить непосредственно в интерфейсе.
Далее подробно описывается создание представлений для ссылок, блоков ссылок, переходов по сайту и статистики.

Ссылки

Блоки ссылок
Переходы по сайту

Статистика

В файле app/view/stats/index.html.erb описана форма представления статистики по количеству кликов по ссылкам блока в web-браузере. Задается таблица при помощи конструкции <tr> </tr>, далее происходит разделение на столбцы при помощи конструкции <td> </td>. Первый столбец- это столбец с названиями блоков ссылок. Во втором столбце отображается количество ссылок в каждом из блоков ссылок. В третьем столбце представлено количество переходов по ссылкам каждого блока ссылок.
Для отображения и осуществления подстчета написан следующий код:

```
<td><%= block.name %></td>
<td><%= block.links.size %></td>
<% movs = 0%>
<% block.links.each do |l| %>
    <% movs += l.moves.size %>
<% end %>
<td><%= movs %></td>
<% movs = 0%>
```

Примеры работы интерфейса

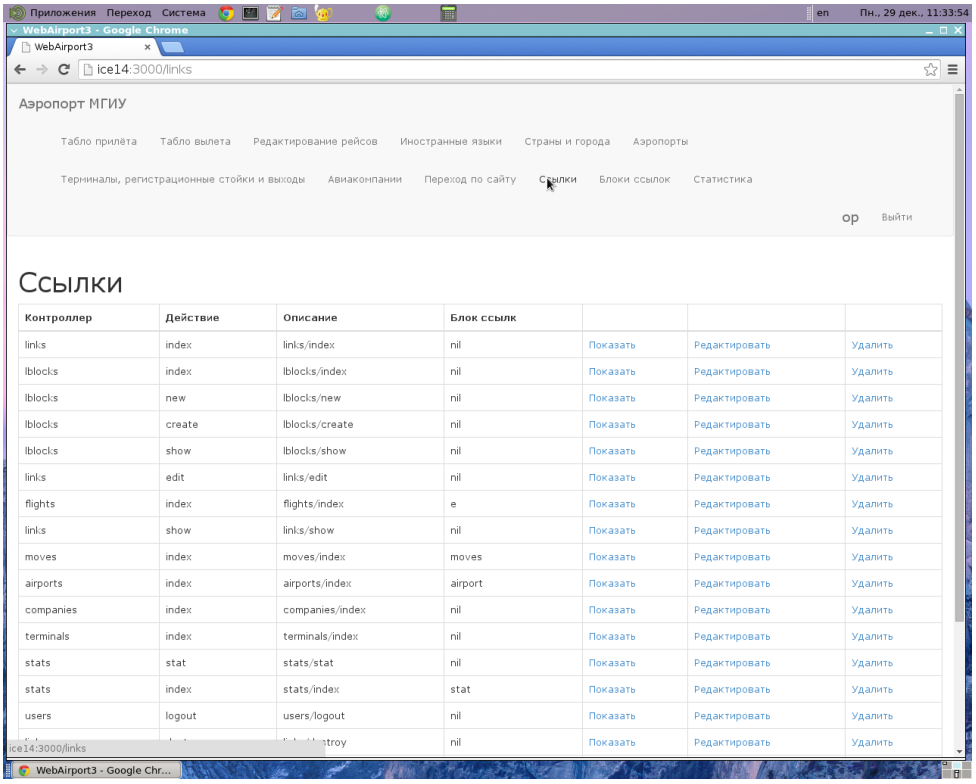

Рис. 3. Визуальное представление для модели «Ссылки»

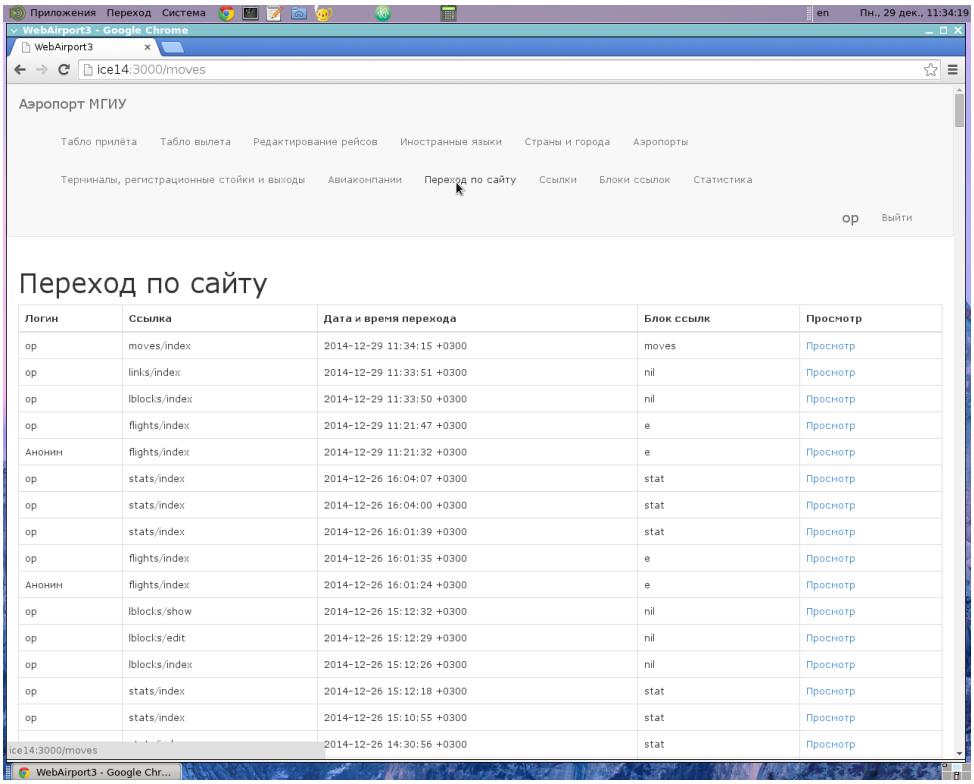

Рис. 4. Визуальное представление для модели «Переход по сайту»

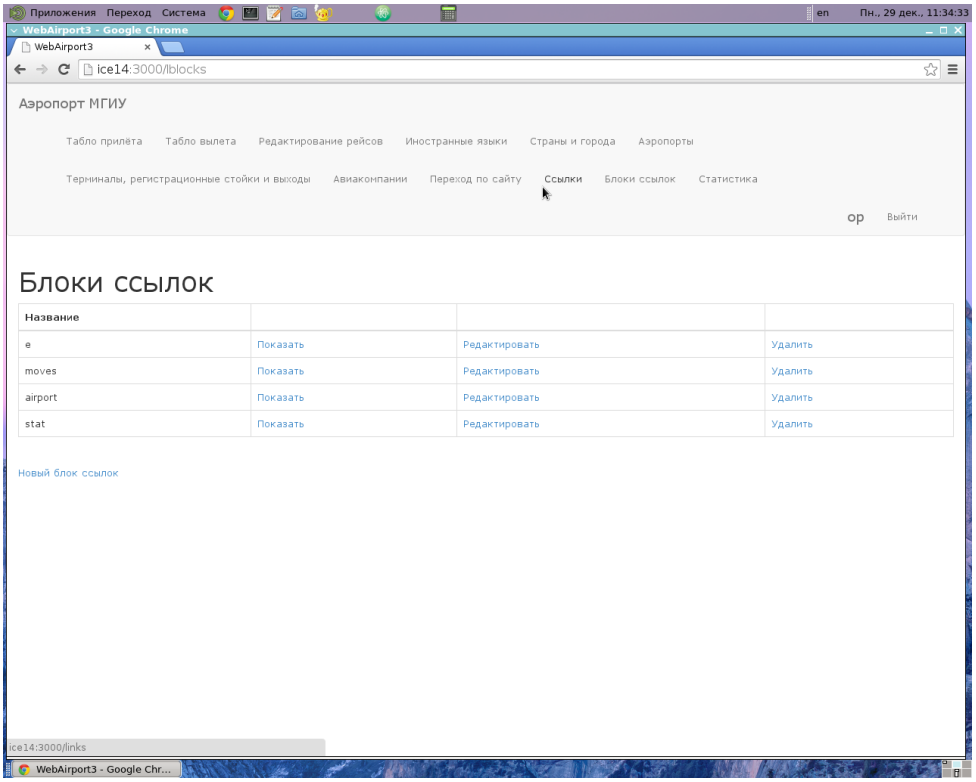

Рис. 5. Визуальное представление для модели «Блоки ссылок»

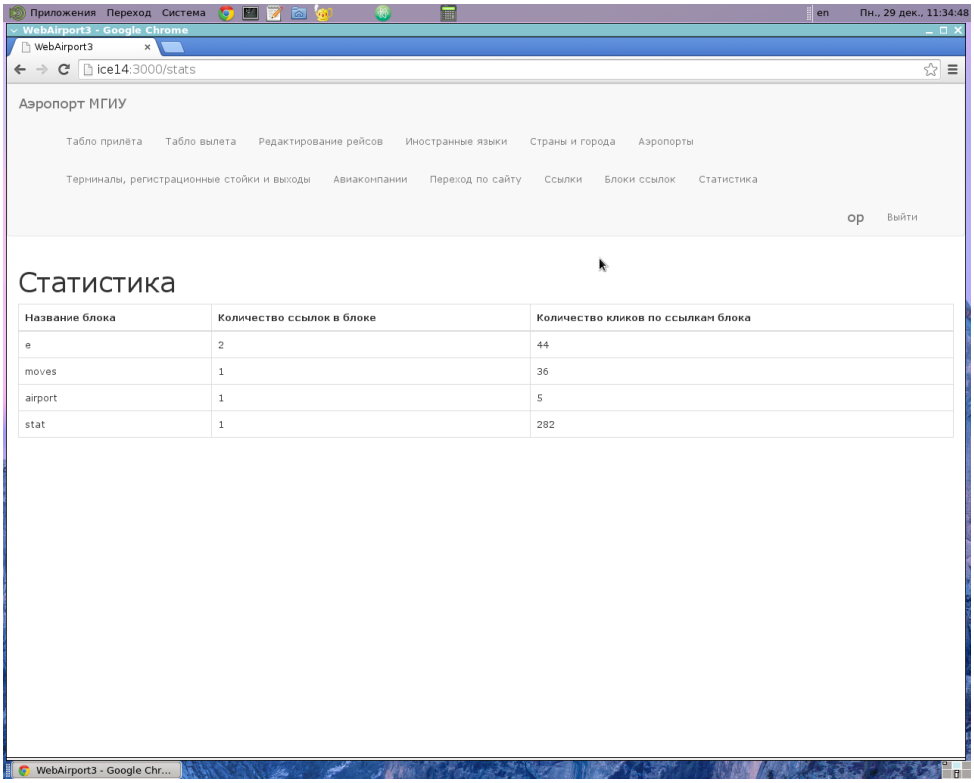

Рис. 6. Визуальное представление для модели «Статистика»

На странице «Статистика» показана сводка по каждой ссылке. Здесь можно посмотреть количество ссылок в блоке и количество переходов по ним.

6. Заключение

Результатом проделанной работы является модифицированный проект «Аэропорт», с дополнениями в виде трех новых моделей: «Переход по сайту», «Ссылка», «Блок ссылок». Также возможность подсчета и просмотра статистики реализованна путем добавления контроллера и view-файла.

При помощи данного проекта возможно выполнение задачи мониторинга посещений сайта различными пользователями, такими, как, операторы, администраторы и иных зарегестрированных администратором пользователей. Так же реализовывается возможность просмотра статистики по посещениям сайта анонимными пользователями.

Для составления данной пояснительной записки использовалась система TEX. Используемые материалы для помощи были взяты из книг С.М.Львовсково[3] и Д.Е.Кнутта[4].

5. Список литературы и интернет-ресурсов

[1] http://ru.wikipedia.org/wiki/Ruby — Википедия о языке Ruby.
[2] С.М. Львовский. Набор и вёрстка в системе LATEX, 3-е изд., испр. и доп. — М., МЦНМО, 2004. Доступны исходные тексты этой книги.
[3] http://rusrails.ru/ — Описание Ruby on Rails.
[4] D. E. Knuth. The TEXbook. — Addison-Wesley, 1984. Русский перевод: Дональд Е. Кнут. Все про TEX. — Протвино, РДТЕХ, 1993.

ПРИЛОЖЕНИЕ

Контроллер ссылок

```
class LinksController < ApplicationController
  before_action :set_link, only: [:show, :edit, :update, :destroy]

  # GET /links/index
  def index
    @links = Link.includes(:lblock).load
  end

  # GET /links/show/1
  def show
  end

  # GET /links/new
  def new
    @link = Link.new
    if params.has_key?('lblock_id')
      lb = Lblock.where(:id => params['lblock_id'].to_i).first
      @link.lblock = lb
    end
  end

  # GET /links/edit/1
  def edit
  end

  # POST /links/create
  def create
    @link = Link.new(link_params)
    lb = Lblock.where(:id => link_params['lblock_id'].to_i).first
    @link.lblock = lb

    respond_to do |format|
      if @link.save
        format.html { redirect_to :action => :show, :id => @link.id, notice: 'Ссылка успешно создана' }
      else
        format.html { render action: 'new' }
      end
    end
  end

  # POST /links/update
  def update
    @link.attributes=link_params
    lb = Lblock.where(:id => link_params['lblock_id'].to_i).first
    @link.lblock = lb
    respond_to do |format|
      if @link.save
        format.html { redirect_to :action => :show, :id => @link.id, notice: 'Link успешно отредактирован.' }
      else
        format.html { render action: 'edit' }
      end
    end
  end

  # GET /links/destroy/1
  def destroy
    @link.destroy
    respond_to do |format|
      format.html do
        if params.has_key?('link_action')
          redirect_to :action => :index, :controller => :lblock
```

```
      else
        redirect_to :action => :index
      end
    end
  end
end

private
  # Use callbacks to share common setup or constraints between actions.
  def set_link
    @link = Link.includes(:lblock).where(:id => params[:id].to_i).first
  end

  # Never trust parameters from the scary internet, only allow the white list through.
  def link_params
    params.require(:link).permit(:controller, :action, :description, :lblock_id)
  end

  def permit?(aname)
    return true if aname == 'show'
    return true if !@user.nil? and @user.roles.include?(OPERATOR_ROLE)
    return false
  end
end
```

Контроллер блока ссылок

```
class LblocksController < ApplicationController
  before_action :set_lblock, only: [:show, :edit, :update, :destroy]

  # GET /lblocks/index
  def index
    @lblocks = Lblock.all
  end

  # GET /lblocks/show/1
  def show
  end

  # GET /lblocks/new
  def new
    @lblock = Lblock.new
  end

  # GET /lblocks/edit/1
  def edit
  end

  # POST /lblocks/create
  def create
    @lblock = Lblock.new(lblock_params)

    respond_to do |format|
      if @lblock.save
        format.html { redirect_to :action => :show, :id => @lblock.id, notice: 'Блок ссылок успешно создан' }
      else
        format.html { render action: 'new' }
      end
    end
  end

  # POST /lblocks/update
  def update
    @lblock.attributes = lblock_params

    respond_to do |format|
      if @lblock.save
```

```
      format.html { redirect_to :action => :show, :id => @lblock.id, notice: 'Блок ссылок успешно обновлен.' }
    else
      format.html { render action: 'edit' }
    end
  end
 end

 # GET /lblocks/destroy/1
 def destroy
  @lblock.destroy
  respond_to do |format|
   format.html { redirect_to :action => :index }
  end
 end

 private
  # Use callbacks to share common setup or constraints between actions.
  def set_lblock
   @lblock = Lblock.find(params[:id])

  end

  # Never trust parameters from the scary internet, only allow the white list through.
  def lblock_params
   params.require(:lblock).permit(:name)
  end

  def permit?(aname)
   return true if aname == 'show'
   return true if !@user.nil? and @user.roles.include?(OPERATOR_ROLE)
   return false
  end
end
```

Контроллер переходов по сайту

```
class MovesController < ApplicationController
 before_action :set_move, only: [:show, :edit, :update, :destroy]

 # GET /moves/index
 def index
  @moves = Move.includes(:user, :link).order(:thetime => :desc).load
 end

 # GET /moves/show/1
 def show
 end

 # GET /moves/new
 def new
  @move = Move.new
 end

 # GET /moves/edit/1
 def edit
 end

 # POST /moves/create
 def create
  @move = Move.new(move_params)
  user = User.where(:id => move_params['user_id']).first
  link = Link.where(:id => move_params['link_id']).first
  @move.user = user
  @move.link = link

  respond_to do |format|
```

```
    if @move.save
      format.html { redirect_to :action => :show, :id => @move.id, notice: 'Переход по сайту успешно создан.' }
    else
      format.html { render action: 'new' }
    end
  end
 end

 # POST /moves/update
 def update
  @move.attributes = move_params
  user = User.where(:id => move_params['user_id']).first
  link = Link.where(:id => move_params['link_id']).first
  @move.user = user
  @move.link = link

  respond_to do |format|
    if @move.save
      format.html { redirect_to :action => :show, :id => @ouser.id, notice: 'Пользователь успешно обновлён.' }
    else
      format.html { render action: 'edit' }
    end
  end
 end

 # GET /moves/destroy/1
 def destroy
  @move.destroy
  respond_to do |format|
    format.html { redirect_to :action => :index }
  end
 end

 private
  # Use callbacks to share common setup or constraints between actions.
  def set_move
    @move = Move.includes(:user).where(:id => params[:id].to_i).first
    @move = Move.includes(:link).where(:id => params[:id].to_i).first
  end

  # Never trust parameters from the scary internet, only allow the white list through.
  def move_params
    params.require(:move).permit(:thetime, :user, :link)
  end

  def permit?(aname)
    return true if aname == 'show'
    return true if !@user.nil? and (@user.roles.include?(OPERATOR_ROLE)|| @user.roles.include?(ADMIN_ROLE))
    return false
  end
end
```

Контроллер статистики

```
class StatsController < ApplicationController

 # GET /lblocks/index
 def index
   @lblocks = Lblock.includes(links: [:moves]).references(:links, :moves).load
 end
end
```

Модифицированная часть application_controller

```
def moving()
   Rails.logger.info("CR user=#{@user.inspect} action=#{action_name} controller=#{controller_name}")
```

```
        link = Link.where(:controller => controller_name, :action => action_name, :description => "#{controller_name}/
#{action_name}").first_or_create!

    Move.new(:user_id => (@user ? @user.id : nil), :link_id => link.id, :thetime => DateTime.now, :parametrs =>
params.inspect).save!

  end
```

Helper для move

```
module MovesHelper
  def select_user(name, selected = nil)
    select_tag(name, options_for_select(
        User.order('login').load.map{ |x| ["#{x.name}", x.id] } + [['', nil]],
        [selected]))
  end
```

Helper для link

```
  def select_link(name, selected = nil)
    select_tag(name, options_for_select(
        Link.order('controller', 'action').load.map{ |x| ["#{x.controller}, #{x.action}", x.id] } + [['', nil]],
        [selected]))
  end
end
```