

**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ
ФЕДЕРАЦИИ
ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
КЕМЕРОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Математический факультет**

Кафедра ЮНЕСКО по Новым информационным технологиям

ДИПЛОМНАЯ РАБОТА

«Автоматизированная система регистрации на семинары в КРИПКиПРО»

студента пятого курса, М-065 группы

Яковлева Александра Яковлевича

**Специальность 010503 - «Математическое обеспечение и администрирование
информационных систем»**

Научный руководитель:

Кандидат педагогических наук, доцент

Л. Е. Шмакова

Работа допущена к защите:

« ____ » _____ 2011г.

Зав. кафедрой

д-р физ.-мат. наук, профессор

К. Е. Афанасьев

Работа защищена:

« ____ » _____ 2011г.

с оценкой _____

Председатель ГАК

Члены ГАК:

Кемерово 2011

Оглавление

Введение	4
Глава 1 Терминология	5
1.1 Основные понятия	5
Глава 2 Постановка задачи	16
2.1 Структура и архитектура ИС	16
2.2 Структурированное текстовое описание процесса	16
2.3 Требования к системе	18
2.4 Обзор сходных информационных систем	19
2.5 Диаграммы	20
2.6 Используемые программные средства	20
Глава 3 Реализация поставленной задачи	25
3.1 Совместимые клиентские программы	25
3.2 Краткий обзор API Kohana	25
3.3 Авторизация пользователя	34
3.4 Регистрация пользователей	35
3.5 Управление мероприятиями	36
3.6 Выставление ограничений	37
3.7 Обзор расписания	38
3.8 Регистрация посетителей	39
3.9 Обзор посетителей	39
3.10 Стил ь программирования	40
Заключение	41

Литература	43
Приложения	45

Введение

В последнее время актуальна регистрация на мероприятия в режиме онлайн.

На рынке существует много систем, оптимизирующих планирование различных мероприятий, но в силу специфики многие организации разрабатывают собственные.

КРИПКиПРО осуществляет целенаправленную работу по повышению квалификации, профессиональной переподготовке, послевузовскому профессиональному образованию и непрерывному образованию педагогических кадров Кемеровской области. Ежегодно в институте обучается около 8 тысяч работников региональной системы образования на факультетах повышения квалификации и профессиональной переподготовки, и более 12 тысяч работников образования охвачены формами непрерывного профессионального образования (семинары, мастер-классы, конференции и др.).

Расписание курсов повышения квалификации, занятий на факультете профессиональной переподготовки, проведения семинаров, конференций формируется в мае на весь учебный год. Для проведения всех мероприятий используется аудиторный фонд института. Поэтому актуальной является разработка инструмента, позволяющего оптимизировать планирование расписания семинаров с учетом других занятий.

Глава 1. Терминология

§1.1. Основные понятия

§1.1.1. API

Application Programming Interface, *англ.* интерфейс программирования приложений – набор доступных извне классов, методов, функций и процедур, предоставляемый подключаемой библиотекой или программным продуктом. Используется для программирования приложений или надстроек, [1].

§1.1.2. Фреймворк

Фреймворк – это программная абстракция; коллекция библиотек, предоставляющих определённый API. Универсальный функционал может быть избирательно заменён пользовательским кодом, чтобы получить прикладную программу.

Фреймворки в корне отличаются от обычных библиотек тем, что фреймворк не подключается к приложению – приложение подключается к фреймворку. Фреймворк диктует модель разработки, потоки управления, иногда – терминологию и поведение программы.

Фреймворк имеет собственное поведение по умолчанию. Это не серия «заглушек», а полезное простое поведение, обычно подходящее для малых проектов.

Фреймворк расширяем; часть его кода может быть переопределена пользователем. Расширяемость фреймворка позволяет писать специальные функции.

Обычно код фреймворка недоступен для изменений; пользователь может расширять ядро, но ему не разрешается его модифицировать. Мягкие ограничения не рекомендуют модифицировать ядро, чтобы сохранить стабильность и предсказуемость всего приложения.

Фреймворк для написания интернет-приложений – это фреймворк, спроектированный для поддержки разработки динамических сайтов, веб-приложений и веб-сервисов. Он стремится автоматизировать написание рутинных и часто используемых в веб-приложениях операций – таких, как доступ к базе данных, управление пользователями и использование шаблонов. Фреймворки стараются поддерживать повторное использование кода.

Такие фреймворки обычно построены в соответствии с концепцией MVC. Известными примерами являются CodeIgniter (PHP), Yii (PHP), Kohana (PHP), Catalyst (Perl), Google Web Toolkit (Python), Ruby On Rails (Ruby), Weblocks (Lisp).

Исторически некоторые интернет-фреймворки получили название систем управления контентом (CMS). Это высокоуровневые программы, которые стремятся полностью избавить пользователя от необходимости замены кода в своём ядре и программирования как такового. Обычно их ядро реализует базовое окружение, а основной функционал разнесен по различным модулям. Популярными фреймворками такого типа являются системы Drupal, Wordpress, Joomla и ModX.

§1.1.3. Javascript

Деревом DOM можно управлять из сценариев на странице. Обычно сценарии пишутся на языке JavaScript. Сценарии – это маленькие программы, которые можно включить в код страницы, используя элемент `script` или при помощи атрибутов перехвата событий.

Листинг 1.1. Пример использования Javascript

```
1 <form name="main">
  <a>Result:</a> <output name="result"></output>
3 <script>
  document.forms.main.elements.result.value = 'Hello
    World';
5 var a = document.links[0];
```

```

    a.href = 'sample.html';
7   a.protocol = 'https';
    a.setAttribute('href', 'http://example.com/');
9  </script>
</form>

```

Строка 4 задаёт значение элемента **output** равным «Hello World»

На строке 5 создаётся переменная-объект, в которой хранится первая гиперссылка документа. Далее задаётся атрибут href, протокол HTTPS (защищённый HTTP) и ещё раз меняется href, уже при помощи метода.

Язык Javascript слабо типизирован, а его возможности жёстко ограничены правилами безопасности браузера. Так, Javascript на клиентской стороне не имеет доступа к файлам пользователя.

Поддержка Javascript различается между браузерами: так, сценарии Internet Explorer пишутся на аналогичном языке JScript, который различается в модели документа. Например, в Internet Explorer сценариям доступны модули ActiveX, [2].

§1.1.4. Веб-сервер

Веб-сервер, интернет-сервер, сервер интернет-приложений – это программа, которая запущена на серверном компьютере, служащем хостом для веб-сайта, [3]. Её основное назначение – это отдавать веб-страницы, что означает ждать запросов от веб-браузеров и отвечать на них, посылая требуемые данные.

Самым популярным и знаменитым веб-сервером является Apache от фонда Apache.

§1.1.5. CGI

Common Gateway Interface (CGI), *англ.* общий интерфейс шлюза – это простой интерфейс для запуска внешних программ, приложений или шлюзов под управлением сервера информации в платформо-независимой манере. На настоящий момент поддерживаемые серверы информации – это серверы HTTP. [4]

Интерфейс использовался всемирной сетью Интернет с 1993го года.

CGI разработан таким образом, чтобы программист мог использовать любой язык программирования, который может работать со стандартными устройствами ввода/вывода. Такими возможностями обладают даже скрипты для стандартных командных интерпретаторов операционных систем, поэтому в тех случаях, когда нет нужды в сложной функциональности, могут использоваться даже такие простые командные скрипты.

Серьёзной проблемой CGI на загруженных сайтах является то, что при каждом обращении веб-сервер запускает новый экземпляр программы. Таким образом – особенно при использовании интерпретируемых языков – достаточно большое количество одновременных запросов заставят сервер запустить достаточно много экземпляров программы, и свободная оперативная память закончится.

§1.1.6. PHP

PHP – это широко используемый универсальный язык программирования сценариев, который особенно подходит для разработки в Интернете и может быть вставлен в HTML. [5]

При этом клиент получает результат работы скрипта, но не может увидеть код программы.

Официальный интерпретатор PHP является de facto стандартом языка; спецификации выпущено не было.

Обычно для запуска интернет-приложений на PHP используется веб-сервер Apache с установленным модулем `mod_php`; тем не менее, возможно использовать другие серверы приложений, обращаясь к PHP по интерфейсу CGI. PHP может применяться для написания клиентских приложений, в том числе с графическим интерфейсом пользователя, но он не получил популярности в этой роли.

§1.1.7. Браузер

Интернет-браузер – это клиентская программа для получения, обработки, предоставления информации и навигации по Всемирной Сети (InterNet). [6]

Существует много различных программ, отличающихся в основном обработкой и визуализацией информации. В результате постоянных отклонений от стандартов в различных реализациях большинство браузеров поддерживают стандарт de facto. Так, поддержка тега `<marquee>` была введена в Microsoft Internet Explorer 5.0 и, хотя тег не включён ни в один действующий стандарт, он поддерживается также всеми новыми версиями Internet Explorer, Mozilla Firefox, Google Chrome и Opera.

§1.1.8. HTTP

Hypertext Transfer Protocol, *англ.* протокол передачи гипертекста. [7]

Протокол программного уровня для распределённых, совместных, гипермедиа информационных систем. Универсальный протокол без состояний, который может быть использован для многих задач кроме гипертекста: например, для серверов имён и распределённых систем управления объектами. Он позволяет строить системы независимо от передаваемых данных.

HTTP использовался информацией Интернета с 1990го года.

§1.1.9. XML

Extensible Markup Language, *сокр.* XML, *англ.* расширяемый язык разметки, описывает класс объектов данных, называется документами XML и частично описывает поведение компьютерных программ, которые обрабатывают их. XML – это усечённая форма SGML, стандартного универсального языка разметки [ISO 8879]. Документы XML разработаны так, что являются правильными документами SGML. [8]

XML был разработан Рабочей Группой XML в 1996-м году. Основными целями при разработке XML были:

1. XML будет просто использовать через Интернет,
2. XML будет поддерживать широкий спектр приложений,
3. XML будет совместим с SGML,
4. Парсер XML будет прост для написания,
5. Необязательных возможностей в XML быть не должно,
6. Документы XML должны читаться человеком и относительно понятны,
7. Дизайн XML должен быть прост и быстр для подготовки,
8. Дизайн XML должен быть формальным и точным,
9. XML документы должны просто создаваться.

§1.1.10. CSS

Cascading Style Sheets, *сокр.* CSS, *англ.* Каскадные Таблицы Стилей – это язык описания презентации веб-страниц, включая цвета, разметку и шрифты. Он позволяет адаптировать страницы к различным типам устройств, таким как большие мониторы, маленькие мониторы или принтеры. [9]

CSS независим от HTML и может быть использован с любым языком разметки на основе XML.

Разделение страницы на разметку и стиль упрощает управление сайтом, позволяет объединять стили разных страниц. Содержимое становится независимым от оформления.

§1.1.11. JSON

JavaScript Object Notation, *англ.* Объектная Нотация Javascript – лёгкий формат обмена данными. [10] Он понятен для человека и прост для разбора машиной. JSON построен на подмножестве третьей редакции Javascript (1999), а потому прекрасно совместим с JavaScript. Встроенные средства разбора и кодирования JSON есть в интерпретаторе PHP 5.

§1.1.12. AJAX

Asynchronous JavaScript and XML, *англ.* Асинхронный JavaScript и XML – группа не связанных между собой техник, используемых на стороне клиента для создания интерактивных интернет-программ, [11]. Термин был введён в 2005-м году в статье Джессе Джеймса Гарретта, [12].

Идея AJAX в том, чтобы расширить роль браузера от ультратонкого клиента, который содержит минимум логики и почти не выполняет вычислений, исключительно представляя информацию пользователю, до толстого клиента, который может самостоятельно работать с данными, полученными от сервера. [3]

В традиционной модели «тонкий клиент-сервер» клиент должен выполнить запрос (GET или POST) каждый раз для изменения данных. В течение выполнения запроса работа с веб-страницей невозможна, так как после выполнения запроса загружается новая страница.

Модель AJAX предлагает асинхронное выполнение запросов, не прерывающее работу с данными во время их изменения. Каждый запрос HTTP заменяется запросом к Javascript; всё, что не требует соединения с сервером — проверка данных, редактирование данных, возможно навигация — проводится на стороне клиента.

Слово «AJAX» было введено как короткий аналог фразы «Асинхронный JavaScript, CSS, DOM и XMLHttpRequest».

Важнейшей технологией из группы AJAX является вызов Javascript XMLHttpRequest. Он был реализован в различных браузерах в 2000-2005м годах; в версиях Internet Explorer 5, 5.5 и 6 его роль выполнял модуль ActiveX библиотеки MSXML. В настоящее время ведётся работа над черновиком XMLHttpRequest уровня 2.

По соображениям безопасности запросы к другим доменам с данного запрещены; это ограничение обходится разными путями, самым простым из которых является HTTP заголовок `Access-Control-Allow-Origin`.

§1.1.13. DOM

Document Object Model, *англ.* объектная модель документа.

Согласно Консорциуму Мировой Паутины (World Wide Web Consortium), DOM — это платформи- и языконезависимый интерфейс, который позволяет программам динамически получать доступ и обновлять содержимое, структуру и стиль документов. Обработка документа может быть продолжена, и результаты обработки могут быть внедрены обратно в просматриваемую страницу. [13]

Объектная модель документа — это стандарт W3C для доступа к HTML и XML документам. Она задаёт объекты и свойства всех элементов документа, а также методы (интерфейс) управления ими.

§1.1.14. HMVC

Hierarchical Model-View-Controller, *англ.* иерархические Модель-Вид-Контроллер. [3]

Каждая тройка модель-вид-контроллер называется триадой. Каждая триада независима и отвечает за один объект, описываемый Моделью.

Модель и только Модель имеет знания о природе объекта, например, где хранятся данные и как их необходимо проверять. Контроллер отвечает за логику работы с объектом. Он вычисляет значения переменных, инициирует изменение данных, передаёт данные в Вид. Вид отвечает за представление и только за представление данных; в Виде не производится никаких вычислений.

Каждая триада совершенно независима и может выполняться при отсутствии любой другой. Все запросы к триадам идут через их контроллеры. Любая часть системы может выполнить запрос к любой триаде. Так, вид триады главной страницы может вызвать запросы к триаде рекламного блока, к триаде текстового блока и триаде навигационного блока. При этом модификация, например, навигационного блока сможет проходить независимо от других — а изменения отразятся во всех видах, вызывающих эту триаду.

§1.1.15. HTML

HyperText Markup Language, *англ.* язык гипертекстовой разметки. Основной язык разметки страниц Интернета, [14].

Простой документ HTML выглядит так:

Листинг 1.2. Простой документ HTML

```
<!DOCTYPE html>
2 <html>
  <head>
4   <title>Sample page</title>
  </head>
6  <body>
    <h1>Sample page</h1>
8   <p>This is a <a href="demo.html">simple</a> sample.</p>
    <!-- this is a comment -->
10  </body>
</html>
```

Документы HTML состоят из дерева элементов и текста. Начало каждого элемента указывается открывающим тегом, например, **<body>**, а конец – закрывающим, напр. **</body>**. Иногда один из тегов может опускаться, где он подразумевается другими тегами.

Теги должны быть вложенными, пересекаться элементы не могут.

Элементы имеют атрибуты, которые управляют их поведением. На строке 8 примера выше тег **a**, определяющий элемент гиперссылки, имеет атрибут **href**. Атрибуты помещаются в открывающий тег.

Пользовательские программы HTML (например, веб-браузеры) разбирают HTML разметку, превращая её в дерево DOM. Дерево DOM – это представление документа в памяти.

Деревья DOM могут включать разные ветки, например, ветку DOCTYPE, элементы, текст и комментарии.

Документ HTML – это медиа-независимое описание интерактивного содержимого. HTML документ может быть либо отрисован на мониторе либо пропущен через синтезатор речи или дисплей Брайля. Для настройки отрисовки элементов авторы могут использовать язык стилей, такой как CSS.

В настоящее время в разработке находится пятая версия стандарта HTML [15]. Отдельные положения нового стандарта уже реализованы в новых версиях некоторых популярных браузеров. Новый стандарт намного удобнее для программистов, чем использование HTML 4 и связок «Javascript и Java» и «Javascript и Flash». Например, вместо внедрения на страницу Flash-контейнера с проигрывателем видео, можно указать тег `<video>`, в формах можно указывать новые типы полей, такие как поле ввода даты (валидация реализуется в браузере) и времени, а кодировка документа объявляется упрощённым тегом META.

К сожалению, нововведения поддерживаются далеко не в полном объёме и только в новых версиях браузеров. Список поддерживаемых тегов также различается между программами. На данный момент упомянутые типы полей форм не поддерживаются нигде, а тег `<video>` имеет ограниченную функциональность и зависит от видеокодеков.

Для простоты, когда говорят о HTML 5, часто также имеют в виду новые версии других стандартов и новые Web-технологии: CSS 3, SVG, Javascript Canvas и другие. Вместе эти технологии находятся на одной ступени развития: они медленно внедряются в браузеры и постепенно получают новые свойства. Их очень просто и удобно использовать программисту, но в реальных веб-приложениях часто приходится вставлять различные обработчики для старых браузеров.

Документ HTML не отвечает правилам XML. Для совместимости с XML следует использовать спецификацию XHTML 1.0, [16].

§1.1.16. VCS

Version Control System, *англ.* система контроля версий. Эта система управляет файлами с учётом времени, [17]. Использование этих систем позволяет отой-

ти от эволюционного подхода разработки и улучшить контроль над проектом. Чаще всего применяются универсальные VCS, которые не разделяют текстовые файлы на файлы исходного кода и файлы данных.

§1.1.17. Git

Git – это систему контроля версий, написанная в 2005м году Линусом Торвальдсом для управления разработкой ядра Linux, [18]. Самой большой проблемой, по словам Линуса, была скорость существующих систем контроля версий: применение патча в проекте такого размера, как Linux, занимало полминуты – и это при условии отсутствия конфликтов.

Основными критериями в разработке дизайна Git для Торвальдса были: [19]

1. не повторять CVS¹,
2. поддерживать децентрализованную модель разработки кода,
3. обеспечить очень сильную защиту от повреждений, будь то случайные либо преднамеренные,
4. обеспечить очень высокое быстродействие.

Первоначально Git разрабатывался как низкоуровневая система, над которой уже можно было бы дорабатывать фронтенды, но со временем он превратился в полноценную систему контроля версий.

Скорость работы Git не зависит от размера проекта, [20].

Git – децентрализованная система. Это означает, что, в отличие от традиционного централизованного подхода, в котором есть один сервер, где хранится код, в Git каждый клиент также является сервером. Центральный репозиторий может лишь быть одним из многих, куда обращаются разработчики чтобы сверить свои правки.

¹Одна из первых VCS, централизованная система. Её идеи были продолжены в другой системе, SVN (Subversion).

Глава 2. Постановка задачи

Цель данной работы — разработка информационной системы (ИС), предоставляющей инструментарий для управления семинарами, проводимыми в КРИПКиПРО.

Для достижения цели необходимо:

1. Изучить предметную область
2. Сформулировать пользовательские требования к ИС
3. Выдвинуть специальные требования к ИС
4. Провести обзор и анализ существующих ИС, решающих смежные задачи
5. Построить функциональные и ER модели
6. Разработать ИС
7. Протестировать ИС

§2.1. Структура и архитектура ИС

Информационная система построена по архитектуре «клиент-сервер», см. рис. 2.1.

В качестве хранилища данных выступает реляционная система управления базами данных MySQL.

§2.2. Структурированное текстовое описание процесса

§2.2.1. Сценарий 1. Регистрация на мероприятие

Актор: Посетитель.

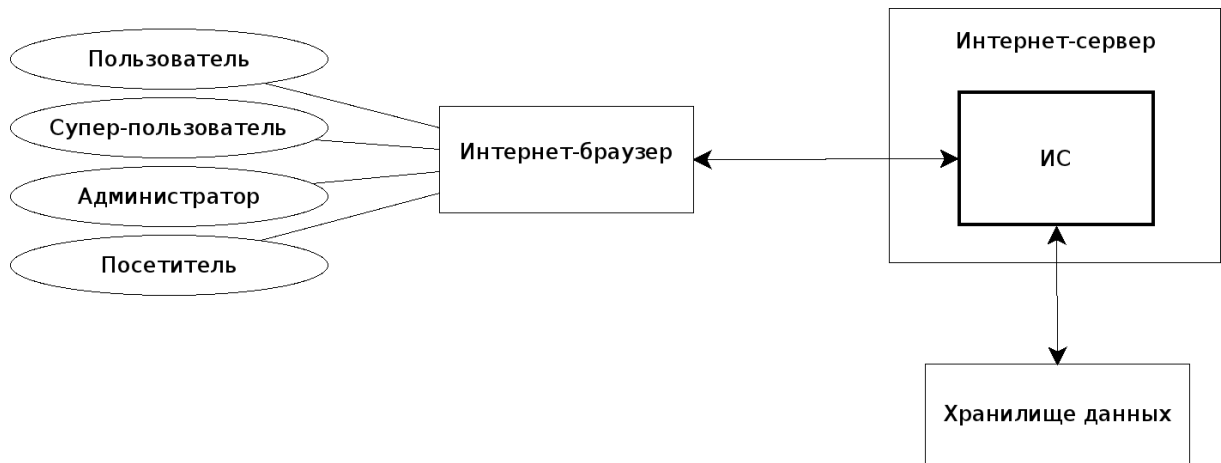


Рис. 2.1. Архитектура системы

Основной поток

1. Актор решает посетить мероприятие.
2. Актор выбирает мероприятие.
3. Актор регистрируется на мероприятие.

Расширения

- 3.a Актор указал неверные данные. Переход к п.3. сценария 1.
- 3.b Регистрация на мероприятие закрыта.

§2.2.2. Сценарий 2. Добавление ограничений

Актор: Супер-пользователь.

Основной поток

1. Актор создаёт ограничение для одного дня.

Расширения

- 1.a Актор создаёт ограничение для периода дней.
- 1.b Актор создаёт ограничение для интервала дней.

§2.2.3. Сценарий 3. Добавление мероприятия

Актор: Пользователь.

Основной поток

1. Актор получает информацию о мероприятии.
2. Актор создаёт мероприятие.
3. Актор заполняет информацию о мероприятии.

Расширения

- 3.a** Информация не проходит валидацию. Повтор пункта.
- 3.b** Актор загружает программу в формате DOC.
- 3.c** Актор добавляет ограничение на количество посетителей.
- 3.d** Мероприятие уже утверждено. Изменения отвергаются.

§2.3. Требования к системе

К информационным системам, решающим задачи подобного рода, выстав-
ляются следующие требования:

Изучение предметной области дало следующие требования к внедряемой
информационной системе:

1. Информационная система должна быть написана на языке PHP и исполь-
зовать базу данных MySQL в качестве хранилища данных,
2. ИС должна быть бесплатной,
3. ИС должна поддерживать управление событиями,
4. В ИС должна быть предусмотрена эскалация привилегий, распростра-
няемая на события,

5. ИС должна следить за регистрацией на мероприятия.
6. ИС должна иметь модульную архитектуру,
7. ИС должна иметь открытый интерфейс, при помощи которого возможно было бы обеспечить обмен данных с другими ИС, либо предоставлять средства создания подобного.

Дополнительно были выдвинуты специальные требования:

1. ИС должна поддерживать Unicode, чтобы не возникло проблем с кириллицей и совместимостью кодировок.
2. ИС должна иметь русскоязычный интерфейс.
3. ИС должна иметь средства для представления календаря мероприятий в электронном и печатном видах.

§2.4. Обзор сходных информационных систем

По вышеприведённым требованиям были отобраны три информационные системы:

1. Tiki Wiki CMS Groupware
2. Midgard CMS
3. Sitellite CMS

В ходе обзора каждая из систем проверялась на соответствие специальным требованиям, см. табл. 2.1.

По результатам обзора ни одна из рассмотренных ИС не удовлетворила предъявленные требования. Было принято решение о разработке собственной системы.

Критерий	Tiki Wiki	Midgard	Sitellite
Поддержка Unicode	есть	есть	есть
Русскоязычный интерфейс	частично	доступен по запросу	нет
Регистрация посетителей	нет	нет	нет
Представление календаря в электронном виде	есть	есть	нет
Представление календаря в печатном виде	вручную ¹	нет	нет
Открытый интерфейс	нет	нет	нет

Таблица 2.1. Сравнение некоторых сходных информационных систем

§2.5. Диаграммы

Система спланирована с учётом методологии HMVC. Благодаря этому любая из триад может быть изменена или удалена без ущерба для остальных.

Каждый модуль обрабатывает свой тип запроса. Тип запроса определяет загрузчик.

§2.6. Используемые программные средства

§2.6.1. Kohana 3.1

Основной причиной при выборе Kohana стало знакомство с фреймворком. Мало того, что автор использовал Kohana в течение года, имелся уже реализованный скелет проекта.

Наличие готовых модулей ORM и авторизации, предложение архитектуры HMVC, чёткое оформление кода, активная разработка и стабильное состояние программной базы также повлияли на решение.

Отличительной особенностью Kohana 3.1 является его исключительная модульность. Он реализует модель HMVC.

Краткий обзор API фреймворка размещён в параграфе 3.2

§2.6.2. jQuery

Это мощная Javascript библиотека, позволяющая проводить запросы AJAX, реализовывать визуальные эффекты и работать с DOM без оглядки на совместимость браузеров и разницу в вызовах событий.

На настоящий момент jQuery – самая быстрая из всех популярных (согласно [21]) библиотек. Также к ней написано много готовых бесплатных свободно распространяемых плагинов.

§2.6.3. jQueryUI

Плагин для jQuery, позволяющий создавать привычные пользовательские интерфейсы – в том числе диалоги.

§2.6.4. jEditable

Плагин для jQuery, позволяющий редактировать текст на месте, не покидая страницы и не нарушая разметки.

§2.6.5. fullCalendar

Плагин для jQuery, имитирующий Google Calendar. Принимает список мероприятий в формате JSON; визуализирует его в виде календаря с указанием перекрывающихся событий.

§2.6.6. Git

Удобная децентрализованная система управления версиями.

Отличается очень развитым механизмом веток и быстрой скоростью работы.

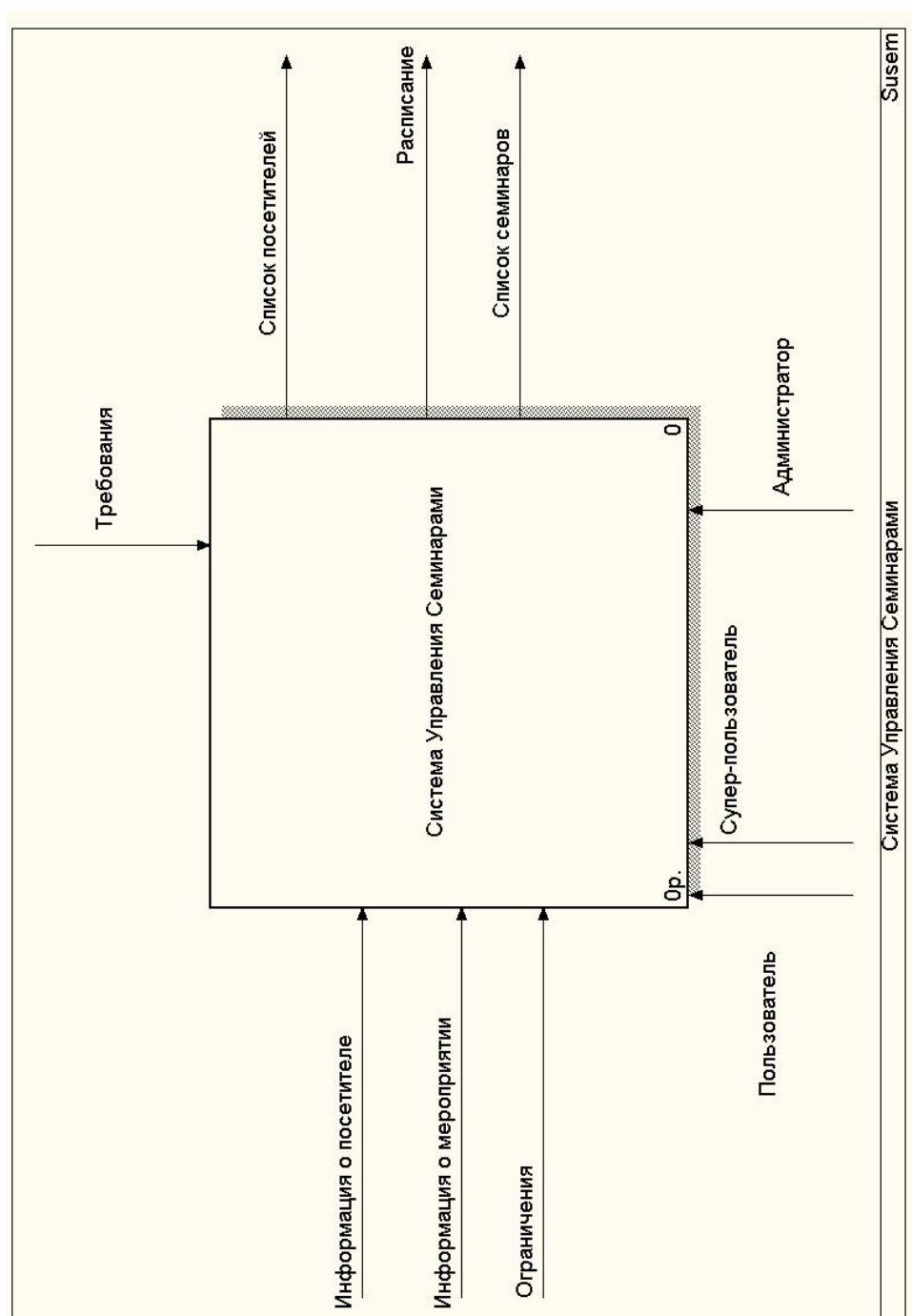


Рис. 2.2. Диаграмма IDEF0

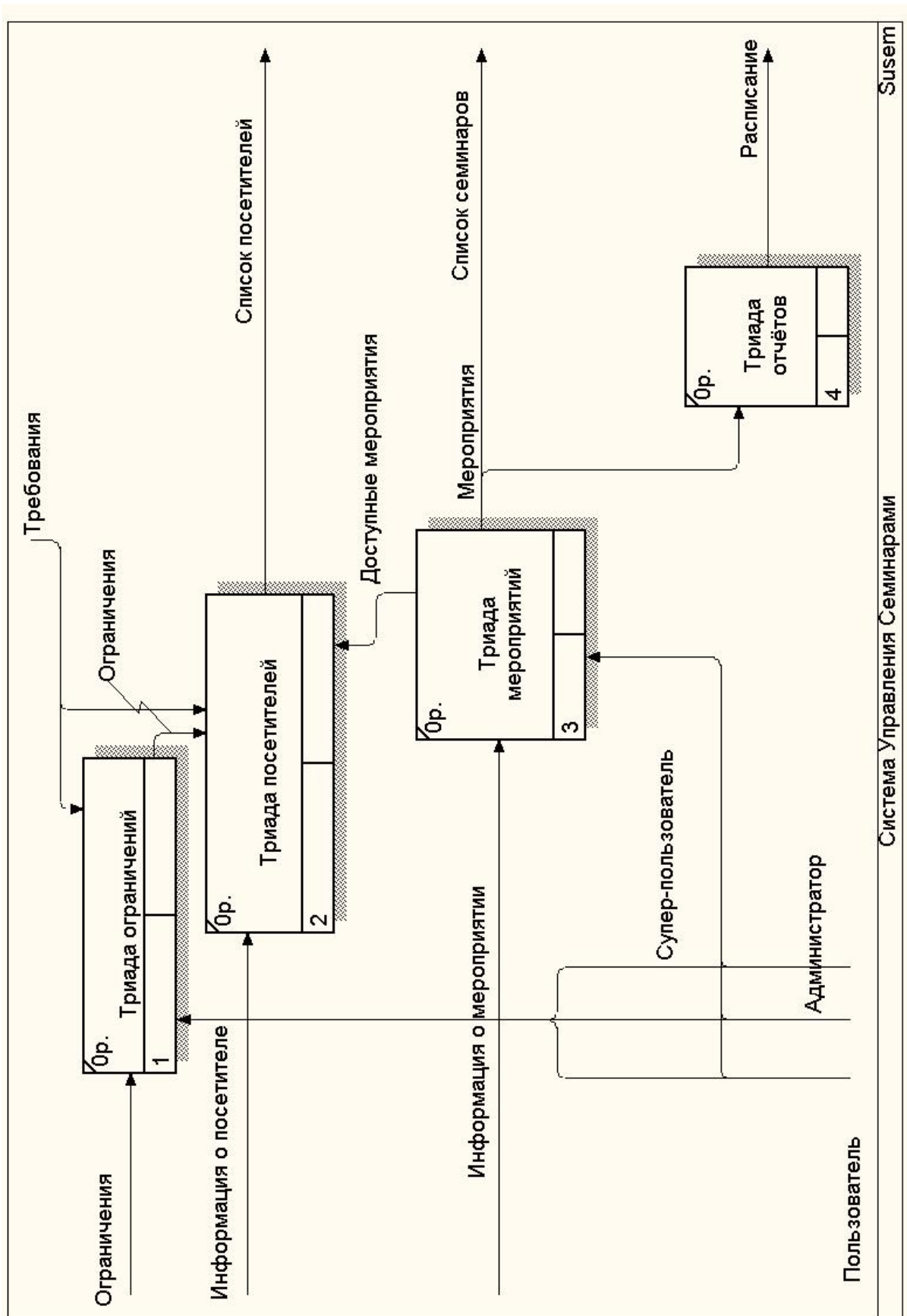


Рис. 2.3. Диаграмма IDEF3

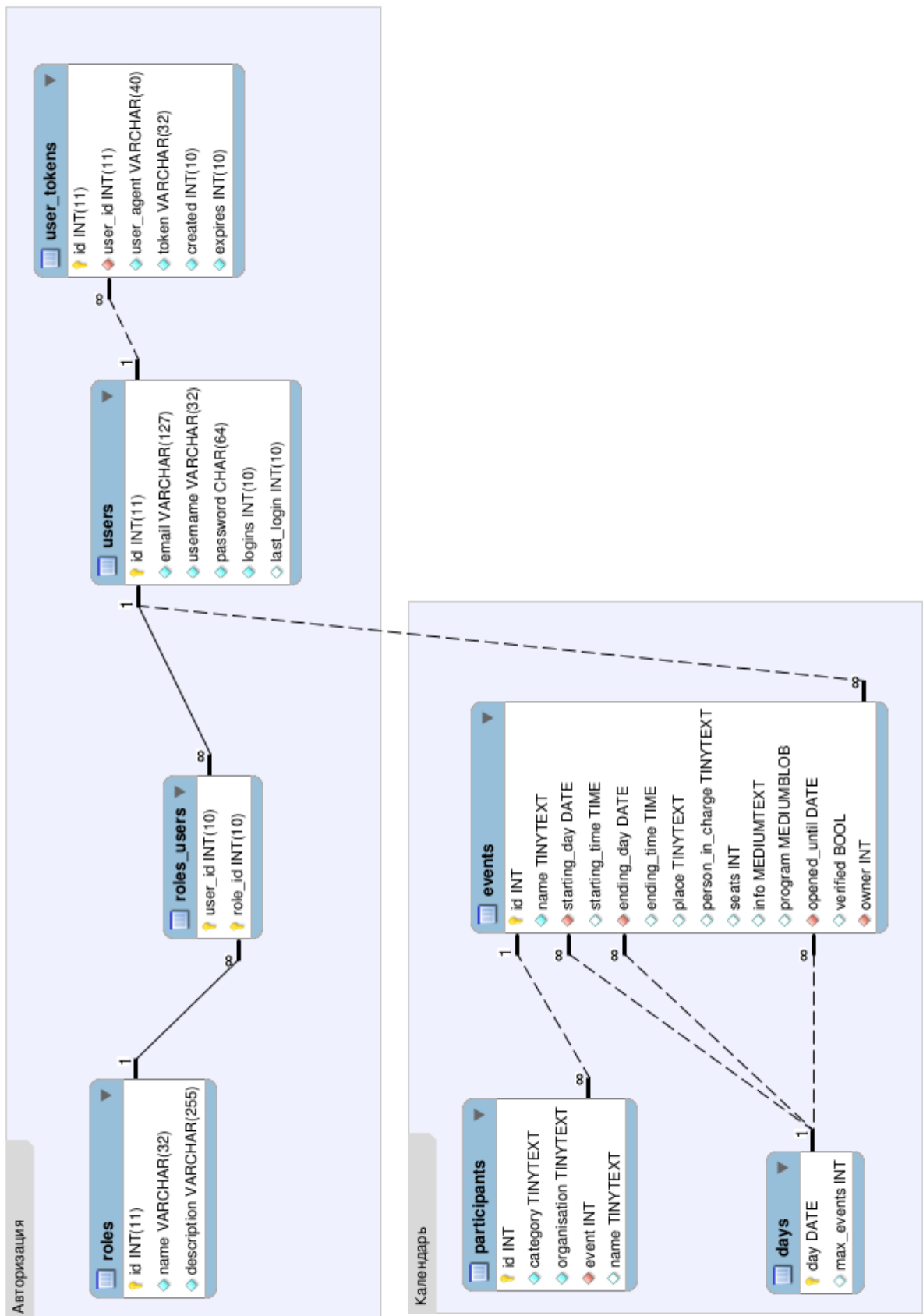


Рис. 2.4. ER-модель

Глава 3. Реализация поставленной задачи

§3.1. Совместимые клиентские программы

Предполагается, что клиент использует браузер, поддерживающий скрипты языка Javascript 1.1 и стандарты CSS 2, а также рекомендации стандарта HTML 5.

§3.2. Краткий обзор API Kohana

Фреймворк Kohana – это бывшая ветка фреймворка CodeIgniter. Третья версия была почти полностью переписана с нуля.

Версия 3.1 – это последняя стабильная в настоящее время. В ней был расширен объект Request, а также улучшена валидация форм.

§3.2.1. Файловая организация

В обычном проекте Kohana три директории: `application`, `modules` и `system`.

Каталог `application` – это собственно проект. Файлы модулей и ядра править не потребуется.

Каталог `system` – это ядро Kohana.

Каталог `modules` – это модули Kohana. Сейчас написано более 500 модулей для Kohana различных версий. Включение и отключение модулей производится в файле `bootstrap.php`.

На файл `index.php` при помощи `.htaccess` перенаправляются все запросы к несуществующим файлам. То есть, если набрать `/1.jpg`, то Apache либо покажет вам картинку, либо передаст запрос в Kohana (вместо ошибки 404). Дальше действовать будет она. В файле `index.php` можно написать обработку ошибок (например, с кодами 404 и 500).

kohana /

name
 application/
 modules/
 .gitignore
 .gitmodules
 .gitmodules-dev
 LICENSE.md
 README.md
 build.xml
 example.htaccess
 index.php
 install.php
 system - ed12c18

Рис. 3.1. Корневой каталог сайта

Файл `bootstrap.php` – это основной настроечный файл Kohana. В нём выставляются язык, часовой пояс, модули, пути, кодировка и маршруты (routes). К маршрутам я ещё вернусь, а пока что – по директориям.

Директория `cache` нужна только если в `bootstrap` включено кэширование. В ней лежат уже сгенерированные страницы видов.

Директория `logs` – это логи Kohana. По умолчанию туда скидываются только ошибки (исключения, Exception) Kohana – причём те, которые не были обработаны (т.е. если вы ловите ошибку 404, то она не будет записана в лог).

Директория `config` – это настройки. Сюда и только сюда стоит скидывать настройки для модулей, например, Database. На то есть веская причина.

Kohana может обслуживать несколько проектов сразу. Директории ядра системы и модулей остаются теми же, а вот папки с проектом подключаются раз-

kohana / application

name	age
..	
cache/	Decembe
classes/	Decembe
config/	May 29,
i18n/	May 29,
logs/	Decembe
messages/	May 29,
views/	May 16,
bootstrap.php	Februar

Рис. 3.2. Каталог application, минимальное наполнение

ные. Естественно, что разные проекты требуют разных настроек – поэтому логично хранить настройки вместе со всем проектом.

Загрузчик Kohana отдаёт больший приоритет файлам application, чем файлам модулей, поэтому можно переопределять их.

Каталог views хранит шаблоны страниц. С подключением определённых модулей становится возможным писать шаблоны не на PHP, а на специальных языках, например, Twig (где сама возможность программирования логики жёстко отключена).

Каталог classes – самый главный каталог. Здесь хранятся все классы проекта: контроллеры и модели.

Kohana 3 – это объектно-ориентированный фреймворк. Каждый объект объявляется в своём классе. Каждый класс занимает собственный файл. Имя класса однозначно определяет путь файла.

В названии класса слова разделяются подчёркиваниями, например:

Underscore_Class

Controller_Rule_Of_Three

Примеры неправильных названий:

CamelCased

Mumble-Wimble-Class

Загрузка класса идёт по обращению к нему. Класс `Underscore_Class` должен быть объявлен в файле `classes/underscore/class.php`. Все имена файлов и директорий должны быть в нижнем регистре (это важно, особенно на Unix-системах).

§3.2.2. Маршруты

Листинг 3.1. Пример маршрута

```
1 Route::set('default', '(<controller>(/<action>(/<id>)))')
  ->defaults(array(
3     'controller' => 'welcome',
     'action'      => 'index',
5 ));
```

Маршрут – это то правило, по которому контроллеры сопоставляются адресам. По умолчанию есть лишь одно правило: `default`. Его пример приведён выше.

В первой строке примера задаются название маршрута и сам маршрут: имя контроллера – слэш и название действия – ID. Скобки показывают, что параметр необязателен (например, ID передаётся далеко не всем контроллерам). Вместо слэша может быть всё что угодно, например, если поставить запятую, то разделителем станет любой небуквенный символ. Тогда Kohana станет понимать запросы вида `welcome-index#21`. Маршрут преобразовывается в регулярное выражение, ему всё равно.

В строках 3 и 4 задаются параметры по умолчанию. Если не написать `action`, то будет автоматически подставлено `index`. Если же вообще ничего не написать

(например, обратиться к корню), то будут автоматически подставлены имя контроллера и действие. Так можно управлять точками входа.

Предположим, что мы решили создать каталог `admin`, в котором будут также лежать контроллеры. Тогда напишем такой маршрут:

Листинг 3.2. Маршрут для директории контроллеров

```
1 Route::set('administration',  
    'admin/( <controller>(/<action>(/<id>)))')  
->defaults(array(  
3     'controller' => 'welcome',  
    'action'      => 'index',  
5     'directory'  => 'admin'  
));
```

В новом маршруте все запросы, начинающиеся со строки `admin/`, будут обрабатываться контроллерами в директории `admin`. По умолчанию в запрос будут подставляться значения имени контроллера `admin/welcome` и действия `index`.

§3.2.3. Контроллеры

Контроллер должен наследовать класс `Controller` или `Controller_Template`, если вид – это шаблон, а не HTML страница.

Контроллер – это единственная необходимая составляющая триады. Вот типичный пример минимально работающего контроллера. Он не требует ни вида, ни модели.

Листинг 3.3. Минимальный контроллер

```
<?php defined('SYSPATH') or die('No direct access.');
```

```
2 class Controller_Welcome extends Controller {  
    public function action_index()  
        { $this->response->body('hello , world!');}  
4 }
```

Всё, что он делает – это выводит строку „Hello, world!”.

Обратите внимание на первую строку. Так как `.htaccess` перенаправляет в Kohana запросы только к несуществующим файлам, то необходимо запретить чтение `php` файлов прямым обращением. Если переменная `SYSPATH` (файл `bootstrap.php`) не задана, то значит, файл выполняется в обход Kohana, что не разрешено. С этой строки рекомендуется начинать каждый файл класса.

Каждый контроллер имеет свойство `$this->request`. Вот его некоторые часто используемые методы и свойства:

1. `$this->request->directory()` – директория, где лежит контроллер
2. `$this->request->controller` – имя контроллера
3. `$this->request->action` – имя вызываемого действия
4. `$this->request->param()` – любой другой параметр маршрута
5. `$this->request->redirect()` – безусловный редирект на указанный адрес (вывод данных невозможен)
6. `$this->request->query()` – установить или взять параметр GET
7. `$this->request->post()` – установить или взять параметр POST

Перед вызовом любого из действий выполняется метод `before()`, после – метод `after()`. Это можно использовать, например, для проверки прав пользователя.

§3.2.4. ORM

ORM расшифровывается как Object Relationship Modeling. Она представляет данные в виде объектов и даёт методы для манипулирования.

ORM позволяет хранить данные как в файлах, так и в базе данных; драйвер для БД поставляется с модулем `Database`. Для простоты про файлы я рассказывать не буду.

Каждая модель должна наследовать класс `ORM`.

В простоте ORM многое основано на принятых обозначениях:

1. Таблицы всегда названы по-английски во множественном числе, например, `users`.
2. Модели называются по-английски в единственном числе, например, `User`; к названию модели добавляется `_Model`. Это переопределяется переменной `$table_name`.
3. По умолчанию первичным ключом (primary key) считается столбец `ID` типа `INT` (любой `INT`) с установленным `AUTO_INCREMENT.UNSIGNED` необязателен. Это переопределяется переменной `$primary_key`.
4. Каждый foreign key называется по схеме название-модели_id, например, `user_id`
5. Промежуточные таблицы (в отношениях много-ко-многим) называются составлением родительских таблиц в алфавитном порядке. Так, промежуточная между `users` и `roles` таблица называется `roles_users`.

Эти обозначения значительно сокращают запись моделей и помогают понятно организовать саму базу данных.

Загрузка новой модели происходит так (оба варианта равнозначны):

```
$user = ORM::factory('user');  
$user = new Model_User();
```

При этом загрузки данных не происходит (так называемая ленивая загрузка). Данные загружаются или изменяются только тогда, когда подаются команды загрузки или изменения.

Листинг 3.4. Пример работы с моделью

```
$user = ORM::factory('user');
```

```

2 $user->first_name = 'Trent';
  $user->last_name = 'Reznor';
4 $user->city = 'Mercer';
  $user->state = 'PA';
6 $user->create();

```

Строки 2-5 задают значения полей таблицы. Сохранение данных в базе происходит только на последней строке.

Можно объявлять модель сразу определённой записи:

Листинг 3.5. Варианты объявления модели с определённым primary key

```

$user = ORM::factory('user')
2   ->where('id', '=', 20)
    ->find();
4 $user = ORM::factory('user', 20);
$user = new Model_User(20);

```

Сохранение идёт либо методом `create()` (создание), либо `update()` (обновление).

Метод `delete()` удаляет запись.

Отношения

belongs_to

Отношение `belongs_to` описывает отношение «один-ко-многим». Объявляется таким образом:

Листинг 3.6. Объявление `belongs_to`

```

1 protected $_belongs_to = array('[alias]' =>
    array('model' => '[model]', 'foreign_key' =>
        '[column]'))

```

Если один или несколько параметров опущены, подставляются стандартные значения. По умолчанию псевдоним совпадает с именем модели, а foreign key будет псевдонимом с добавленным в конце суффиксом `_id`.

Использование отношения `belongs_to` иногда требует объявления парного `has_many` по другую сторону связи.

has_many

Отношение `has_many` – «много-к-одному» – противоположно предыдущему. Обычно оно объявляется по другую сторону.

Листинг 3.7. Объявление `has_many`

```
1 protected $_has_many = array('alias name' =>
    array('model' => '[model name]', 'foreign_key' =>
        '[column]'))
```

has_one

Отношение `has_one` – «один-к-одному» почти идентично `has_many`, но описывает уникальные связи.

has_many „through”

Отношение «много-ко-многим» описывается всегда через промежуточную таблицу. Например, это пользователи и их роли: пользователь может иметь несколько ролей, роль может принадлежать нескольким пользователям. Чтобы связать обе модели, нужна дополнительная таблица.

В одной из них объявляется:

Листинг 3.8. Объявление `has_many_through`, часть 1

```
1 protected $_has_many = array('alias' => array('model'
    => 'model2', 'through' => 'table_through'))
```

Здесь `alias` – это псевдоним отношения, а `table_through` – это промежуточная таблица.

Создавать новую модель не надо. С другого конца соединения необходима:

Листинг 3.9. Объявление `has_many_through`, часть 2

```
1 protected $_belongs_to = array('model_name' => array())
```

После этого можно манипулировать связями:

Листинг 3.10. Манипулирование связями

```
1 $post->has( ' category ', $category )  
   $post->add( ' category ', $category )  
3 $post->remove( ' category ', $category )
```

§3.2.5. Auth

Модуль Auth также может работать с файлами и базой данных; здесь файлы не рассматриваются.

Модуль предоставляет базовые функции управления пользователями. Каждому пользователю сопоставляется несколько ролей – предполагается, что каждая роль имеет свои привилегии. Роль `login` закреплена жёстко, это – право заходить на сайт. Если пользователь не имеет этой роли, он не может авторизоваться и фактически не имеет доступа.

В модуле определены модели `Model_User` и `Model_Role`.

§3.3. Авторизация пользователя

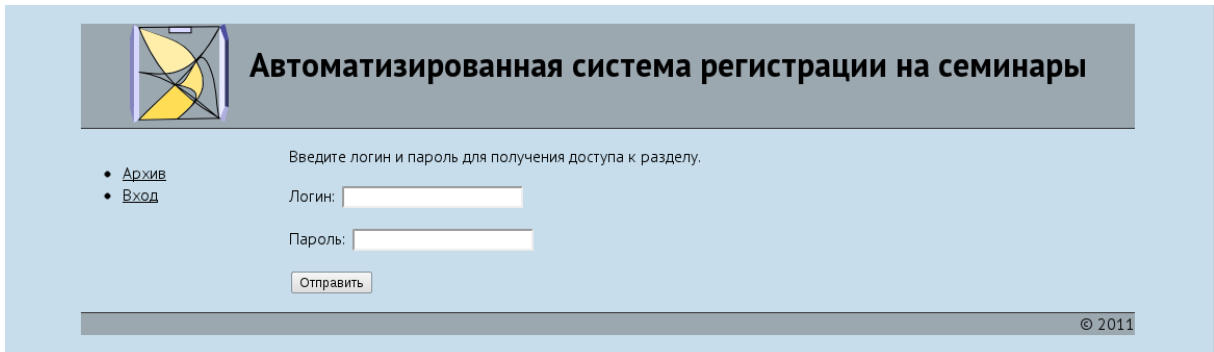


Рис. 3.3. Ввод логина и пароля

Каждый пользователь имеет несколько ролей. В системе существуют четыре роли:

1. Администратор (admin)
2. Супер-пользователь (superuser)

3. Пользователь (user)

4. Владелец пароля (login)

О роли login уже было сказано выше; это – право авторизации в системе.

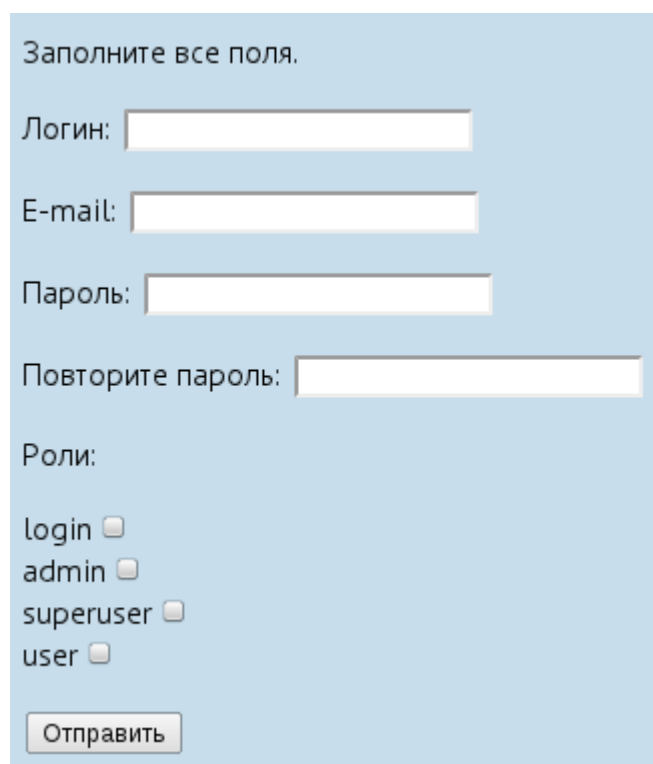
Роль user даёт право управления мероприятиями. Пользователю также доступны просмотр расписания и просмотр посетителей.

Роль superuser добавляет функцию утверждения расписания, а также управление ограничениями. Помимо этого, супер-пользователь может изменять мероприятия, созданные пользователями.

Роль admin даёт неограниченные права, а также открывает доступ к управлению пользователями.

Если введённые логин и пароль неверны, об этом сообщается пользователю.

§3.4. Регистрация пользователей



Заполните все поля.

Логин:

E-mail:

Пароль:

Повторите пароль:

Роли:

login ☐

admin ☐

superuser ☐

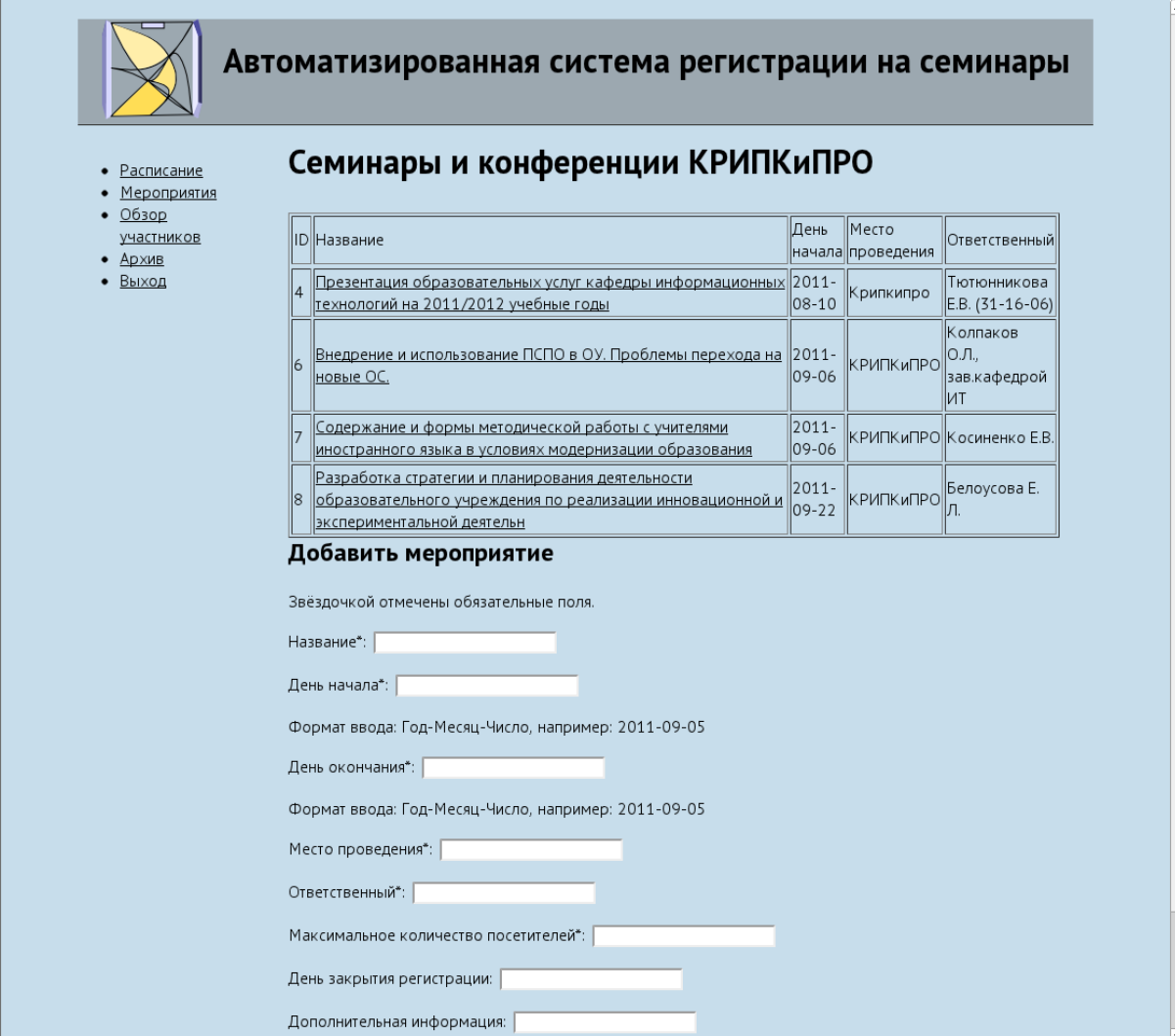
user ☐

Рис. 3.4. Интерфейс создания пользователя

Регистрация пользователей доступна только администраторам.

При регистрации администратор указывает логин, e-mail и пароль нового пользователя. Также он выставляет его роли.

§3.5. Управление мероприятиями



Автоматизированная система регистрации на семинары

- Расписание
- Мероприятия
- Обзор участников
- Архив
- Выход

Семинары и конференции КРИПКиПРО

ID	Название	День начала	Место проведения	Ответственный
4	Презентация образовательных услуг кафедры информационных технологий на 2011/2012 учебные годы	2011-08-10	Крипкипро	Тютюнникова Е.В. (31-16-06)
6	Внедрение и использование ПСПО в ОУ. Проблемы перехода на новые ОС.	2011-09-06	КРИПКиПРО	Колпаков О.Л., зав.кафедрой ИТ
7	Содержание и формы методической работы с учителями иностранного языка в условиях модернизации образования	2011-09-06	КРИПКиПРО	Косиненко Е.В.
8	Разработка стратегии и планирования деятельности образовательного учреждения по реализации инновационной и экспериментальной деятельн	2011-09-22	КРИПКиПРО	Белоусова Е.Л.

Добавить мероприятие

Звёздочкой отмечены обязательные поля.

Название*:

День начала*:

Формат ввода: Год-Месяц-Число, например: 2011-09-05

День окончания*:

Формат ввода: Год-Месяц-Число, например: 2011-09-05

Место проведения*:

Ответственный*:

Максимальное количество посетителей*:

День закрытия регистрации:

Дополнительная информация:

Рис. 3.5. Обзор мероприятий

В разделе мероприятий перечисляются все доступные мероприятия. Также есть фильтр по текущему пользователю, см. рис. 3.5.

Название каждого мероприятия является ссылкой на страницу с деталями. Если текущий пользователь имеет права на редактирование этого мероприятия, то он может отредактировать данные. По нажатию левой кнопки мыши любое поле превращается в форму, в которой можно будет изменить его значение.

Внедрение и использование ПСПО в ОУ. Проблемы перехода на новые ОС.

Проходит с: 2011-09-06 по: 2011-09-06

Место проведения: КРИПКиПРО

Ответственный:

Колпаков О.Л., зав.кафедрой ИТ

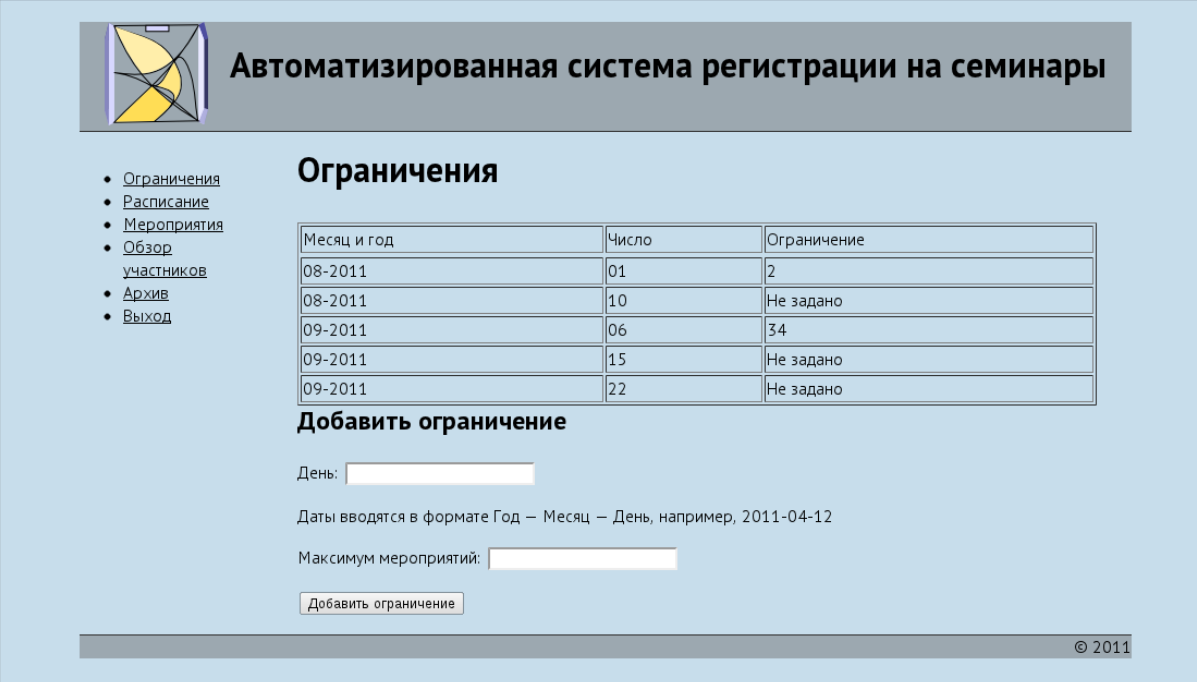
Количество мест: 4

Регистрация открыта до: 2011-08-10

Рис. 3.6. Редактирование мероприятия

Добавление мероприятий на определённые дни закрывается по достижении максимального значения для этого дня.

§3.6. Выставление ограничений



Автоматизированная система регистрации на семинары

Ограничения

Месяц и год	Число	Ограничение
08-2011	01	2
08-2011	10	Не задано
09-2011	06	34
09-2011	15	Не задано
09-2011	22	Не задано

Добавить ограничение

День:

Даты вводятся в формате Год — Месяц — День, например, 2011-04-12

Максимум мероприятий:

© 2011

Рис. 3.7. Выставление ограничений

Выставление ограничений доступно только администраторам и суперпользователям.

На странице выставления ограничений перечисляются все дни, данные о которых есть в базе. Для каждого из этих дней существует либо ограничение, либо связь как минимум с одним мероприятием. По этой причине удаление ограни-

чений невозможно – пользователь может лишь выставить ограничение в 0 или найти какое-нибудь высокое значение.

По нажатию левой кнопки мыши на числах в третьем столбце возникает форма редактирования. При потере фокуса значение формы сохраняется.

Ошибки появляются в специальном поле вверху страницы.

§3.7. Обзор расписания

Обзор расписания доступен всем авторизованным пользователям, рис. ??.

Автоматизированная система регистрации на семинары

Отчёт

• [Расписание](#)
• [Мероприятия](#)
• [Обзор участников](#)
• [Архив](#)
• [Выход](#)

◀ ▶ today **Oct 2 – 8 2011** month week day

Sun 10/2	Mon 10/3	Tue 10/4	Wed 10/5	Thu 10/6	Fri 10/7	Sat 10/8
				Внедрение и использование ПСПО в ОУ. Проблемы перехода на новые ОС. Содержание и формы методической работы с учителями иностранного языка в условиях модернизации образования		

Недостаточно прав для утверждения расписания.

© 2011

Рис. 3.8. Обзор расписания

Администратору и супер-пользователю также открывается функция утверждения расписания.

Утверждённые мероприятия недоступны для изменений.

§3.8. Регистрация посетителей

Гости сайта могут регистрироваться на мероприятия.

Гостям доступен список будущих мероприятий, рис. 3.9. Также есть раздел «Архив», где хранится информация о прошедших мероприятиях.

Автоматизированная система регистрации на семинары

- [Архив](#)
- [Вход](#)

Мероприятия

День начала	Название	Место проведения	Ответственный	
Август 2011				
2011-08-10	Презентация образовательных услуг кафедры информационных технологий на 2011/2012 учебные годы	Крипкипро	Тютюнникова Е.В. (31-16-06)	Зарегистрироваться
Сентябрь 2011				
2011-09-06	Внедрение и использование ПСПО в ОУ. Проблемы перехода на новые ОС.	КРИПКиПРО	Колпаков О.Л., зав.кафедрой ИТ	Зарегистрироваться
2011-09-06	Содержание и формы методической работы с учителями иностранного языка в условиях модернизации образования	КРИПКиПРО	Косиненко Е.В.	Регистрация закрыта.

Рис. 3.9. Обзор мероприятий – вид посетителя

Кнопка «Зарегистрироваться» появляется если не достигнуты ограничения, связанные с данным мероприятием.

При нажатии на кнопку «Зарегистрироваться» появляется диалог, изображённый на рис. 3.10.

Введённые значения сохраняются в таблице participants базы данных.

§3.9. Обзор посетителей

Количество зарегистрированных посетителей для каждого мероприятия – это ссылка на детальную таблицу, в которой перечислены все зарегистрированные. Обзор посетителей доступен всем авторизованным в системе пользователям, см. рис. 3.11.

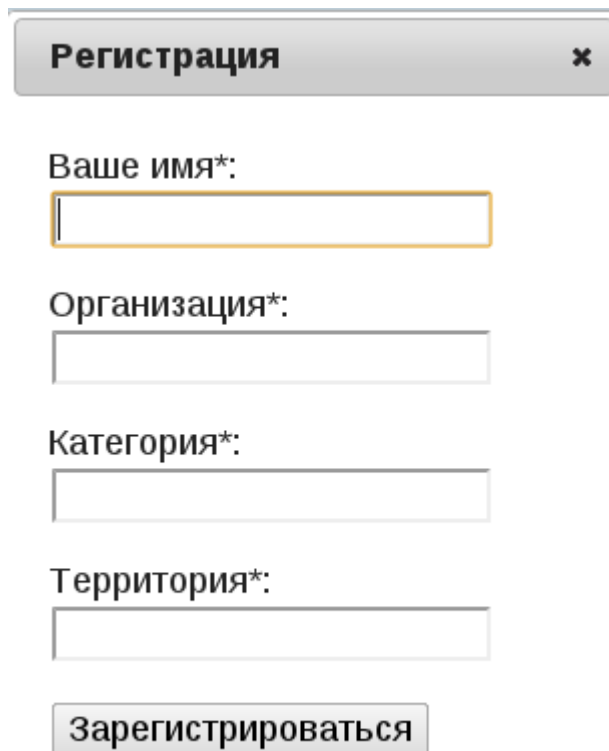
A registration dialog box with a title bar 'Регистрация' and a close button 'x'. It contains four text input fields labeled 'Ваше имя*', 'Организация*', 'Категория*', and 'Территория*'. At the bottom is a button labeled 'Зарегистрироваться'.

Рис. 3.10. Диалог регистрации

§3.10. Стил ь программирования

При редактировании кода учитывались следующие правила:

1. Следовать концепции строгого HMVC. Библиотеки PHP оформлять как модули. Ничто, связанное с проектом, не может находиться вне триад MVC.
2. Запросы к БД производить исключительно через ORM.
3. Обращение к базе данных (составление запросов) происходит только в модели. Контроллер использует API модели.
4. В виде не используется PHP, кроме `echo`, циклов¹ и вызовов контроллеров других триад.
5. Отступы реализованы при помощи пробелов. Размер отступа – 1 пробел.

¹Это правило кое-где сознательно нарушено, чтобы не перегружать контроллер лишним циклом.



Автоматизированная система регистрации на семинары

- Расписание
- Мероприятия
- Обзор участников
- Архив
- Выход

Посетители

#	Название мероприятия	Посетители
1	Презентация образовательных услуг кафедры информационных технологий на 2011/2012 учебные годы	1
2	Внедрение и использование ПСПО в ОУ. Проблемы перехода на новые ОС.	0
3	Содержание и формы методической работы с учителями иностранного языка в условиях модернизации образования	0
4	Разработка стратегии и планирования деятельности образовательного учреждения по реализации инновационной и экспериментальной деятельн	0

© 2011

Рис. 3.11. Обзор посетителей

Также существуют гайдлайны самой Kohana [22], но в проекте они не выполняются.

Заключение

Итогом подробного анализа предметной области стали следующие выводы.

В настоящее время проблема регистрации на получение дополнительного образования посредством сети Интернет действительно актуальна.

Изучение предметной области дало понять, что информационная система должна соответствовать определённому набору требований.

Проведенный обзор существующих на данный момент систем, решающих подобные задачи, выявление их достоинств и недостатков, позволяет создать информационную систему, в которой учтены существующие недочеты и добавлены те службы и сервисы, которых нет в остальных;

В ходе обзора существующие системы показали отсутствие средств управления посетителями мероприятий и плохую открытость интерфейса. С другой стороны, они предоставляют удобный онлайн-интерфейс управления мероприятиями.

Разработанная ИС позволяет создавать пользователей и назначать им привилегии; пользователи могут в зависимости от привилегий управлять мероприятиями, выставять ограничения на мероприятия и смотреть расписание. Также ИС предоставляет возможность регистрации посетителей на мероприятия, закрывая автоматически регистрацию на заполненные и прошедшие (точная дата настраивается) мероприятия.

Для тестирования был написан автоматический тест, проверяющий на существование все страницы сайта. Его исходный код приведён в приложениях. В ходе тестирования ошибок найдено не было.

Таким образом, поставленная цель достигнута, все задачи выполнены. Разработанная ИС готова к внедрению в КРИПКиПРО.

Литература

1. *Lindley, Cody*. jQuery Cookbook / Cody Lindley. — O'Reilly Media, 2010. — 451 pp.
2. *Пауэлл, Томас*. Полный справочник по Javascript / Томас Пауэлл, Крис Шнайдер. — 2 изд. — М.: ИД «Вильямс», 2006. — 960 с.
3. *Крейн, Дейв*. AJAX в действии / Дейв Крейн, Эрик Паскарелло. — М.: ИД «Вильямс», 2008. — 639 с.
4. [Электронный ресурс]. <http://tools.ietf.org/html/rfc3875>.
5. РНР 5 для профессионалов / Эд Леки-Томпсон, Алек Коув, Стивен Новицки, Хьяо Айде-Гудман. — М.: ИД «Вильямс», 2006. — 608 с.
6. *Гудман, Дэнни*. Javascript. Библия пользователя / Дэнни Гудман, Майкл Моррисон. — 5 изд. — М.: ИД «Вильямс», 2006. — 1184 с.
7. [Электронный ресурс]. <http://tools.ietf.org/html/rfc2616>.
8. [Электронный ресурс]. <http://www.w3.org/TR/REC-xml/>.
9. *Седерхолм, Дэн*. CSS ручной работы / Дэн Седерхолм. — СПб.: Питер, 2011. — 240 с.
10. *Уэнц, Кристиан*. Javascript. Карманный справочник / Кристиан Уэнц. — М.: ИД «Вильямс», 2007. — 272 с.
11. *Маклафлин, Б.* Изучаем Ajax / Б. Маклафлин. — СПб.: Питер, 2007. — 448 с.
12. [Электронный ресурс]. <http://www.adaptivepath.com/ideas/e000385>.
13. [Электронный ресурс]. <http://www.w3.org/DOM/>.

14. *Хольцшлаг, Молли Э.* Использование HTML и XHTML / Молли Э. Хольцшлаг. — 2 изд. — М.: ИД «Вильямс», 2003. — 736 с.
15. [Электронный ресурс]. <http://dev.w3.org/html5/spec/Overview.html>.
16. *Валентайн, Челси.* XHTML / Челси Валентайн, Крис Минник. — М.: ИД «Вильямс», 2001. — 480 с.
17. *Collins-Sussman, Ben.* Version Control with Subversion / Ben Collins-Sussman, Bryan Fitzpatrick, Carl Pilato. — LA, CA: O'Reilly, 2004. — 328 pp.
18. *Loeliger, Jon.* Version control with Git / Jon Loeliger. — O'Reilly, 2009. — 310 pp.
19. [Электронный ресурс]. <http://www.youtube.com/watch?v=4XpnKHJAok8>.
20. *Chacon, Scott.* Pro Git / Scott Chacon, Patrick Aljord. — Springer, 2009. — 288 pp.
21. [Электронный ресурс]. <http://mootools.net/slickspeed/>.
22. [Электронный ресурс]. <http://kohanaframework.org/3.1/guide/kohana/conventions>.

Приложения

Листинг 11. Автоматический тест на Perl

```
1 #!/usr/bin/perl
   use strict;
3 my $in="";
   my $usestdin;
5 my $geturl = $ARGV[0];
   my $firsturl= $geturl;
7 my %rooturls;
   $rooturls{$ARGV[0]}=1;
9 my $linkcheck = "curl -s -m 20";
   my $linkcheckfull = "curl -s -m 20";
11 my $htmlget = "curl";
   my $getprotocol;
13 my $getserver;
   my $getpath;
15 my $getdocument;
   my %done;
17 my %tagtype;
   my $allcount=0;
19 my $badlinks=0;
   my @links;
21 sub SplitURL {
   my $inurl = $_[0];
23   if($inurl=~ /^([^:]+):\\\/\\\/([^\\/]*)\\\/(.*)\\\/(.*)/ ) {
     $getprotocol = $1;
25     $getserver = $2;
     $getpath = $3;
27     $getdocument = $4;
   }
29   elsif ($inurl=~ /^([^:]+):\\\/\\\/([^\\/]*)\\\/(.*)/ ) {
     $getprotocol = $1;
```

```

31 $getserver = $2;
    $getpath = $3;
33 $getdocument = "";
    if($getpath !~ /\:\/\/) {
35     $getpath = "";
        $getdocument = $3;
37     }
    }

39 elseif ($inurl=~ /^([^:]+) :\/\:\/\/(.*)\/ ) {
    $getprotocol = $1;
41     $getserver = $2;
        $getpath = "";
43     $getdocument = "";
    }
45 else {die ("Cannot parse URL");}
}

47 my @indoc;
    sub GetRootPage {
49     my $geturl = $_[0];
        my $in="";
51     my $code=200;
        my $type="text/plain";
53     my $pagemoved=0;
        open(HEADGET, "$linkcheck $geturl|") || die;
55     while(<HEADGET>) {
        if($_ =~ /HTTP\/1\.[01] (\d\d\d) /) {
57         $code=$1;
            if($code =~ /^3/) {
59         $pagemoved=1;
            }
61     }

        elseif($_ =~ /^Content-Type: ([\a-zA-Z]+)/) {
63         $type=$1;
        }

65     elseif($pagemoved &&

```

```

        ($_ =~ /^Location: (.*)/)) {
67   $geturl = $1;

69   &SplitURL( $geturl);

71   $pagemoved++;
      last;
73 }
    }

75 close(HEADGET);
    if($pagemoved == 1) {
77   print "Page is moved\n";
      exit;
79 }

    open(WEBGET, "$htmlget $geturl|") || die;
81 while(<WEBGET>) {
    my $line = $_;
83   push @indoc, $line;
      $line =~ s/\n/ /g;
85   $line =~ s/\r//g;
      $in=$in.$line;
87 }

    close(WEBGET);
89   return ($in, $code, $type);
  }

91 sub LinkWorks {
    my $check = $_[0];
93   my @doc = '$linkcheck \"$check\"';
      my $head = 1;
95   boo:
      if( $doc[0] =~ /^HTTP[^\ ]+ (\d+)/ ) {
97   my $error = $1;
      if($error < 400 ) {
99     return "GOOD";
      }

```

```

101 else {
    if($head && ($error >= 500)) {
103 @doc = '$linkcheckfull \"$check\"';
        $head = 0;
105 goto boo;
    }
107 return "BAD";
    }
109 }
    return "BAD";
111 }

sub GetLinks {
113 my $in = $_[0];
    my @result;
115 while($in =~ /[^\<]*(\<[^\>]+\>)/g ) {
    my $tag = $1;
117
    if($tag =~ /\<!--/) {}
119 else {
        if($tag =~ /(src|href|background|archive) *=
            *("\ "[^\"]"\ " | [^\ \>]*)/i) {
121 my $url=$2;
            if($url =~ /\^\ "(.*)\ "$/) { $url=$1;}
123 $url =~ s/([^\#]*)\#.*$/1/g;
            if($url eq "") {
125 next;
            }
127 if($done{$url}) {
                $done{$url}++;
129 next;
            }
131 $done{$url} = 1;
            push @result, $url;
133 if($tag =~ /< *([^\ ]+)/) {
                $tagtype{$url}=$1;

```



```

135     }
        }
137     }
        }
139     return @result;
    }
141 while(1) {
    $geturl=-1;
143 for(keys %rooturls) {
    if($rooturls{$_} == 1) {
145     if($_ !~ /^$firsturl/) {
        $rooturls{$_} += 1000;
147     next;
    }
149     $geturl=$_;
    last;
151 }
    }
153 if($geturl == -1) {
    last;
155 }
    &SplitURL($geturl);
157 my ($in, $error, $ctype) = &GetRootPage($geturl);
    $rooturls{$geturl}++;
159 if($ctype ne "text/html") {}
    if($error >= 400) {
161     print "$geturl return $error, exiting\n";
    exit;
163 }
    for(@links) {
165     my $url = $_;
    my $link;
167     if($url =~ /^([^:]+):/) {
        my $prot = $1;
169     if($prot !~ /http/i) {

```

```

        next;
171     }
        $link = $url;
173     }
    else {
175         if($url =~ /^\/\//) {
            $link = "$getprotocol:// $getserver$url";
177         }
        else {
179             my $nyurl=$url;
            if(length($getpath) &&
181                ($getpath !~ /\$/)) &&
                ($nyurl !~ /^\/\//)) {
183                $nyurl = "/" . $nyurl;
            }
185            $link = "$getprotocol:// $getserver/$getpath$nyurl";
        }
187     }
    my $success = &LinkWorks($link);
189    my $count = $done{$url};
    $allcount += $count;
191    print "$success $count <". $tagtype{$url}."> $link
        $url\n";
    $rooturls{$link}++;
193    if("BAD" eq $success) {$badlinks++;}
}
195 }

```