**АННОТАЦИЯ**

Данная дипломная работа посвящена разработке автоматизированной системы коммуникации субъектов учебного процесса с использованием механизмов организации социальных сетей. В качестве таких субъектов рассмотрены студенты и преподаватели образовательных учреждений.

Назначением автоматизированной системы является достижение эффективного сотрудничества между пользователями данного ресурса с помощью реализации функций общения.

Использование системы предполагается осуществлять в сети интернет, для того чтобы каждый пользователь мог зайти в нее с любого компьютера, используя свою учетную запись. Система реализовывается в виде социальной сети, в которой у каждого авторизованного пользователя есть личная страница, а основная активность пользователей разворачивается на страницах сообществ таких как: факультет, кафедра, учебная группа, дисциплины, сообщества по интересам.

Процесс разработки и проектирования состоит из разработки функциональной модели системы в виде отображения внутренних процессов при помощи построения операционных диаграмм стандарта IDEF0; разработки структуры базы данных для хранения всей необходимой информации; разработки алгоритмов и непосредственное написание программного кода для реализации функций системы.

В качестве средства для построения операционных диаграмм стандарта IDEF0 использована программа визуального моделирования BPwin. Для разработки базы данных использовалась СУБД MySQL 5.5. Основным инструментом для разработки системы является динамическая веб-платформа Groovy on Rails (Grails), предназначенная для создания приложений на динамическом языке программирования groovy.

Оглавление

[ВВЕДЕНИЕ 7](#_Toc358000284)

[ГЛАВА 1. ОПИСАНИЕ ЗАДАЧИ В ПРЕДМЕТНОЙ ОБЛАСТИ 8](#_Toc358000285)

[1.1 Назначение системы 8](#_Toc358000286)

[1.2 Описание аналогов 9](#_Toc358000287)

[1.2.1 Учебный портал ECONOMIST Факультета Экономики Российского Университета Дружбы Народов. 9](#_Toc358000288)

[1.3 Постановка задачи 13](#_Toc358000289)

[ГЛАВА 2. РАЗРАБОТКА ФУНКЦИОНАЛЬНОЙ МОДЕЛИ СИСТЕМЫ 17](#_Toc358000290)

[2.1 Методологии моделирования предметной области 17](#_Toc358000291)

[2.2 Операционные диаграммы автоматизируемых процессов 20](#_Toc358000292)

[2.2.1 Композиционная диаграмма системы 20](#_Toc358000293)

[2.2.2 Диаграмма функциональной декомпозиции 21](#_Toc358000294)

[2.2.3 Диаграмма декомпозиции функции «определение уровня доступа в систему» 22](#_Toc358000295)

[2.2.4 Диаграмма декомпозиции функции «редактирование профиля студента» 23](#_Toc358000296)

[На рисунке 8 изображена диаграмма декомпозиции функции «редактирование профиля студента». 23](#_Toc358000297)

[2.2.5 Диаграмма декомпозиции функции «управление профилем преподавателя» 24](#_Toc358000298)

[На рисунке 9 изображена диаграмма декомпозиции функции «управление профилем преподавателя». 24](#_Toc358000299)

[2.2.6 Диаграмма функционального блока «сообщества» 25](#_Toc358000300)

[На рисунке 10 изображена диаграмма функционального блока «сообщества». 25](#_Toc358000301)

[ГЛАВА 3. РАЗРАБОТКА БАЗЫ ДАННЫХ СИСТЕМЫ 26](#_Toc358000302)

[3.1 Инфологическое проектирование. Определение предметной области 26](#_Toc358000303)

[3.2 Логическое проектирование реляционной БД 32](#_Toc358000304)

[3.2.1 Преобразование ER–диаграммы в схему базы данных 32](#_Toc358000305)

[3.2.2 Составление реляционных отношений 33](#_Toc358000306)

[3.2.3 Нормализация полученных отношений 37](#_Toc358000307)

[3.2.4 Определение дополнительных ограничений целостности 41](#_Toc358000308)

[3.2.5 Описание групп пользователей и прав доступа 44](#_Toc358000309)

[3.3 Выбор СУБД 45](#_Toc358000310)

[3.3.1 Microsoft SQL Server 45](#_Toc358000311)

[3.3.2 Oracle 46](#_Toc358000312)

[3.3.3 PostgreSQL 46](#_Toc358000313)

[3.3.4 MySQL 47](#_Toc358000314)

[3.3.5 Сравнительный анализ рассматриваемых СУБД 48](#_Toc358000315)

[3.4 Реализация проекта базы данных 52](#_Toc358000316)

[ГЛАВА 4. РАЗРАБОТКА АЛГОРИТМОВ РАБОТЫ СИСТЕМЫ 56](#_Toc358000317)

[4.1 Схемы переходов между страницами 56](#_Toc358000318)

[4.2 Разработка алгоритмов 59](#_Toc358000319)

[4.2.1 Алгоритм входа в систему 59](#_Toc358000320)

[4.2.2 Алгоритм регистрации первого этапа 59](#_Toc358000321)

[4.2.3 Алгоритм регистрации второго этапа 59](#_Toc358000322)

[4.2.4 Алгоритм РМ «Администратор» 60](#_Toc358000323)

[4.2.5 Алгоритм РМ «Управляющий» 63](#_Toc358000324)

[4.2.6 Алгоритм РМ «Обычный пользователь» 66](#_Toc358000325)

[4.2.7 Алгоритм РМ «Преподаватель» 68](#_Toc358000326)

[4.2.8 Алгоритм РМ «Студент» 71](#_Toc358000327)

[4.3 Выбор средства реализации 73](#_Toc358000328)

[4.4 Описание структур кода. Особенности программирования на Grails 78](#_Toc358000329)

[ЗАКЛЮЧЕНИЕ 80](#_Toc358000330)

[СПИСОК ЛИТЕРАТУРЫ 81](#_Toc358000331)

[ПРИЛОЖЕНИЕ 83](#_Toc358000332)

# ВВЕДЕНИЕ

За последние 50 лет общество претерпело немалые изменения, как в социальном, так и в научном плане. Во многом, эти изменения связаны с бурным технологическим ростом в различных сферах человеческой деятельности (компьютерных науках, нанотехнологиях, генетике, биологии, медицине и других), а также, с постоянно увеличивающимся темпом развития человечества в целом.

В прогрессивной тенденции развития современного информационного общества важную роль занимают такие интернет проекты как социальные сети. Социальная сеть – это особый вид интернет ресурсов, на которых люди могут размещать информацию о себе, общаться друг с другом и обмениваться различными данными. С развитием технологий Web 2.0 социальные сети обрели осязаемую основу в виде порталов и веб-сервисов.

Использование социальной сети это один из наиболее эффективных способов организовать централизованное общение между группами людей, образовывающих социальную структуру. Именно поэтому автоматизированную систему было решено построить с использованием этих механизмов.

Использование такой автоматизированной системы способствует тому, чтобы вся необходимая информация находилась в одном месте, на одном ресурсе, что не допускает ситуаций получения разной информации по одному и тому же вопросу из разных источников.

Использование единого портала обеспечит возможность предоставления всегда актуальной информации, с которой должны быть осведомлены все лица, участвующие в учебном процессе.

Открытость и доступность информации способствует более эффективному сотрудничеству студентов и преподавателей, что в свою очередь повысит качество работы и тех и других.

# ГЛАВА 1. ОПИСАНИЕ ЗАДАЧИ В ПРЕДМЕТНОЙ ОБЛАСТИ

## 1.1 Назначение системы

В процессе обучения в университетах для большей успеваемости студенты изучают научные дисциплины не только во время учебных занятий, но также и в свободное от академических часов время. Вследствие этого, возникает необходимость в дополнительном общении с преподавателями за рамками учебных занятий. Организация такого общения могла бы обеспечить более эффективное взаимодействие между людьми, чтобы было легче обмениваться знаниями, опытом и навыками. В частности, речь идет об онлайн консультациях, указании на недостатки в студенческих работах, оповещениях изменений в расписании, предложениях по улучшению учебного процесса и многое другое.

В наше время многие компании и организации имеют собственные интернет-сервисы, которые демонстрируют результаты их работы и позволяют решать некоторое количество необходимых задач. Институты и университеты не исключение. В основном учебные заведения используют обычные сайты, которые представляют собой только набор статичных страниц с общей информацией об университете, факультетах и кафедрах, контактными данными и другой текстовой информацией.

В данной дипломной работе разрабатывается автоматизированная система, представленная в виде сайта, которая включает в себя, кроме статичной информации, набор функциональных возможностей для обеспечения общения между студентами, между студентами и преподавателями, а также между преподавателями.

Такую структуру можно было бы организовать и на уже существующих открытых сервисах, таких как livejournal.com или vkontakte.ru, но такой способ является слишком трудоёмким и не сможет охватить полностью все требования к студенческому порталу. Поэтому необходимо разработать автоматизированную систему, настроенную именно под учебный процесс. Это повысит удобство сопровождения этого процесса, позволит ускорить обмен информации между студентами и преподавателями и повысит качество получаемых студентами знаний.

Одним из наиболее эффективных способов организовать централизованное общение между группами людей, образовывающих социальную структуру, является использование социальной сети. Социальная сеть – это особый вид интернет ресурсов, на которых люди могут размещать информацию о себе, общаться друг с другом и обмениваться различными данными. Автоматизированная система коммуникации реализовывается с использованием механизмов социальной сети. Каждый авторизованный пользователь имеет личную страницу, на которой отображаются его личные данные (фамилия, имя, отчество, факультет, группа, информация о себе, фотография и т.п.). Со своей личной страницы пользователь может переходить на страницы сообществ, в которых при помощи тем и комментариев может осуществлять общение. Для стимуляции учебного процесса и повышения мотивации студентов к учебе разрабатывается функция присвоения рейтинга. Более подробное описание рассмотрено в разделе 1.3. Постановка задачи.

## 1.2 Описание аналогов

1.2.1 Учебный портал ECONOMIST Факультета Экономики Российского Университета Дружбы Народов.

Адрес портала: <http://economist.rudn.ru/>

На рисунке 1 представлена главная страница учебного портала студентов экономического факультета университета РУДН. Портал позволяет регистрироваться. Имеется и раздел со свободным доступом, который несет общую познавательную информацию о факультете.

Главная страница интереса не представляет, так как станицы статичны, состоят из HTML (от англ. HyperText Markup Language — «язык разметки гипертекста») кода с подключением CSS (англ. Cascading Style Sheets — каскадные таблицы стилей) стилей и JavaScript — прототипно-ориентированный сценарный язык программирования. Возможность создания страниц без динамических данных достаточно просто реализована в CMS Sapid (англ. Content management system и XML Sapiens Demonstrator — система управления контентом немодульного типа, написанная на языке PHP).

В разрабатываемой автоматизированной системе коммуникации предполагается размещение аналогичной общей информацией, но не только о факультете, но и об институте, факультетах и кафедрах. Предполагается более широкий круг пользователей, чем на рассматриваемом портале.

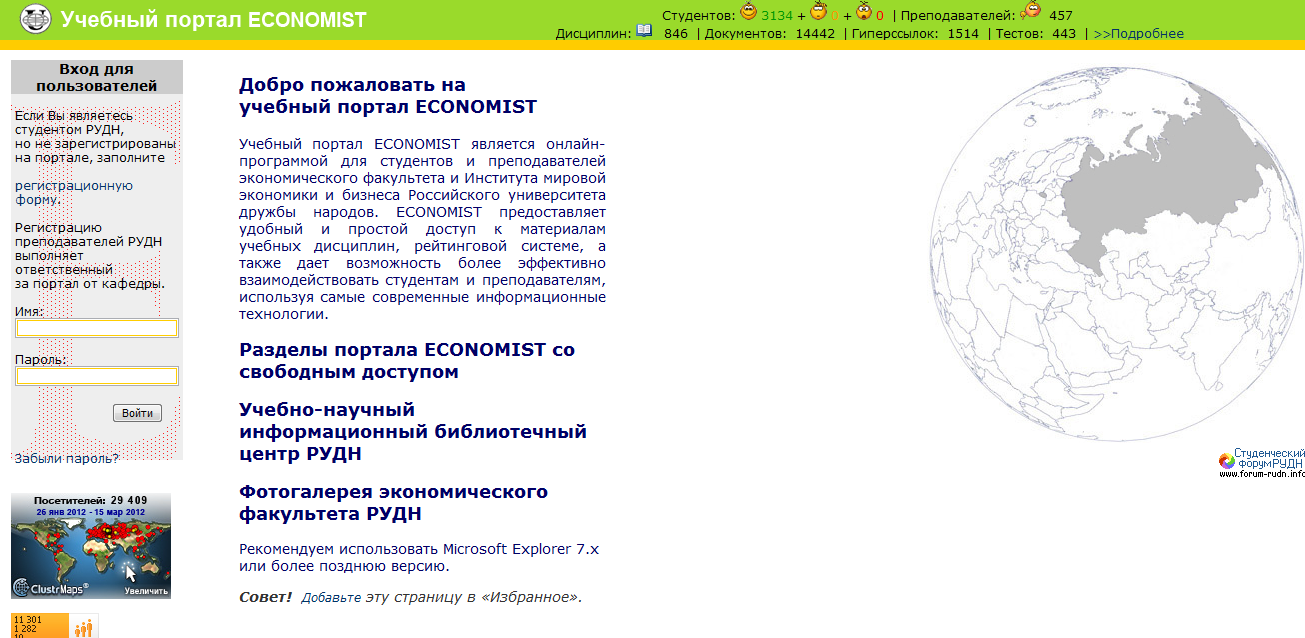


Рисунок 1 – Главная страница «Учебного портала ECONOMIST Факультета Экономики Российского Университета Дружбы Народов» до Авторизации

После регистрации в левом краю страницы появляется меню с выбором информационных страниц (рисунок 2). Зарегистрированные студенты могут просматривать объявления факультета, с прикрепленными к ним файлами, объявления спецкурсов, расписания конференций, расписание консультаций преподавателей, с возможностью записи на них. Имеется возможность посмотреть кафедры факультета, их сотрудников и объявления на них, а так же посмотреть список студентов твоей группы. В любой момент можно изменить данные пользователя.

После регистрации в системе коммуникации, пользователю необходимо дождаться подтверждения его роли администратором, прежде чем иметь возможность полностью использовать фунционал системы. До этого момента он имеет лишь функционал обычного пользователя, зарегистрированного в системе, без прав на просмотр институтских объявлений и публикацию сообщений.

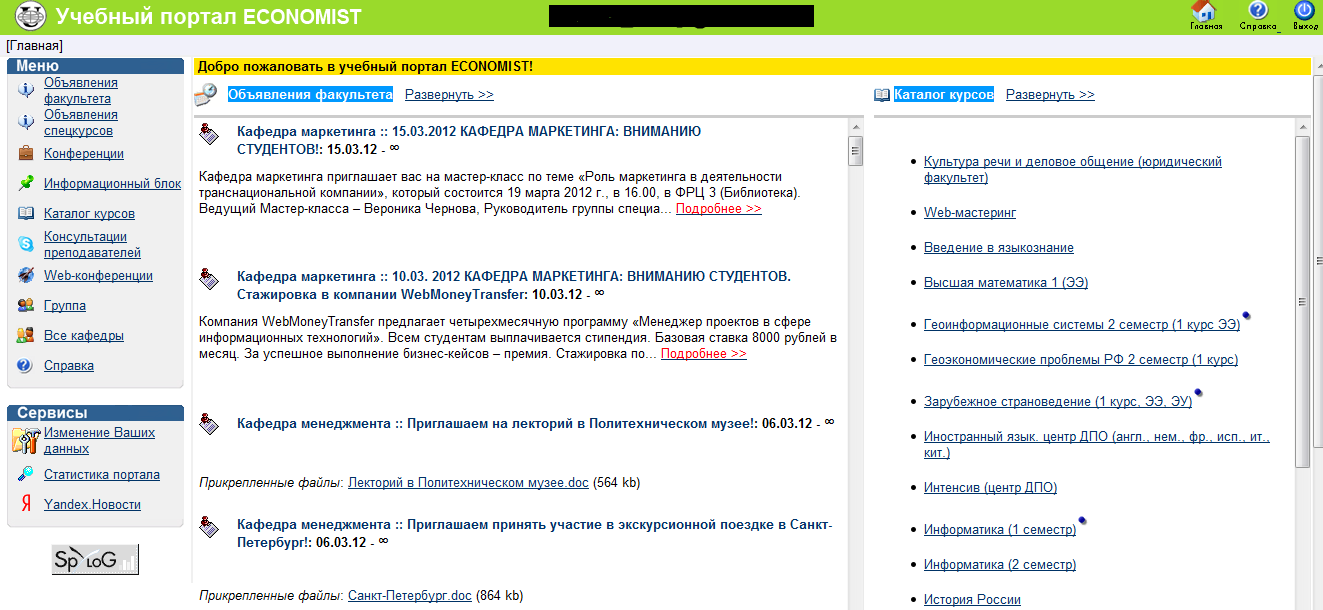


Рисунок 2 –Главная страница «Учебного портала ECONOMIST Факультета Экономики Российского Университета Дружбы Народов» после Авторизации

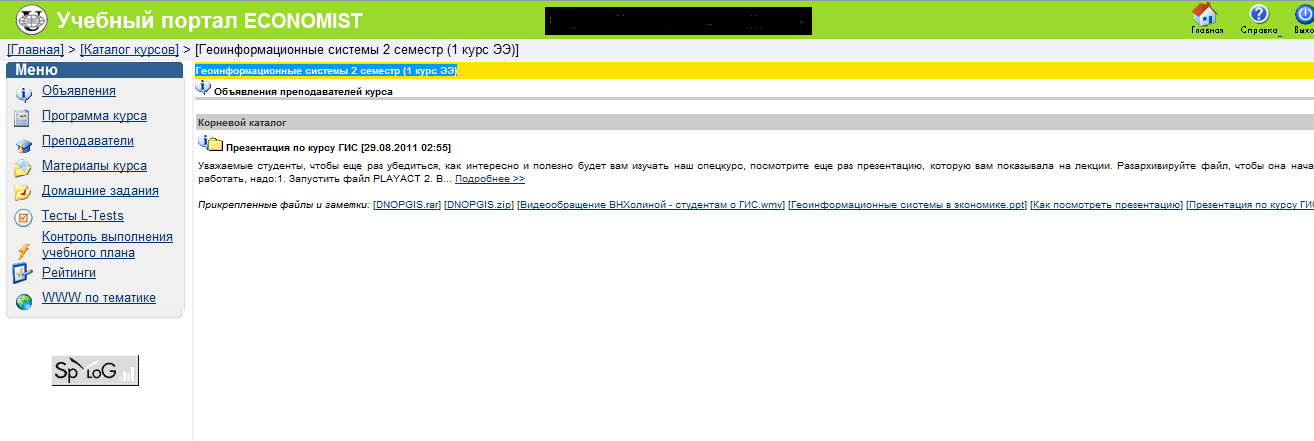


Рисунок 3 –Страница Объявлений «Учебного портала ECONOMIST Факультета Экономики Российского Университета Дружбы Народов».

Студенты РУДН могут зайти на страницу каждой дисциплины, присутствующей в его курсе обучения (рисунок 3). Здесь они могут узнать программу курса, домашние задания, рейтинги студентов, посмотреть список преподавателей и скачать учебные материалы.

Преподаватели помимо прав, присутствующих у студентов, имеют возможность выкладывать материалы и создавать объявления.

В системе коммуникации такое общение реализуется с помощью существующих сообществ (для факультетов, кафедр, дисциплин) и публикации в них обсуждений и комментариев. В обсуждениях могут участвовать как студенты, так и преподаватели. У преподавателей есть возможность выставлять студентам рейтинг, который отображается на личной странице студентов.

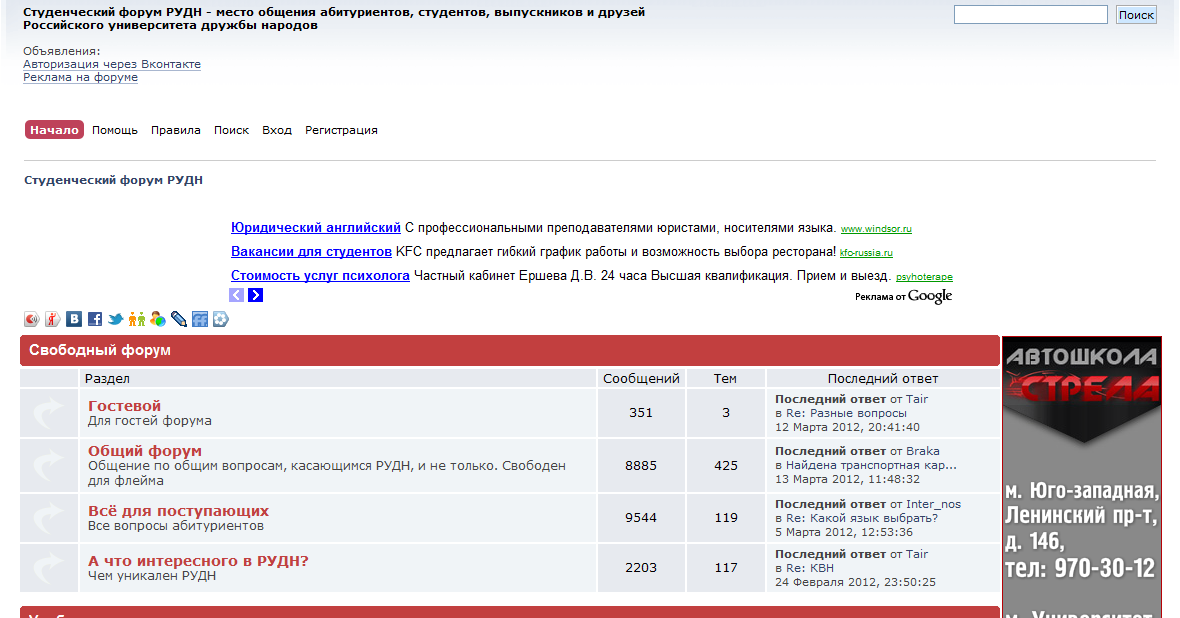


Рисунок 4 – Форум «Учебного портала ECONOMIST Факультета Экономики Российского Университета Дружбы Народов

На портале РУДН присутствует студенческий форум, на нем есть раздел для гостей (рисунок 4). В остальных разделах гости могут только просматривать темы. Добавлять и комментировать темы могут только зарегистрированные пользователи.

Имеется фотогалерея факультета, где зарегистрированные пользователи могут выкладывать фотографии.

## 1.3 Постановка задачи

Целью дипломной работы является разработка автоматизированной системы коммуникации, предназначенной для повышения эффективности организации части образовательного процесса, которая позволит студентам и преподавателям обмениваться информацией, поддерживать ее актуальность и стимулировать учебный процесс при помощи функций обмена сообщениями и присвоения рейтинга.

Назначениями автоматизированной системы являются: реализация эффективного сотрудничества между пользователями данного ресурса, отражение процесса обучения с целью повышения его эффективности.

В основе разработки данной системы лежит стремление сделать процесс обучения в университетах более эффективным, и структурно организовать взаимодействие между лицами, участвующими в учебном процессе.

При входе в систему пользователю доступна только возможность просматривать общедоступную статичную информацию («о проекте», «об институте», «о кафедре», «сообщества» и «контакты»). Для того чтобы начать полноценно пользоваться системой пользователю необходимо в ней авторизоваться. Чтобы это сделать ему необходимо ввести логин и пароль в форму авторизации. Если пользователь заходит в систему впервые, то для того чтобы начать пользоваться функциями приложения ему необходимо пройти процедуру регистрации для создания аккаунта. Создание аккаунта должно начинаться с ввода регистрационных данных, среди которых:

- Login (Логин - имя для аутентификации);

- Password (Пароль – секретный ключ для аутентификации, идёт в паре с Login);

- Email (Адрес почтового ящика – используется для смены пароля и для подтверждения регистрации);

После успешной регистрации пользователь попадает на страницу редактирования профиля, где он может заполнить поля, такие как: фамилия, имя, отчество, группа, дата рождения, информация о себе, статус, место работы и загрузить фотографию. После регистрации любой пользователь имеет статус «обычный пользователь». Чтобы изменить статус, пользователю необходимо обратиться лично к администратору своего учебного отделения и предоставить документ, удостоверяющий его личность. Обычный пользователь, в отличие от авторизованного пользователя, обладает ограниченными возможностями. Он может управлять профилем и смотреть доступные ему сообщества. После авторизации пользователю предоставляется личная страница (профиль), через которую он может осуществлять просмотр всех доступных ему страниц на данном сайте и, в зависимости от статуса, выполнять те или иные действия.

Авторизованные пользователи могут по своему усмотрению пользоваться доступными им возможностями. Студенты получают возможность просматривать страницы других студентов, преподавателей и обычных пользователей, вступать в сообщества, просматривать их содержание, публиковать в них сообщения.

Каждый студент в процессе учебы может получать рейтинг. Рейтинг – целая величина, отображающая уровень знаний студента. Начислять единицы рейтинга может преподаватель за успешно выполненные задачи. Каждый зарегистрированный пользователь может редактировать свой профиль. «Управляющие» и «администраторы» могут редактировать и удалять профили любых пользователей.

В автоматизированной системе присутствует один главный администратор и несколько «управляющих», которые обладают правами администрирования и модерирования в пределах своих полномочий. В список их задач входит инспектирование предоставленной в их распоряжение области для контроля. В качестве таких областей могут быть различные сообщества, такие как факультет, кафедра, группа и т.п. Конкретные функции руководителей учебного заведения: подтверждение статусов преподавателей и студентов в системе, управление сообществом учебного заведения, публикации в сообществах учебного заведения.

На основании описанного технологического процесса можно выделить следующие задачи, которые должна выполнять разрабатываемая система:

- Создание и управление профилями пользователей;

- Создание и управление сообществами.

Данные задачи предназначены для обеспечения следующих групп пользователей:

- РМ «Преподаватель»;

- РМ «Студент»;

- РМ «Обычный пользователь»;

- РМ «Управляющий»;

- РМ «Администратор».

Группы пользователей, входящие в состав системы, должны выполнять следующие функции:

1. РМ "Преподаватель":

- Аутентификация;

- Редактирование личной страницы;

- Действия над сообществами (вступить, покинуть, создать, редактировать);

- Публикация тем (постов) и комментариев в сообществах;

- Присуждение рейтинга студентам.

2. РМ "Студент":

- Аутентификация;

- Редактирование личной страницы;

- Действия над сообществами (вступить, покинуть, создать, редактировать);

- Публикация тем (постов) и комментариев в сообществах.

3. РМ "Обычный пользователь":

- Аутентификация;

- Редактирование личной страницы;

- Действия над сообществами (вступить, покинуть);

- Публикация тем (постов) и комментариев в сообществах.

4. РМ "Управляющий":

- Аутентификация;

- Редактирование личных страниц пользователей;

- Действия над сообществами (вступить, покинуть, создать, редактировать, удалить);

- Публикация тем (постов) и комментариев в сообществах;

- Управление рейтингом пользователей.

5. РМ "Администратор":

- Аутентификация;

- Редактирование личных страниц пользователей ;

- Действия над сообществами (вступить, покинуть, создать, редактировать, удалить);

- Публикация тем (постов) и комментариев в сообществах;

- Управление рейтингом пользователей.

# ГЛАВА 2. РАЗРАБОТКА ФУНКЦИОНАЛЬНОЙ МОДЕЛИ СИСТЕМЫ

## 2.1 Методологии моделирования предметной области

«В основе проектирования систем лежит моделирование предметной области. Для того чтобы получить проект системы, адекватный предметной области, необходимо иметь целостное, системное представление модели, которое отражает все аспекты функционирования будущей системы. При этом под моделью предметной области понимается некоторая система, имитирующая структуру или функционирование исследуемой предметной области и отвечающая основному требованию – быть адекватной этой области.

Предварительное моделирование предметной области позволяет сократить время и сроки проведения проектировочных работ и получить более эффективный и качественный проект. Без проведения моделирования предметной области велика вероятность допущения большого количества ошибок в решении стратегических вопросов, приводящих к экономическим потерям и высоким затратам на последующее перепроектирование системы.

Для отображения структурного аспекта моделей предметных областей в основном используются графические методы, которые должны гарантировать представление информации о компонентах системы. Главное требование к графическим методам документирования — простота. Графические методы должны обеспечивать возможность структурной декомпозиции спецификаций системы с максимальной степенью детализации и согласований описаний на смежных уровнях декомпозиции. IDEF0 (Function Modeling — методология функционального моделирования и графическая нотация) является наиболее распространенным стандартом построения таких моделей.

С помощью наглядного графического языка IDEF0 изучаемая система предстает перед разработчиками и аналитиками в виде набора взаимосвязанных функций (функциональных блоков — в терминах IDEF0). Как правило, моделирование средствами IDEF0 является первым этапом изучения любой системы» [1].

**История IDEF0**

«Методологию IDEF0 можно считать следующим этапом развития хорошо известного графического языка описания функциональных систем SADT (Structured Analysis and Design Technique).Несколько лет назад в России небольшим тиражом вышла одноименная книга, посвященная описанию основных принципов построения SADT-диаграмм. Исторически, IDEF0, как стандарт был разработан в 1981 году в рамках обширной программы автоматизации промышленных предприятий, которая носила обозначение ICAM (Integrated Computer Aided Manufacturing) и была предложена департаментом Военно-Воздушных Сил США. Собственно семейство стандартов IDEF унаследовало свое обозначение от названия этой программы (IDEF=ICAM DEFinition). В процессе практической реализации, участники программы ICAM столкнулись с необходимостью разработки новых методов анализа процессов взаимодействия в промышленных системах. При этом кроме усовершенствованного набора функций для описания бизнес-процессов, одним из требований к новому стандарту было наличие эффективной методологии взаимодействия в рамках “аналитик-специалист”. Другими словами, новый метод должен был обеспечить групповую работу над созданием модели, с непосредственным участием всех аналитиков и специалистов, занятых в рамках проекта.

В результате поиска соответствующих решений родилась методология функционального моделирования IDEF0. C 1981 года стандарт IDEF0 претерпел несколько незначительных изменения, в основном ограничивающего характера, и последняя его редакция была выпущена в декабре 1993 года Национальным Институтом По Стандартам и Технологиям США (NIST)» [2].

##### **Основные элементы IDEF0**

«Графический язык IDEF0 удивительно прост и гармоничен. В основе методологии лежат четыре основных понятия:

Первым из них является понятие функционального блока (Activity Box). Функциональный блок графически изображается в виде прямоугольника и олицетворяет собой некоторую конкретную функцию в рамках рассматриваемой системы.

Каждая из четырех сторон функционального блока имеет своё определенное значение (роль), при этом:

1) Верхняя сторона имеет значение “Управление” (Control);   
2) Левая сторона имеет значение “Вход” (Input);   
3) Правая сторона имеет значение “Выход” (Output);   
4) Нижняя сторона имеет значение “Механизм” (Mechanism).

Вторым “китом” методологии IDEF0 является понятие интерфейсной дуги (Arrow). Также интерфейсные дуги часто называют потоками или стрелками. Интерфейсная дуга отображает элемент системы, который обрабатывается функциональным блоком или оказывает иное влияние на функцию, отображенную данным функциональным блоком.

В зависимости от того, к какой из сторон подходит данная интерфейсная дуга, она носит название “входящей”, “исходящей” или “управляющей”. Кроме того, “источником” (началом) и “приемником” (концом) каждой функциональной дуги могут быть только функциональные блоки, при этом “источником” может быть только выходная сторона блока, а “приемником” любая из трех оставшихся.

Моделирование процессов, как правило, выполняется с помощью case-средств. К таким средствам относятся BPwin (PLATINUM technology), Silverrun (Silverrun technology), Oracle Designer (Oracle), Rational Rose (Rational Software) и др. Для моделирования процессов разрабатываемой автоматизированной системы будет использоваться case-средство BPwin, которое поддерживает методологию функционального моделирования (IDEF0)» [3] .

## 2.2 Операционные диаграммы автоматизируемых процессов

### 2.2.1 Композиционная диаграмма системы

На рисунке 5 изображена композиционная диаграмма системы.

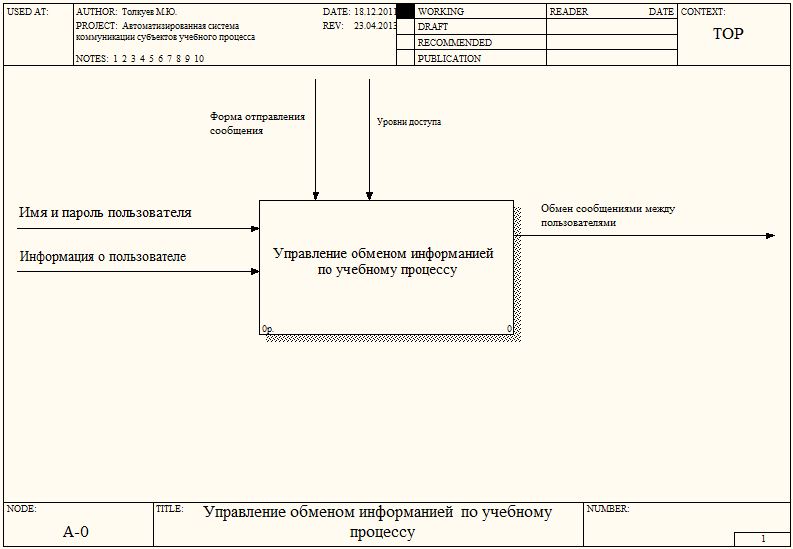


Рисунок 5 – Композиционная диаграмма системы

### 

### 2.2.2 Диаграмма функциональной декомпозиции

На рисунке 6 изображена диаграмма функциональной декомпозиции.

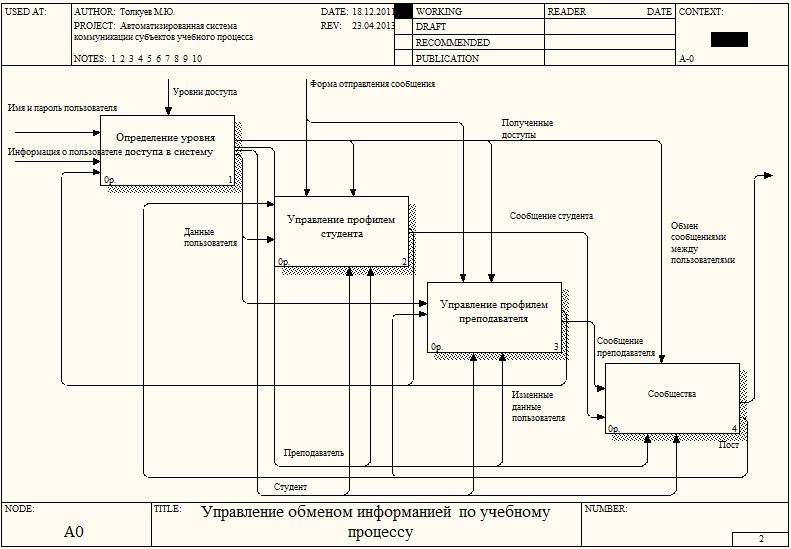


Рисунок 6 – диаграмма функциональной декомпозиции

### 2.2.3 Диаграмма декомпозиции функции «определение уровня доступа в систему»

На рисунке 7 изображена диаграмма декомпозиции функции «определение уровня доступа в систему».

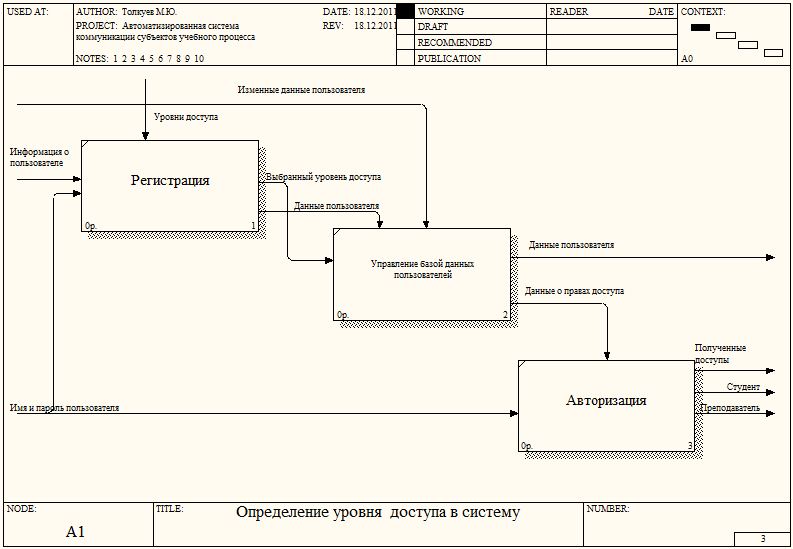
****

Рисунок 7 – Диаграмма декомпозиции фукнции «определения уровня доступа в систему»

### 2.2.4 Диаграмма декомпозиции функции «редактирование профиля студента»

### На рисунке 8 изображена диаграмма декомпозиции функции «редактирование профиля студента».

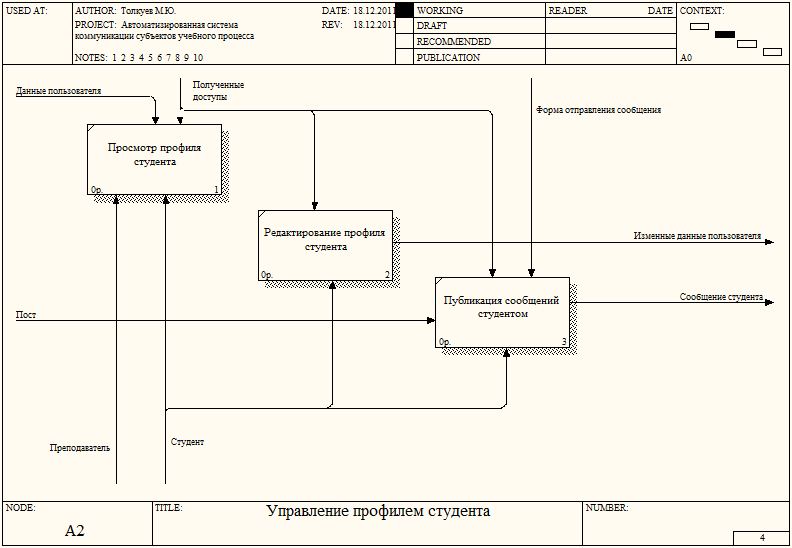
****

Рисунок 8 – Диаграмма декомпозиции функции «редактирование профиля студента»

### 2.2.5 Диаграмма декомпозиции функции «управление профилем преподавателя»

### На рисунке 9 изображена диаграмма декомпозиции функции «управление профилем преподавателя».

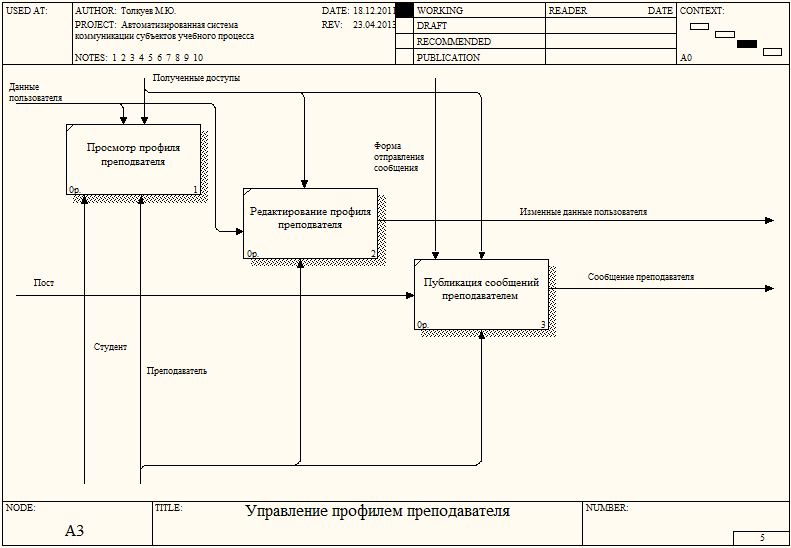
****

Рисунок 9 – Диаграмма декомпозиции функции «управление профилем преподавателя»

### 2.2.6 Диаграмма функционального блока «сообщества»

### На рисунке 10 изображена диаграмма функционального блока «сообщества».

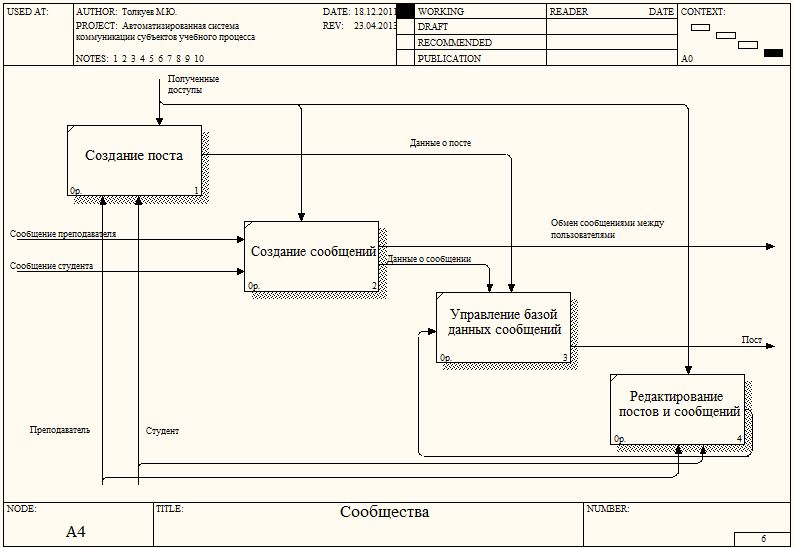


Рисунок 10 – Диаграмма функционального блока «сообщества»

# ГЛАВА 3. РАЗРАБОТКА БАЗЫ ДАННЫХ СИСТЕМЫ

## 3.1 Инфологическое проектирование. Определение предметной области

«Проектирование базы данных (БД) – одна из наиболее сложных и ответственных задач, связанных с созданием автоматизированных систем. Взаимодействие конечных пользователей с БД обычно осуществляется с помощью интерфейсного приложения, входящего в состав автоматизированной системы.

База данных создаётся для автоматизированной системы коммуникации субъектов учебного процесса для удобного, простого и эффективного общения между преподавателями и студентами, а также для общения студентов между собой. БД должна содержать данные о пользователях, сообществах и информацию, которой обмениваются пользователи.

Первый этап проектирования базы данных - построение инфологической модели предметной области. Инфологический подход не содержит формальных способов моделирования реальности, но он закладывает основы методологии проектирования базы данных. Задачей инфологического проектирования является определение предметной области системы, позволяющее изучить информационные потребности будущих пользователей.

Существуют разные подходы к инфологическому проектированию. Основные из них это:

* **Функциональный подход к проектированию БД.**

Этот метод реализует принцип "от задач" и применяется в том случае, когда известны функции некоторой группы лиц и/или комплекса задач, для обслуживания информационных потребностей которых создаётся рассматриваемая БД.

* **Предметный подход к проектированию БД.**

Предметный подход применяется в тех случаях, когда у разработчиков есть чёткое представление о самой ПО и о том, какую именно информацию они хотели бы хранить в БД, а структура запросов не определена или определена не полностью. Тогда основное внимание уделяется исследованию ПО и наиболее адекватному её отображению в БД с учётом самого широкого спектра информационных запросов к ней.

* **Проектирование с использованием метода "сущность–связь".**

Метод "сущность–связь" (Entity–Relation, ER–method) был разработан в 1976 г. П.Ченом (Chen P.P.). Он является комбинацией двух предыдущих и обладает достоинствами обоих.

ER-метод является наиболее распространённым методом проектирования БД, поэтому именно этот метод используется в данной дипломной работе» [4].

Для создания ER-модели необходимо выделить сущности предметной области:

1. Университет:

* название
* аббревиатура
* контактная информация
* описание

2. Факультет:

* название
* аббревиатура
* контактная информация
* описание

3. Кафедра:

* название
* аббревиатура
* контактная информация
* описание

4. Группа:

* название
* аббревиатура
* контактная информация
* описание

5. Сообщество:

* название
* описание
* тип

6. Профиль сообщества:

* участники
* посты

7. Пост:

* наименование
* содержание
* комментарии

8. Комментарий:

* содержание
* пост-родитель

9. Пользователь:

* логин
* пароль
* электронная почта
* контакты
* роль
* ФИО
* дата рождения
* группа
* кафедра
* направление деятельности

10. Профиль пользователя:

* личная информация (о себе)
* рейтинг
* фото

11. Студент:

* ФИО
* номер группы
* дата рождения
* логин
* пароль
* электронная почта
* контакты

12. Преподаватель:

* ФИО
* направление деятельности
* дата рождения
* логин
* пароль
* электронная почта
* контакты

В соответствии с предметной областью система строится с учетом следующих особенностей:

- Каждый преподаватель принадлежит к определенному учебному заведению, факультету, кафедре.

- Каждый студент принадлежит к определенному учебному заведению, факультету, кафедре, группе.

- В каждом учебном заведении может быть несколько факультетов.

- На каждом факультете может числиться несколько кафедр.

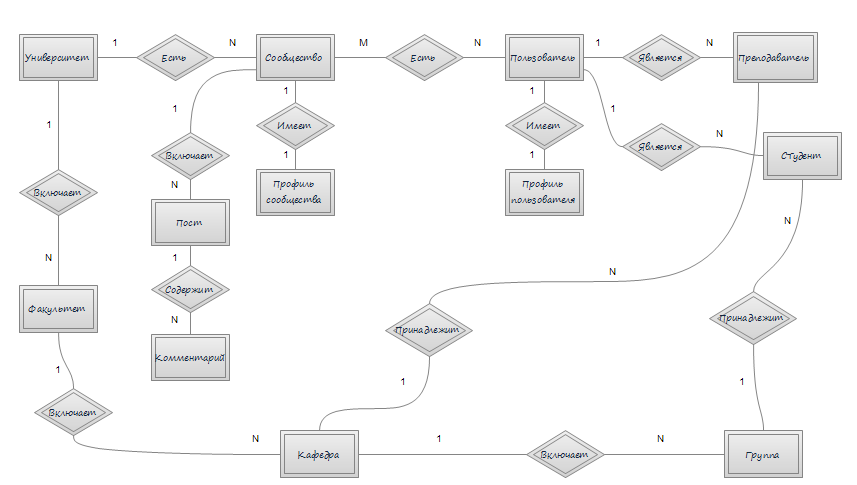
- На каждой кафедре может числиться несколько групп.

- К каждой кафедре могут быть прикреплены несколько преподавателей.

- В каждой группе числится несколько студентов.

ER-диаграмма предметной области представлена на рисунке 11.

|  |
| --- |
| Рисунок 11 – ER-диаграмма предметной области |



## 3.2 Логическое проектирование реляционной БД

«На этапе логического проектирования инфологическая модель ПО, представленная в виде ER-диаграммы, преобразуется в логическую (концептуальную схему БД). Результатом выполнения этапа логического проектирования являются схемы БД концептуального и внешнего уровней архитектуры, составленные на языке определения данных (DDL, Data Definition Language) выбранной СУБД (Система управления базами данных)» [5].

### 3.2.1 Преобразование ER–диаграммы в схему базы данных

На рисунке 12 представлена схема реляционной базы данных, полученная из ER–диаграммы.

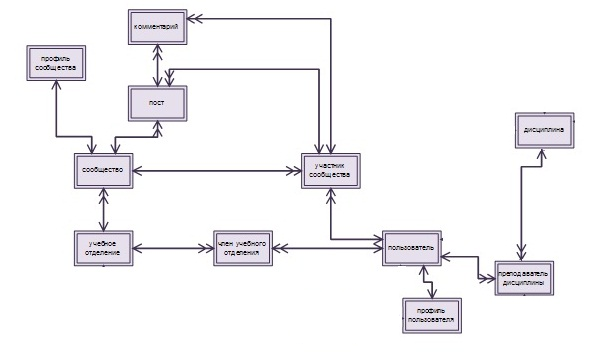
****

Рисунок 12 – Схема реляционной базы данных

### 3.2.2 Составление реляционных отношений

Таблица 1 – Схема отношения УЧЕБНОЕ ОТДЕЛЕНИЕ (STUDYING DIVISION)

|  |  |  |  |
| --- | --- | --- | --- |
| *Содержание поля* | *Имя поля* | *Тип, длина* | *Примечания* |
| Название | sd\_name | С(80) | обязательноеполе |
| Аббревиатура | sd\_short\_name | C(8) | обязательное поле |
| Контактная информация | sd\_contact\_info | C(40) | необязательное многозначное поле |
| Описание | sd\_describe | C(300) | необязательное поле |
| Учебное отделение\_id | sd\_id | N(4) | **суррогатный первичный ключ** |

Таблица 2 – Схема отношения СООБЩЕСТВО (COMMUNITY)

|  |  |  |  |
| --- | --- | --- | --- |
| *Содержание поля* | *Имя поля* | *Тип, длина* | *Примечания* |
| Название | c\_name | С(80) | обязательноеполе |
| Описание | c\_describe | C(300) | обязательное поле |
| Тип | c\_type | C(16) | обязательное поле |
| Учебное отделение\_id | c\_division | N(4) | внешний ключ (к STUDYING DIVISION) |
| Сообщество\_id | c\_id | N(6) | **суррогатный первичный ключ** |

Таблица 3 – Схема отношения ПРОФИЛЬ СООБЩЕСТВА (COMMUNITY PROFILE)

|  |  |  |  |
| --- | --- | --- | --- |
| *Содержание поля* | *Имя поля* | *Тип, длина* | *Примечания* |
| Участники | cp\_member | С(80) | обязательноеполе |
| Сообщество\_id | cp\_id | N(6) | внешний ключ (к COMMUNITY) |

Таблица 4 – Схема отношения ПОСТ (POST)

|  |  |  |  |
| --- | --- | --- | --- |
| *Содержание поля* | *Имя поля* | *Тип, длина* | *Примечания* |
| Наименование | p\_name | С(80) | обязательноеполе |
| Содержание | p\_content | C(300) | обязательное поле |
| Комментарии\_id | p\_comment | N(6) | внешний ключ (к COMMENT) |
| Участник сообщества\_id | p\_member | N(6) | внешний ключ (к MEMBER OF COMMUNITY) |
| Сообщество\_id | p\_id | N(6) | внешний ключ (к COMMUNITY) |
| Пост\_id | p\_id | N(6) | **суррогатный первичный ключ** |

Таблица 5 – Схема отношения КОММЕНТАРИЙ (COMMENT)

|  |  |  |  |
| --- | --- | --- | --- |
| *Содержание поля* | *Имя поля* | *Тип, длина* | *Примечания* |
| Содержание | com\_content | С(200) | обязательное поле |
| Пост\_id | com\_post | N(6) | внешний ключ (к POST) |
| Комментарий\_id | com\_id | N(6) | **суррогатный первичный ключ** |

Таблица 6 – Схема отношения ПОЛЬЗОВАТЕЛЬ (USER)

|  |  |  |  |
| --- | --- | --- | --- |
| *Содержание поля* | *Имя поля* | *Тип, длина* | *Примечания* |
| Логин | u\_login | С(20) | обязательное уникальноеполе |
| Пароль | u\_password | C(16) | обязательное поле |
| Электронная почта | u\_e-mail | C(40) | обязательное уникальное поле |
| Контакты | u\_conacts | C(300) | необязательное многозначное поле |
| Роль | u\_role | C(30) | обязательное поле |
| ФИО | u\_fio | C(70) | обязательное поле |
| Дата рождения | u\_birthday | N(8) | необязательное поле |
| Группа | u\_group | N(4) | необязательное поле |
| Кафедра | u\_department | N(4) | необязательное поле |
| Направление деятельности | u\_work | C(20) | необязательное поле |
| Пользователь\_id | u\_id | N(6) | **суррогатный первичный ключ** |

Таблица 7 – Схема отношения УЧАСТНИК СООБЩЕСТВА (MEMBER OF COMMUNITY)

|  |  |  |  |
| --- | --- | --- | --- |
| *Содержание поля* | *Имя поля* | *Тип, длина* | *Примечания* |
| Пользователь\_id | mc\_user | N(6) | внешний ключ (к USER) |
| Cообщество\_id | mc\_community | N(6) | внешний ключ (к COMMUNITY) |

Таблица 8 – Схема отношения ПРОФИЛЬ ПОЛЬЗОВАТЕЛЯ (USER PROFILE)

|  |  |  |  |
| --- | --- | --- | --- |
| *Содержание поля* | *Имя поля* | *Тип, длина* | *Примечания* |
| Личная информация | up\_info | С(300) | необязательноеполе |
| Рейтинг | up\_rating | N(3) | необязательное поле |
| Фото | up\_foto | C(20) | необязательное поле |
| Пользователь\_id | up\_user | N(6) | внешний ключ (к USER) |

Таблица 9 – Схема отношения ДИСЦИПЛИНА (DISCIPLINE)

|  |  |  |  |
| --- | --- | --- | --- |
| *Содержание поля* | *Имя поля* | *Тип, длина* | *Примечания* |
| Название | dis\_name | С(20) | обязательноеполе |
| Дисциплина\_id | dis\_id | N(5) | **суррогатный первичный ключ** |

Таблица 10 – Схема отношения ПРЕПОДАВАТЕЛЬ ДИСЦИПЛИНЫ (TEACHER DISCIPLINE)

|  |  |  |  |
| --- | --- | --- | --- |
| *Содержание поля* | *Имя поля* | *Тип, длина* | *Примечания* |
| Дисциплина\_id | td\_discipline | N(5) | внешний ключ (к DISCIPLINE) |
| Преподаватель\_id | td\_teacher | N(6) | внешний ключ (к USER) |

Таблица 11 – Схема отношения ЧЛЕН УЧЕБНОГО ОТДЕЛЕНИЯ (SD\_MEMBER)

|  |  |  |  |
| --- | --- | --- | --- |
| *Содержание поля* | *Имя поля* | *Тип, длина* | *Примечания* |
| Учебное отделение\_id | sdm\_division | N(4) | внешний ключ (к STUDYING DIVISION) |
| Пользователь\_id | sdm\_user | N(4) | внешний ключ (к USER ) |

«После составления концептуальной (логической) схемы БД необходимо проверить её на отсутствие аномалий модификации данных. Эти аномалии обусловлены ограниченностью структуры РМД (Реляционная модель данных). Различают три вида аномалий: аномалии обновления, удаления и добавления. Аномалия обновления может возникнуть в том случае, когда информация дублируется. Другие аномалии возникают тогда, когда две и более сущности объединены в одно отношение.

В рамках реляционной модели данных Э.Ф. Коддом был разработан аппарат нормализации отношений и предложен механизм, позволяющий любое отношение преобразовать к третьей нормальной форме. Нормализация схемы отношения выполняется путём декомпозиции схемы» [4].

### 3.2.3 Нормализация полученных отношений

Таблица 12 – Схема отношения УЧЕБНОЕ ОТДЕЛЕНИЕ (STUDYING DIVISION)

|  |  |  |  |
| --- | --- | --- | --- |
| *Содержание поля* | *Имя поля* | *Тип, длина* | *Примечания* |
| Аббревиатура | sd\_short\_name | C(8) | обязательное поле |
| Описание | sd\_describe | C(300) | необязательное поле |
| Название | sd\_name | С(80) | обязательное поле |
| Учебное отделение\_id | sd\_id | N(4) | **суррогатный первичный ключ** |

Таблица 13 – Схема отношения КОНТАКТЫ (CONTACTS)

|  |  |  |  |
| --- | --- | --- | --- |
| *Содержание поля* | *Имя поля* | *Тип, длина* | *Примечания* |
| Контактная информация | cont\_contact\_info | C(40) | необязательное многозначное поле |
| Учебное отделение\_id | cont\_division | N(4) | внешний ключ (к STUDYING DIVISION) |

Таблица 14 – Схема отношения СООБЩЕСТВО (COMMUNITY)

|  |  |  |  |
| --- | --- | --- | --- |
| *Содержание поля* | *Имя поля* | *Тип, длина* | *Примечания* |
| Название | c\_name | С(80) | обязательноеполе |
| Описание | c\_describe | C(300) | обязательное поле |
| Тип | c\_type | C(16) | обязательное поле |
| Учебное отделение\_id | c\_division | N(4) | внешний ключ (к STUDYING DIVISION) |
| Сообщество\_id | c\_id | N(6) | **суррогатный первичный ключ** |

Таблица 15 – Схема отношения ПРОФИЛЬ СООБЩЕСТВА(COMMUNITY PROFILE)

|  |  |  |  |
| --- | --- | --- | --- |
| *Содержание поля* | *Имя поля* | *Тип, длина* | *Примечания* |
| Участники | cp\_member | С(80) | обязательноеполе |
| Сообщество\_id | cp\_id | N(6) | внешний ключ (к COMMUNITY) |

Таблица 16 – Схема отношения ПОСТ (POST)

|  |  |  |  |
| --- | --- | --- | --- |
| *Содержание поля* | *Имя поля* | *Тип, длина* | *Примечания* |
| Наименование | p\_name | С(80) | обязательноеполе |
| Содержание | p\_content | C(300) | обязательное поле |
| Комментарии\_id | p\_comment | N(6) | внешний ключ (к COMMENT) |
| Участник сообщества\_id | p\_member | N(6) | внешний ключ (к MEMBER OF COMMUNITY) |
| Сообщество\_id | p\_c\_id | N(6) | внешний ключ (к COMMUNITY) |
| Пост\_id | p\_id | N(6) | **суррогатный первичный ключ** |

Таблица 17 – Схема отношения КОММЕНТАРИЙ (COMMENT)

|  |  |  |  |
| --- | --- | --- | --- |
| *Содержание поля* | *Имя поля* | *Тип, длина* | *Примечания* |
| Содержание | com\_content | С(200) | обязательное поле |
| Пост\_id | com\_post | N(6) | внешний ключ (к POST) |
| Комментарий\_id | com\_id | N(6) | **суррогатный первичный ключ** |

Таблица 18 – Схема отношения ПОЛЬЗОВАТЕЛЬ (USER)

|  |  |  |  |
| --- | --- | --- | --- |
| *Содержание поля* | *Имя поля* | *Тип, длина* | *Примечания* |
| Пароль | u\_password | C(16) | обязательное поле |
| Логин | u\_login | С(20) | обязательное уникальноеполе |
| Электронная почта | u\_e\_mail | C(40) | обязательное уникальное поле |
| Роль | u\_role | C(30) | обязательное поле |
| ФИО | u\_fio | C(70) | обязательное поле |
| Дата рождения | u\_birthday | N(8) | необязательное поле |
| Группа | u\_group | N(4) | необязательное поле |
| Кафедра | u\_department | N(4) | необязательное поле |
| Направление деятельности | u\_work | C(20) | необязательное поле |
| Пользователь\_id | u\_id | N(6) | **суррогатный первичный ключ** |

Таблица 19 – Схема отношения КОНТАКТЫ ПОЛЬЗОВАТЕЛЯ (USER CONTACTS)

|  |  |  |  |
| --- | --- | --- | --- |
| *Содержание поля* | *Имя поля* | *Тип, длина* | *Примечания* |
| Контакты | uc\_conacts | C(300) | необязательное многозначное поле |
| Пользователь\_id | uc\_user | N(6) | внешний ключ (к USER) |

Таблица 20 – Схема отношения УЧАСТНИК СООБЩЕСТВА (MEMBER OF COMMUNITY)

|  |  |  |  |
| --- | --- | --- | --- |
| *Содержание поля* | *Имя поля* | *Тип, длина* | *Примечания* |
| Пользователь\_id | mc\_user | N(6) | внешний ключ (к USER) |
| Сообщество\_id | mc\_community | N(6) | внешний ключ (к COMMUNITY) |
| Участ\_сообщ\_id | mc\_id | N(6) | **суррогатный первичный ключ** |

Таблица 21 – Схема отношения ПРОФИЛЬ ПОЛЬЗОВАТЕЛЯ (USER PROFILE)

|  |  |  |  |
| --- | --- | --- | --- |
| *Содержание поля* | *Имя поля* | *Тип, длина* | *Примечания* |
| Личная информация | up\_info | С(300) | необязательноеполе |
| Настроение (статус) | up\_status | C(20) | необязательное поле |
| Фото | up\_foto | C(20) | необязательное поле |
| Пользователь\_id | up\_id | N(6) | внешний ключ (к USER) |

Таблица 22 – Схема отношения РЕЙТИНГ ПОЛЬЗОВАТЕЛЯ (USER RATING)

|  |  |  |  |
| --- | --- | --- | --- |
| *Содержание поля* | *Имя поля* | *Тип, длина* | *Примечания* |
| Рейтинг | upr\_rating | N(3) | необязательное поле |
| Пользователь\_id | upr\_user | N(6) | внешний ключ (к USER PROFILE) |

Таблица 23 – Схема отношения ДИСЦИПЛИНА (DISCIPLINE)

|  |  |  |  |
| --- | --- | --- | --- |
| *Содержание поля* | *Имя поля* | *Тип, длина* | *Примечания* |
| Название | dis\_name | С(20) | обязательноеполе |
| Учебный курс\_id | dis\_course | N(5) | внешний ключ (к TRAINING COURSE) |
| Дисциплина\_id | dis\_id | N(5) | **суррогатный первичный ключ** |

Таблица 24 – Схема отношения ПРЕПОДАВАТЕЛЬ ДИСЦИПЛИНЫ (TEACHER DISCIPLINE)

|  |  |  |  |
| --- | --- | --- | --- |
| *Содержание поля* | *Имя поля* | *Тип, длина* | *Примечания* |
| Дисциплина\_id | td\_discipline | N(5) | внешний ключ (к DISCIPLINE) |
| Преподаватель\_id | td\_teacher | N(6) | внешний ключ (к USER) |

Таблица 25 – Схема отношения ЧЛЕН УЧЕБНОГО ОТДЕЛЕНИЯ (SD\_MEMBER)

|  |  |  |  |
| --- | --- | --- | --- |
| *Содержание поля* | *Имя поля* | *Тип, длина* | *Примечания* |
| Учебное отделение\_id | sdm\_division | N(4) | внешний ключ (к STUDYING DIVISION) |
| Пользователь\_id | sdm\_user | N(4) | внешний ключ (к USER ) |

### 3.2.4 Определение дополнительных ограничений целостности

1. Атрибут ТИП отношения СООБЩЕСТВО может принимать одно из следующих значений "научное", "университет","факультет", "кафедра", "группа", "по интересам".

2. Атрибут ПАРОЛЬ отношения ПОЛЬЗОВАТЕЛЬ должен включать в себя минимум 6 знаков.

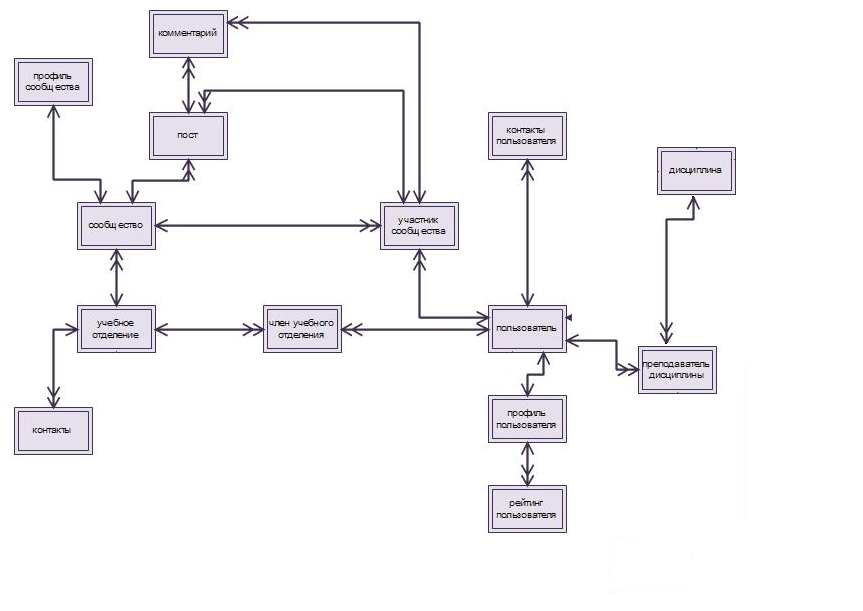
3. Атрибут РОЛЬ отношения ПОЛЬЗОВАТЕЛЬ может принимать одно из следующих значений "преподаватель", "студент", "обычный пользователь" , "управляющий", "администратор".

4. Атрибут РЕЙТИНГ отношения РЕЙТИНГ ПОЛЬЗОВАТЕЛЯ не должен превышать значения "100".

Все эти ограничения выполняются программно.

Окончательная схема базы данных представлена на рисунке 13.

|  |
| --- |
| Рисунок 13 – Окончательная схема базы данных автоматизированной системы коммуникации |



### 3.2.5 Описание групп пользователей и прав доступа

S – чтение данных (select); I – добавление данных (insert); U – модификация данных (update); D – удаление данных(delete).

Таблица 26 – Описание групп пользователей и прав доступа

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Таблицы | Группы пользователей (роли) | | | | |
| Администратор | Руководитель учебного заведения | Обычный пользователь | Преподаватель | Студент |
| сообщество | SUID | SIUD | S | SIUD | SIUD |
| пост | SUID | SIUD | SIUD | SIUD | SIUD |
| комментарий | SUID | SIUD | SIUD | SIUD | SIUD |
| профиль сообщества | SUID | S | S | S | S |
| учебное отделение | SUID | S | S | S | S |
| название | SUID | S | S | S | S |
| контакты | SUID | S | S | S | S |
| член учебного отделения | SUID | S | S | S | S |
| пользователь | SUID | S | S | S | S |
| участник сообщества | SUID | S | S | S | S |
| почта пользователя | SUID |  |  |  |  |
| логин пользователя | SUID |  |  |  |  |
| контакты пользователя | SUID | S | S | S | S |
| рейтинг пользователя | SUID | S | S | S | S |
| профиль пользователя | SUID | S | S | S | S |
| дисциплина | SUID | S |  | S | S |

## 3.3 Выбор СУБД

«Выбор СУБД является одним из важнейших моментов в разработке проекта БД, так как он принципиальным образом влияет на весь процесс проектирования БД и реализации системы.

Теоретически при осуществлении этого выбора нужно принимать во внимание десятки факторов. Но на практике разработчики руководствуются лишь собственной интуицией и несколькими наиболее важными критериями, к которым, в частности, относятся:

* тип модели данных, которую поддерживает данная СУБД, адекватность модели данных структуре рассматриваемой предметной области;
* характеристики производительности СУБД;
* запас функциональных возможностей для дальнейшего развития системы;
* степень оснащенности СУБД инструментарием для персонала администрирования данными;
* удобство и надежность СУБД в эксплуатации;
* стоимость СУБД и дополнительного программного обеспечения» [5].

На сегодня известно большое число различных серверов баз данных SQL (англ. Structured Query Language — «язык структурированных запросов»). Рассмотрим более подробно следующие четыре ведущих серверных СУБД -Microsoft SQL Server, Oracle, PostgreSQL и MySQL, проведем их сравнение и выберем одну из них.

### 3.3.1 Microsoft SQL Server

 «Система управления реляционными базами данных (СУБД), разработанная корпорацией Microsoft. Основной используемый язык запросов — Transact-SQL, создан совместно Microsoft и Sybase. Transact-SQL является реализацией стандарта ANSI/ISO по структурированному языку запросов (SQL) с расширениями. Используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия» [6].

### 3.3.2 Oracle

«Современная СУБД Oracle это мощный программный комплекс, позволяющий создавать приложения любой степени сложности. Ядром этого комплекса является база данных, хранящая информацию, количество которой за счет предоставляемых средств масштабирования практически безгранично. ORACLE поддерживает самые большие базы данных, потенциального размера до сотен гигабайт.

ORACLE удовлетворяет промышленно принятым стандартам по языку доступа к данным, операционным системам, интерфейсам с пользователем и сетевым протоколам. Работает под Sun Solaris, Linux, Windows и других операционных систем» [7].

### 3.3.3 PostgreSQL

«В профессиональной среде коротко называется «постгрес» — свободная объектно-реляционная система управления базами данных. Система PostgreSQL основана на ядре, созданном множеством разработчиков.

Существует в реализациях для множества UNIX-like платформ, включая AIX, различные BSD-системы, HP-UX, IRIX, Linux, Mac OSX, Solaris/OpenSolaris, Tru64,

QNX, а также для Windows.

PostgreSQL базируется на языке SQL и поддерживает многие из возможностей стандарта SQL:2003 (ISO/IEC 9075)

Имеет BSD лицензию, существует так же коммерческая лицензия (которая предполагает техническую поддержку).

Поддерживает транзакции, подзапросы, триггеры, представления, внешние ключи, пользовательские типы и их наследование. Поддержка языка запросов PL/pgSQL, который очень похож на PL/SQL Oracle» [8].

### 3.3.4 MySQL

«СУБД MySQL является одной из самых известных, надежных и быстрых из всего семейства существующих СУБД. Исходные коды скомпилированы под множество платформ.

 Разработку и поддержку MySQL осуществляет корпорация Oracle, получившая права на торговую марку вместе с поглощённой Sun Microsystems, которая ранее приобрела шведскую компанию MySQL AB. Продукт распространяется как под GNU General Public License, так и под собственной коммерческой лицензией.

СУБД является идеальным решением для малых и средних приложений. MySQL – сервер является бесплатным для некоммерческого использования. MySQL – это компактный многопоточный сервер SQL БД, широко распространенный в качестве SQL – движка сайтов благодаря удачному сочетанию пользовательских свойств, открытым исходным кодам и хорошей технической поддержке. Исходный язык MySQL – C. СУБД MySQL поддерживает язык запросов SQL в стандарте ANSI 92, и, кроме этого, имеет множество расширений к этому стандарту, которых нет ни в одной другой СУБД» [9].

### 3.3.5 Сравнительный анализ рассматриваемых СУБД

Таблица 27 – Сравнение СУБД по общей информации

|  | **Компания** | **Дата первого публичного релиза** | **Последняя стабильная версия** | **Последняя дата релиза** | **Лицензия на программное обеспечение** |
| --- | --- | --- | --- | --- | --- |
| **Microsoft SQL Server** | Microsoft | 1989 | 2012 (v11) |  | Патентованный  (Proprietary) |
| **MySQL** | Sun Microsystems(в настоящее время корпорации Oracle) | 1995-11 | 5.5.29 | 2012-12-21 | GPL или Патентованный |
| **Oracle** | Корпорация Oracle | 1979-11 | 11g Release 2 | 2009-09 | Патентованный |
| **PostgreSQL** | Всемирная группа разработчиков PostgreSQL | 1989-06 | 9.2.3 | 2013-02-07 | Лицензия PostgreSQL (a liberal Open Source license) |

Таблица 28 – Сравнение СУБД по поддержке операционных систем

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Windows** | **Mac OS X** | **Linux** | **BSD** | **UNIX** | **AmigaOS** | **Symbian** | **z/OS** |
| **Oracle** | Есть | Есть | Есть | Нет | Есть | Нет | Нет | Есть |
| **MysQL** | Есть | Есть | Есть | Есть | Есть | Есть | Есть | Есть |
| **SQL Server** | Есть | Нет | Нет | Нет | Нет | Нет | Нет | Нет |
| **PostgreSQL** | Есть | Есть | Есть | Есть | Есть | Нет | Нет | Нет |

Таблица 29 – Сравнение СУБД по основным функциям

|  | **ACID** | **Целостность ссылочных данных** | **Транзакции** | **Unicode** | **Интерфейс** |
| --- | --- | --- | --- | --- | --- |
| **Microsoft SQL Server** | Да | Да | Да | Да | GUI и SQL |
| **MySQL** | Да | Частичное | Да | Да | GUI  и SQL |
| **Oracle** | Да | Да | Да | Да | API и GUI и SQL |
| **PostgreSQL** | Да | Да | Да | Да | API и GUI и SQL |

Таблица 30 – Сравнение СУБД по пределам и ограничениям

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **MS SQL Server** | **Oracle** | **PostgreSQL** | **MysQL** |
| **Максимальный размер БД** | 524272 ТБ (32 767 файлов \* 16 ТБ Максимальный размер файла) | Unlimited (4 Гб \* размер блока для каждого табличного пространства) | Неограниченный | Неограниченный |
| **Максимальный размер таблицы** | 524272 ТБ | 4 Гб \* размер блока для каждого табличного пространства | 32 ТБ | MyISAM хранения пределов: 256 ТБ; пределы Innodb хранения: 64 ТБ |
| **Максимальный размер строки** | 8060 байт | 8 Кб | 1,6 ТБ | 64 Кб |
| **Максимум столбцов в строке** | 30000 | 1000 | 250-1600 в зависимости от типа | 4096 |
| **Максимальный размер типа CHAR** | 2 Гб | 4000 B | 1 Гб | 64 Кб |
| **Максимальный размер типа Number** | 126 бит | 126 бит | Неограниченный | 64 бит |

Таблица 31 – Сравнение СУБД по возможностям базы данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **MS SQL Server** | **Oracle** | **PostgreSQL** | **MysQL** |
| **Union** | Да | Да | Да | Да |
| **Intersect** | Да | Да | Да | Нет |
| **Except** | Да | Да | Да | Нет |
| **Inner joins** | Да | Да | Да | Да |
| **Outer joins** | Да | Да | Да | Да |
| **Inner selects** | Да | Да | Да | Да |
| **Merge joins** | Да | Да | Да |  |
| **BlobsandClobs** | Да | Да | Да | Да |
| **Common Table Expressions** | Да | Да | Да | Нет |
| **Windowing Functions** | Да | Да | Да | Нет |
| **Parallel Query** | Да | Да | Нет | Нет |

Таблица 32 – Сравнение СУБД по возможностям базы данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **MS SQL Server** | **Oracle** | **PostgreSQL** | **MysQL** |
| **Type system** | Static | Static + Dynamic | Static | Static |
| **Integer** | TINYINT, SMALLINT, INT, BIGINT | NUMBER | SMALLINT (16-bit), INTEGER (32-bit), BIGINT (64-bit) | TINYINT (8-bit), SMALLINT (16-bit), MEDIUMINT (24-bit), INT (32-bit), BIGINT (64-bit) |
| **Floating point** | FLOAT, REAL | BINARY\_FLOAT, BINARY\_DOUBLE | REAL (32-bit), DOUBLE PRECISION (64-bit) | FLOAT (32-bit), DOUBLE (aka REAL) (64-bit) |
| **Decimal** | NUMERIC, DECIMAL, SMALLMONEY, MONEY | NUMBER | DECIMAL, NUMERIC | DECIMAL |
| **String** | CHAR, VARCHAR, TEXT, NCHAR, NVARCHAR, NTEXT | CHAR, VARCHAR2, CLOB, NCLOB, NVARCHAR2, NCHAR, LONG (deprecated) | CHAR, VARCHAR, TEXT | CHAR, BINARY, VARCHAR, VARBINARY, TEXT, TINYTEXT, MEDIUMTEXT, LONGTEXT |
| **Binary** | BINARY, VARBINARY, IMAGE, FILESTREAM | BLOB, RAW, LONG RAW (deprecated), BFILE | BYTEA | TINYBLOB, BLOB, MEDIUMBLOB, LONGBLOB |
| **Date/Time** | DATE, DATETIMEOFFSET, DATETIME2, SMALLDATETIME, DATETIME, TIME | DATE, TIMESTAMP (with/without TIMEZONE), INTERVAL | DATE, TIME (with/without TIMEZONE), TIMESTAMP (with/without TIMEZONE), INTERVAL | DATETIME, DATE, TIMESTAMP, YEAR |
| **Boolean** | BIT | N/A | BOOLEAN | BIT(1), BOOLEAN (aka BOOL) = synonym for TINYINT |
| **Other** | CURSOR, TIMESTAMP, HIERARCHYID, UNIQUEIDENTIFIER, SQL\_VARIANT, XML, TABLE | SPATIAL, IMAGE, AUDIO, VIDEO, DICOM, XMLType | ENUM, POINT, LINE, LSEG, BOX, PATH, POLYGON, CIRCLE, CIDR, INET, MACADDR, BIT, UUID, XML, arrays | ENUM, SET, GIS data types (Geometry, Point, Curve, LineString, Surface, Polygon, GeometryCollection, MultiPoint, MultiCurve, MultiLineString, MultiSurface, MultiPolygon) |

По итогам сравнения было принято решение выбрать СУБД MySQL v.5.5. MySQL поддерживает очень большой список платформ. Начиная с пятой версии MySQL поддерживает хранимые процедуры и функции, обработчики ошибок, курсоры, триггеры, представления. MySQL очень популярен среди Web разработчиков по причине его высокой скорости и простоты использования. Освоение, к примеру, ORACLE является более трудоемким процессом.

Oracle – исходя из своей широкой функциональности, предназначено для больших проектов. Для реализации задач разрабатываемой системы будет вполне достаточно функций и масштабности MySQL. Несмотря на преимущество Oracle над MySQL в быстродействии, малые массивы данных (до 100 000 записей) MySQL обрабатывает быстрее, чем Oracle.

MS SQL Server уступает другим рассматриваемым СУБД по двум важным показателям: программируемость и средства работы.

При разработке клиентских БД приложений на основе языков Java, HTML часто возникает проблема недостаточности программных средств SQL Server и пользоваться этой СУБД будет труднее.

MS SQL Server и Oracle являются коммерческими продуктами, в то время как MySQL и PostgreSQL являются бесплатным программным обеспечением.

MySQL имеет ряд преимуществ над PostgreSQL. Недостатки PostgreSQL по сравнению с MySQL:

* Необходима чистка после работы команд UPDATE и DELETE командой VACUUM. Это затрудняет использование PostgreSQL в постоянно работающих системах.
* Наличие только транзакционных таблиц.
* Значительно более медленная работа команд INSERT, DELETE и UPDATE.

MySQL используется в большинстве веб-проектов. Он не лучший по всем техническим характеристикам, но его достаточно для решения поставленных задач.

## 3.4 Реализация проекта базы данных

# Таблица 1.

CREATE TABLE STUDYING\_DIVISION(

sd\_short\_name VARCHAR(8) NOT NULL,

sd\_describe VARCHAR(300) NULL,

sd\_name VARCHAR(80) NOT NULL,

sd\_id INT(4) NOT NULL AUTO\_INCREMENT,

PRIMARY KEY (sd\_id)

);

# Таблица 2.

CREATE TABLE CONTACTS(

cont\_contact\_info VARCHAR(40) NULL,

cont\_division INT(4) NOT NULL REFERENCES STUDYING\_DIVISION(sd\_id)

);

# Таблица 3.

CREATE TABLE COMMUNITY(

c\_name VARCHAR(80) NOT NULL,

c\_describe VARCHAR(300) NOT NULL,

c\_type VARCHAR(16) NOT NULL,

c\_division INT(4) NOT NULL REFERENCES STUDYNG\_DIVISION(sd\_id),

c\_id INT(4) NOT NULL AUTO\_INCREMENT,

PRIMARY KEY(c\_id)

);

# Таблица 4.

CREATE TABLE COMMUNITY\_PROFILE(

cp\_member VARCHAR(80) NOT NULL,

cp\_id INT(6) NOT NULL REFERENCES COMMUNITY(c\_id)

);

# Таблица 5.

CREATE TABLE `COMMENT`(

com\_content VARCHAR(200) NOT NULL,

com\_post INT(6) NOT NULL REFERENCES POST (c\_id),

com\_id INT(6) NOT NULL AUTO\_INCREMENT,

PRIMARY KEY (com\_id)

);

# Таблица 6.

CREATE TABLE POST(

p\_name VARCHAR(80) NOT NULL,

p\_content VARCHAR(300) NOT NULL,

p\_comment INT(6) NOT NULL REFERENCES COMENT(com\_id),

p\_member INT(6) NOT NULL REFERENCES MEMBER\_OF\_COMMUNITY(mc\_id),

p\_c\_id INT(6) NOT NULL REFERENCES COMMUNITY(c\_id),

c\_id INT(6) NOT NULL AUTO\_INCREMENT,

PRIMARY KEY (c\_id)

);

# Таблица 7.

CREATE TABLE `USER`(

u\_password VARCHAR(16) NOT NULL,

u\_login VARCHAR(20) NOT NULL,

u\_e\_mail VARCHAR(40) NOT NULL UNIQUE,

u\_role VARCHAR(30) NOT NULL,

u\_fio VARCHAR(70) NOT NULL,

u\_birthday DATE NULL,

u\_group INT(4) NULL,

u\_department INT(4) NULL,

u\_work VARCHAR(20) NULL,

u\_id INT(6) NOT NULL AUTO\_INCREMENT,

PRIMARY KEY (u\_id),

FOREIGN KEY (u\_login),

FOREIGN KEY (u\_e)

);

# Таблица 8.

CREATE TABLE USER\_CONTACTS(

uc\_contacts VARCHAR(200) NOT NULL,

uc\_user INT(6) NOT NULL REFERENCES `USER`(u\_id)

);

# Таблица 9.

CREATE TABLE MEMBER\_OF\_COMMUNITY(

mc\_user INT(6) NOT NULL REFERENCES `USER`(u\_id),

mc\_community INT(6) NOT NULL REFERENCES COMMUNITY(c\_id),

mc\_id INT(6) NOT NULL AUTO\_INCREMENT,

PRIMARY KEY (mc\_id)

);

# Таблица 10.

CREATE TABLE USER\_PROFILE(

up\_info VARCHAR(300),

up\_status VARCHAR(20),

up\_foto VARCHAR(20),

up\_id INT(6) NOT NULL REFERENCES `USER`(u\_id)

);

# Таблица 11.

CREATE TABLE USER\_RATING(

upr\_rating INT(3),

upr\_user INT(6) NOT NULL REFERENCES USER\_PROFILE(up\_id)

);

# Таблица 12.

CREATE TABLE DISCIPLINE(

dis\_name VARCHAR(20) NOT NULL,

dis\_course INT(5) NOT NULL REFERENCES TRAINING\_COURSE(tc\_id),

dis\_id INT(5) NOT NULL AUTO\_INCREMENT,

PRIMARY KEY (dis\_id)

);

# Таблица 13.

CREATE TABLE TEACHER\_DISCIPLINE(

td\_discipline INT(5) NOT NULL REFERENCES DISCIPLINE(dis\_id),

td\_teacher INT(6) NOT NULL REFERENCES `USER`(u\_id)

);

# Таблица 14.

CREATE TABLE SD\_MEMBER(

sdm\_division INT(4) NOT NULL REFERENCES STUDYING\_DIVISION(sd\_id),

sdm\_user INT(4) NOT NULL REFERENCES `USER`(u\_id)

);

# ГЛАВА 4. РАЗРАБОТКА АЛГОРИТМОВ РАБОТЫ СИСТЕМЫ

## 4.1 Схемы переходов между страницами

Рассмотрим схемы переходов для всех пользователей и отдельно для администратора и управляющих.

Схемы переходов представлены в виде графа. Вершинами этого графа являются различные экранные формы. Дуги графа обозначают возможность перехода с одной страницы на другую. Это означает, что существуют кнопки или другие навигационные ссылки, которые осуществляют возможность перехода.

Схема навигации для пользователей представлена на рисунке 14.

Для пользователей в системе будет 7 экранных форм:

- Главная страница;

- Страница регистрации;

- Страница профиля;

- Страница редактирования профиля;

- Страница списка сообществ;

- Страница сообщества;

- Страница обсуждения;



Рисунок 14 – Схема навигации для пользователей

Заходя на главную страницу, посетителю предоставляется возможность зарегистрироваться в системе на «странице регистрации». После регистрации пользователь приобретает личной профиль и перенаправляется на «страницу редактирования профиля», где ему необходимо заполнить обязательные поля. После подтверждения своей роли администратором, пользователь может полнофункционально осуществлять навигацию по всей системе. Дуги графа наглядно отображают возможности перехода между страницами. Подробнее технологический процесс описан в главе 1.3 Постановка задачи.

Схема навигации для администратора и управляющих представлена на рисунке 15.

Для администратора в системе добавляется еще 4 экранных формы:

- Страница администраторской;

- Страница редактирования сообществ;

- Страница утверждения ролей пользователей;

- Страница редактирования пользовательских данных;

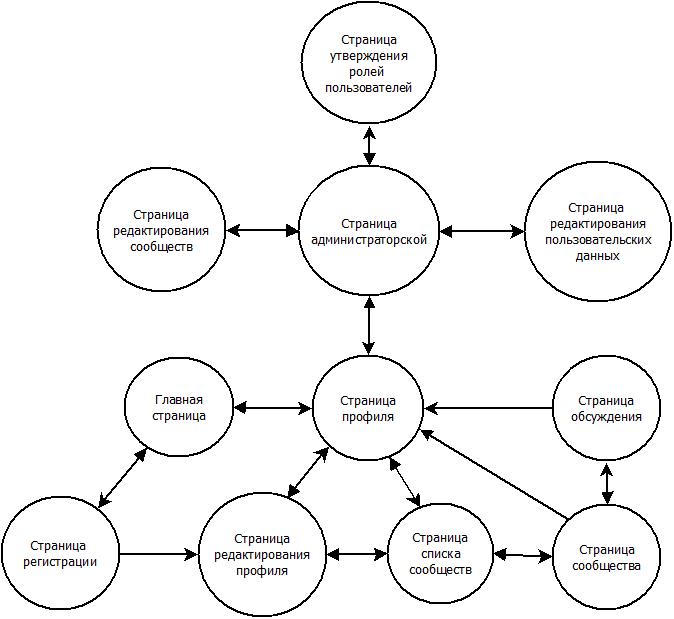


Рисунок 15 – Схема навигации для администратора и управляющих

В автоматизированной системе присутствует один главный администратор и несколько «управляющих», которые обладают правами администрирования и модерирования в пределах своих полномочий. Через «страницу администраторской» они могут осуществлять контроль сообществ, обсуждений и комментариев, также подтверждать роли недавно зарегистрированных пользователей и имеют возможность редактировать их данные.

## 4.2 Разработка алгоритмов

### 4.2.1 Алгоритм входа в систему

**цикл** работы

Вывод формы входа в систему

**цикл** входа

Заполнение формы

**до** нажатия «Вход», логин и пароль верны **кцикл**

**до** выхода из программы **кцикл**

### 4.2.2 Алгоритм регистрации первого этапа

**цикл** работы

Вывод формы регистрации

**цикл** ввода данных

Заполнение формы

**до** корректности данных **кцикл**

Отправка данных на email

**до** отправки данных или нажатия кнопки «Отмена» **кцикл**

### 4.2.3 Алгоритм регистрации второго этапа

**цикл** работы

Вывод формы подтверждения статуса и дополнительных данных

**цикл** ввода данных

Заполнение формы

**до** корректности данных **кцикл**

**если** поле подтверждения статуса заполнено, **то**

Подтверждение статуса

**кесли**

**до** отправки данных или нажатия кнопка «Отмена» **кцикл**

### 

### 4.2.4 Алгоритм РМ «Администратор»

**цикл** работы

Вывод главной страницы

**если** была нажата кнопка «О проекте», **то**

Вывод страницы «О проекте»

**кесли**

**если** была нажата кнопка «Об институте», **то**

Вывод страницы «Об институте»

**кесли**

**если** была нажата кнопка «Факультеты», **то**

Вывод страницы «Факультеты»

**кесли**

**если** была нажата кнопка «Сообщества», **то**

Вывод страницы со списком сообществ

**если** было выбрано сообщество, **то**

Вывод страницы сообщества

**кесли**

**кесли**

**если** была нажата кнопка «Контакты», **то**

Вывод страницы «Контакты»

**кесли**

**если** была нажата кнопка «Моя страница», **то**

Вывод страницы «Моя страница»

**кесли**

**если** была нажата кнопка «Редактировать», **то**

Вывод страницы редактирования профиля

**цикл** редактирования

Изменение данных

**до** корректности данных, нажатия кнопки подтверждения или отмены **кцикл**

**кесли**

**если** была нажата кнопка «Мои сообщества», **то**

Вывод списка моих сообществ

**если** нажата кнопка «Создать сообщество», **то**

Вывести форму создания сообщества

**цикл** создания сообщества

Заполнение формы

**до** корректности данных, нажатия кнопки подтверждения или отмены **кцикл**

**кесли**

**кесли**

**если** профиль пользователя, **то**

Вывод профиля пользователя

**кесли**

**если** страница сообщества, **то**

Вывод страницы сообщества

**если** выбран член сообщества, **то**

Вывод профиля пользователя

**кесли**

**если** пользователь член сообщества, **то**

**если** нажата кнопка «Покинуть» , **то**

Удалить пользователя из списка членов сообщества

**кесли**

**если** нажата кнопка «Создать пост», **то**

Вывести форму создания поста

**цикл** создания поста

Заполнение формы

**до** корректности данных, нажатия кнопки подтверждения или отмены **кцикл**

**кесли**

**если** страница поста, **то**

Отобразить страницу поста

**если** нажата кнопка «Комментировать», **то**

Вывести форму создания комментария

**цикл** создания комментария

Заполнение формы

**до** корректности данных, нажатия кнопки подтверждения или отмены **кцикл**

**кесли**

**кесли**

**кесли**

**кесли**

**до** выхода из программы **кцикл**

### 4.2.5 Алгоритм РМ «Управляющий»

**цикл** работы

Вывод главной страницы

**если** была нажата кнопка «О проекте», **то**

Вывод страницы «О проекте»

**кесли**

**если** была нажата кнопка «Об институте», **то**

Вывод страницы «Об институте»

**кесли**

**если** была нажата кнопка «Факультеты», **то**

Вывод страницы «Факультеты»

**кесли**

**если** была нажата кнопка «Сообщества», **то**

Вывод страницы со списком сообществ

**если** было выбрано сообщество, **то**

Вывод страницы сообщества

**кесли**

**кесли**

**если** была нажата кнопка «Контакты», **то**

Вывод страницы «Контакты»

**кесли**

**если** была нажата кнопка «Моя страница», **то**

Вывод страницы «Моя страница»

**кесли**

**если** была нажата кнопка «Редактировать», **то**

Вывод страницы редактирования профиля

**цикл** редактирования

Изменение данных

**до** корректности данных, нажатия кнопки подтверждения или отмены **кцикл**

**кесли**

**если** была нажата кнопка «Мои сообщества», **то**

Вывод списка моих сообществ

**если** нажата кнопка «Создать сообщество», **то**

Вывести форму создания сообщества

**цикл** создания сообщества

Заполнение формы

**до** корректности данных, нажатия кнопки подтверждения или отмены **кцикл**

**кесли**

**кесли**

**если** страница сообщества, **то**

Вывод страницы сообщества

**если** выбран член сообщества, **то**

Вывод профиля пользователя

**кесли**

**если** пользователь член сообщества, **то**

**если** нажата кнопка «Исключить» , **то**

Удалить пользователя из списка членов сообщества

**кесли**

**если** нажата кнопка «Создать пост», **то**

Вывести форму создания поста

**цикл** создания поста

Заполнение формы

**до** корректности данных, нажатия кнопки подтверждения или отмены **кцикл**

**кесли**

**если** страница поста, **то**

Отобразить страницу поста

**если** нажата кнопка «Комментировать», **то**

Вывести форму создания комментария

**цикл** создания комментария

Заполнение формы

**до** корректности данных, нажатия кнопки подтверждения или отмены **кцикл**

**кесли**

**если** нажата кнопка «Создать пост», **то**

Вывести форму создания поста

**цикл** создания поста

Заполнение формы

**до** корректности данных, нажатия кнопки подтверждения или отмены **кцикл**

**кесли**

**кесли**

**кесли**

**кесли**

**до** выхода из программы **кцикл**

### 4.2.6 Алгоритм РМ «Обычный пользователь»

**цикл** работы

Вывод главной страницы

**если** была нажата кнопка «О проекте», **то**

Вывод страницы «О проекте»

**кесли**

**если** была нажата кнопка «Об институте», **то**

Вывод страницы «Об институте»

**кесли**

**если** была нажата кнопка «Факультеты», **то**

Вывод страницы «Факультеты»

**кесли**

**если** была нажата кнопка «Сообщества», **то**

Вывод страницы со списком сообществ

**если** было выбрано сообщество, **то**

Вывод страницы сообщества

**кесли**

**кесли**

**если** была нажата кнопка «Контакты», **то**

Вывод страницы «Контакты»

**кесли**

**если** была нажата кнопка «Моя страница», **то**

Вывод страницы «Моя страница»

**кесли**

**если** была нажата кнопка «Редактировать», **то**

Вывод страницы редактирования профиля

**цикл** редактирования

Изменение данных

**до** корректности данных, нажатия кнопки подтверждения или отмены **кцикл**

**кесли**

**если** была нажата кнопка «Мои сообщества», **то**

Вывод списка моих сообществ

**кесли**

**если** страница сообщества, **то**

Вывод страницы сообщества

**если** выбран член сообщества, **то**

Вывод профиля пользователя

**кесли**

**если** пользователь член сообщества, **то**

**если** нажата кнопка «Создать пост», **то**

Вывести форму создания поста

**цикл** создания поста

Заполнение формы

**до** корректности данных, нажатия кнопки подтверждения или отмены **кцикл**

**кесли**

**если** страница поста, **то**

Отобразить страницу поста

**если** нажата кнопка «Комментировать», **то**

Вывести форму создания комментария

**цикл** создания комментария

Заполнение формы

**до** корректности данных, нажатия кнопки подтверждения или отмены **кцикл**

**кесли**

**кесли**

**кесли**

**кесли**

**до** выхода из программы **кцикл**

### 4.2.7 Алгоритм РМ «Преподаватель»

**цикл** работы

Вывод главной страницы

**если** была нажата кнопка «О проекте», **то**

Вывод страницы «О проекте»

**кесли**

**если** была нажата кнопка «Об институте», **то**

Вывод страницы «Об институте»

**кесли**

**если** была нажата кнопка «Факультеты», **то**

Вывод страницы «Факультеты»

**кесли**

**если** была нажата кнопка «Сообщества», **то**

Вывод страницы со списком сообществ

**если** было выбрано сообщество, **то**

Вывод страницы сообщества

**кесли**

**кесли**

**если** была нажата кнопка «Контакты», **то**

Вывод страницы «Контакты»

**кесли**

**если** была нажата кнопка «Моя страница», **то**

Вывод страницы «Моя страница»

**кесли**

**если** была нажата кнопка «Редактировать», **то**

Вывод страницы редактирования профиля

**цикл** редактирования

Изменение данных

**до** корректности данных, нажатия кнопки подтверждения или отмены **кцикл**

**кесли**

**если** была нажата кнопка «Мои сообщества», **то**

Вывод списка моих сообществ

**если** нажата кнопка «Создать сообщество», **то**

Вывести форму создания сообщества

**цикл** создания сообщества

Заполнение формы

**до** корректности данных, нажатия кнопки подтверждения или отмены **кцикл**

**кесли**

**кесли**

**если** страница сообщества, **то**

Вывод страницы сообщества

**если** выбран член сообщества, **то**

Вывод профиля пользователя

**кесли**

**если** пользователь член сообщества, **то**

**если** нажата кнопка «Создать пост», **то**

Вывести форму создания поста

**цикл** создания поста

Заполнение формы

**до** корректности данных, нажатия кнопки подтверждения или отмены **кцикл**

**кесли**

**если** страница поста, **то**

Отобразить страницу поста

**если** нажата кнопка «Комментировать», **то**

Вывести форму создания комментария

**цикл** создания комментария

Заполнение формы

**до** корректности данных, нажатия кнопки подтверждения или отмены **кцикл**

**кесли**

**если** нажата кнопка «Создать пост», **то**

Вывести форму создания поста

**цикл** создания поста

Заполнение формы

**до** корректности данных, нажатия кнопки подтверждения или отмены **кцикл**

**кесли**

**кесли**

**кесли**

**кесли**

**до** выхода из программы **кцикл**

### 4.2.8 Алгоритм РМ «Студент»

**цикл** работы

Вывод главной страницы

**если** была нажата кнопка «О проекте», **то**

Вывод страницы «О проекте»

**кесли**

**если** была нажата кнопка «Об институте», **то**

Вывод страницы «Об институте»

**кесли**

**если** была нажата кнопка «Факультеты», **то**

Вывод страницы «Факультеты»

**кесли**

**если** была нажата кнопка «Сообщества», **то**

Вывод страницы со списком сообществ

**если** было выбрано сообщество, **то**

Вывод страницы сообщества

**кесли**

**кесли**

**если** была нажата кнопка «Контакты», **то**

Вывод страницы «Контакты»

**кесли**

**если** была нажата кнопка «Моя страница», **то**

Вывод страницы «Моя страница»

**кесли**

**если** была нажата кнопка «Редактировать», **то**

Вывод страницы редактирования профиля

**цикл** редактирования

Изменение данных

**до** корректности данных, нажатия кнопки подтверждения или отмены **кцикл**

**кесли**

**если** была нажата кнопка «Мои сообщества», **то**

Вывод списка моих сообществ

**если** нажата кнопка «Создать сообщество», **то**

Вывести форму создания сообщества

**цикл** создания сообщества

Заполнение формы

**до** корректности данных, нажатия кнопки подтверждения или отмены **кцикл**

**кесли**

**кесли**

**если** страница сообщества, **то**

Вывод страницы сообщества

**если** выбран член сообщества, **то**

Вывод профиля пользователя

**кесли**

**если** пользователь член сообщества, **то**

**если** нажата кнопка «Создать пост», **то**

Вывести форму создания поста

**цикл** создания поста

Заполнение формы

**до** корректности данных, нажатия кнопки подтверждения или отмены **кцикл**

**кесли**

**если** страница поста, **то**

Отобразить страницу поста

**если** нажата кнопка «Комментировать», **то**

Вывести форму создания комментария

**цикл** создания комментария

Заполнение формы

**до** корректности данных, нажатия кнопки подтверждения или отмены **кцикл**

**кесли**

**кесли**

**кесли**

**кесли**

**до** выхода из программы **кцикл**

## 4.3 Выбор средства реализации

В области разработки платформ для создания веб-приложений на языке Java были достигнуты значительные успехи, однако создание новых приложений на их основе все еще является весьма трудоемким делом. В мире разработки на языке Java существует масса платформ для веб-приложений: Struts, WebWork, JSF, Grails, Spring MVC, Seam, Wicket, tapestry, Stripes, GWT.

Остановим свое внимание на Grails и попутно рассмотрим ее преимущества над другими платформами.

Grails – платформа «нового поколения» разработки веб-приложений на языке Java, которая позаимствовала оптимальные инструментальные средства веб-разработки, приемы и методики из существующих платформ Java и объединила их с мощным и передовым динамическим языком программирования Groovy. Благодаря использованию языка Groovy она предоставляет в распоряжение разработчика мощный Java-подобный язык, а также полный доступ ко всем Java-библиотекам. В результате получилась платформа, предоставляющая стабильные технологии, защищающая от утомительных настроек, сложностей проектирования и необходимости написания шаблонного программного кода, который делает разработку веб-приложений на языке Java несколько утомительной.

Groovy — объектно-ориентированный язык программирования разработанный для платформы Java как альтернатива языку Java с возможностями Python, Ruby и Smalltalk.

Groovy использует Java-подобный синтаксис с динамической компиляцией в JVM байт-код и напрямую работает с другим Java кодом и библиотеками. Язык может использоваться в любом Java проекте или как скриптовый язык.  
  
Возможности Groovy (отличающие его от Java):

* Статическая и динамическая типизация
* Встроенный синтаксис для списков, ассоциативных массивов, массивов и регулярных выражений
* Замыкания
* Перегрузка операций

Существует мнение, что платформа Grails является реализацией Ruby on Rails на языке Java, однако это ошибочное мнение легко опровергается следующими сведениями о Grails:

* Платформа Grails привнесла удивительные инновации в сектор разработки корпоративных приложений.
* На Grails оказал влияние широкий диапазон платформ, таких как Ruby,Python, PHP и Java.
* Многие особенности, имеющиеся в арсенале Grails, отсутствуют в Rails – особенности использования корпоративного окружения, созданные в течение длинной истории развития виртуальных машин Java (JVM).

Семь основных доводов, обеспечивающих платформе Grails доминирующее положение на рынке веб-платформ на языке Java (рисунок 16).

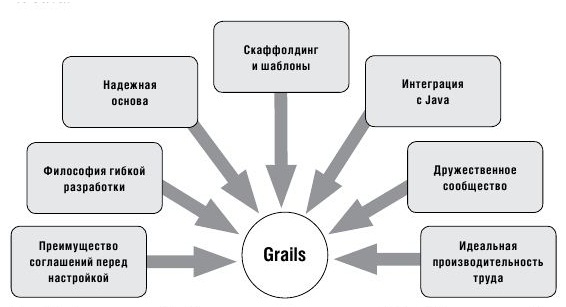


Рисунок 16 – Доводы преимуществ платформы Grails

**1. Преимущество соглашений перед настройками**

В платформе Grails мало файлов с настройками. Большинство решений принимается платформой Grails на основе достаточно разумных значений по умолчанию в программном коде. Однако, это не означает использование соглашений вместо настроек. Если потребуется отладить поведение по умолчанию, для этого имеются все средства.

**2. Философия гибкой разработки**

Платформа Grails обеспечивает серьезные возможности, дающие ей право называться гибкой веб-платформой. Благодаря использованию динамического языка программирования (Groovy) платформа Grails позволяет легко и просто создавать вещи, которые при работе на языке Java были очень трудоемкими. Например, обработка форм, реализация библиотек тегов и написание тестов. Появляется возможность изменять поведение приложения прямо во время его выполнения без перезагрузки сервера. Платформа Grails выводит гибкость разработки веб-приложений на языке Java на совершенно новый уровень.

**3. Надежная основа**

Grails наполнена новыми, ультрасовременными идеями, но основу ее составляют испытанные технологии: Spring и Hibernate. Данные технологии используются во многих проектах на Java, они надежны и проверены временем. Модуль планировщика выполненения заданий построен на базе механизма Quartz, модуль поиска – на базе библиотек Lucence и Compass. А механизм отображения – на базе SiteMesh.

Имеет место что-то вроде максимального улучшения существующих приемов разработки без отказа от надежных и проверенных технологий.

**4. Скаффолдинг и шаблоны**

Если попробовать вручную развернуть приложение на платформе Spring MVC, потребуется создать каталог для файлов JAR, множество файлов с определениями специальных классов бинов (bean), набор файлов web.xml с настройками, связку аннотированных объектов POJO, несколько файлов с настройками для библиотеки Hibernate, сценарий создания базы данных, после чего система сборки сможет превратить все это в работающее приложение. На платформе Grails достаточно всего одной команды чтобы собрать работающее приложение. Командой выполняются те же самые операции, но они опираются на соглашения и разумные значения по умолчанию, а не на настройки определяемые вручную.

Такой шаблонный подход обеспечивает фантастическую производительность труда, благодаря которой можно сконцентрироваться на функциональности и не отвлекаться на написание шаблонного программного кода.

Платформа Grails обладает удивительной особенностью, получившей название «скаффолдинг» (scaffolding). На основе классов, описывающих модель (структуру) базы данных, платформа Grails может динамически генерировать набор представлений и контроллеров, позволяющих выполнять простейшие CRUD-операции – creating (создание), reading (чтение), updating (изменение) и deleting (удаление) – без единой строчки программного кода.

**5. Интеграция с Java**

Одна из уникальных особенностей сообщества Groovy и Grails состоит в том, что в отличие от некоторых других языков, поддерживаемых JVM, сообщество Grails признает, что есть задачи и решения которые лучше реализуются на языке со статической типизацией, поэтому нет проблем при создании классов обработки форм на языке Groovy и реализации высокопроизводительных вычислительных алгоритмов на языке Java.

**6. Дружелюбное сообщество**

Существует большое количество порталов и веб-сайтов сообществ разработчиков Grails, которые содержат информацию и документацию о платформе, и оказывают пользовательскую поддержку. В таких сообществах регулярно публикуется растущий список сторонних расширений для Grails .Существуют буквально сотни расширений, доступных для использования.

**7. Идеальная производительность труда**

Платформа Grails открывает возможность быстро и производительно разрабатывать надежные веб-приложения, увеличивает производительность разработчика благодаря совместному использованию динамического языка программирования, философии преимущества соглашений перед настройками, мощных практичных инструментов и гибких методик, заимствованных из лучших парадигм веб-разработки.

Подводя итог, можно сделать вывод, что Grails является отличным кандидатом для реализации веб-проекта [13].

## 4.4 Описание структур кода. Особенности программирования на Grails

При написании программ на платформе Grails происходит взаимодействие таких объектов как:

**1) Контроллеры (Controllers)**

Контроллеры - основа любого приложения на платформе Grails. Они принимают ввод от веб-браузера пользователя, взаимодействуют с бизнес-логикой и моделью данных и обеспечивают отправку пользователю корректных страниц. Без контроллеров веб-приложение было бы лишь связкой статических страниц.

**2) Представления (views)**

Реализация оформления страниц на платформе Grails происходит при помощи представлений. Представление - это отдельные файлы, содержащие логику управления отображениями.

**3) Макеты**

Макеты обеспечивают возможность определить шаблоны размещения определенных частей приложения.

**4) Модели данных (models)**

Для описания объектов данных, которые могут сохраняться в базе данных, в платформе Grails используется классы предметной области. Классы модели хранятся в директории проекта Grails [13].

**Приложения Grails работают согласно архитектуре «Модель-представление-контроллер» (MVC)**

«Архитектурная схема модель-представление-контроллер (MVC) осуществляет разделение между “логикой предметной области” (также называемой “бизнес-логикой”) от логики ввода и логики представления, связанной с графическим интерфейсом пользователя (GUI). В случае веб-приложений, “логика предметной области” обычно состоит из модели данных для таких вещей как пользователи, статьи, продукты, а GUI это просто веб-страница в браузере.

Взаимодействуя с приложением Grails, браузер отправляет запрос, который принимается веб-сервером и передается контроллеру Grails, отвечающему за то, что делать дальше (рисунок 17). В некоторых случаях контроллер сразу визуализирует представление, которое является шаблоном, конвертирующим полученное в HTML и отсылающим назад к браузеру. Более обычно для динамичных сайтов, когда контроллер взаимодействует с моделью, которая является объектом Groovy, который представляет собой элемент сайта (такой как пользователь) и отвечает за коммуникацию с базой данных. После вызова модели, контроллер затем визуализирует представление и возвращает полную веб-страницу браузеру как HTML» [15].

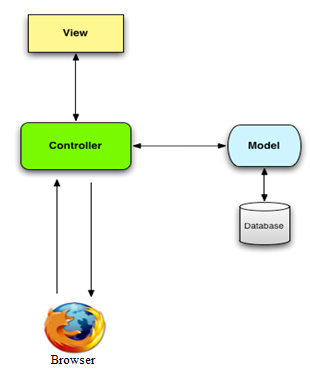


Рисунок 17 – Схематичное изображение модель-представление-контроллер (MVC) архитектуры

Общий программный код представляет собой совокупность программного кода контроллеров, представлений, моделей и макетов. Реализация всех составляющих структур кода приведена в Приложении.

# ЗАКЛЮЧЕНИЕ

Итогом проделанной работы является созданная автоматизированная система коммуникации субъектов учебного процесса. В процессе разработки были выполнены следующие задачи:

* Описана предметная область и сформулирована постановка задачи дипломной работы
* Разработана функциональная модель системы в виде отображения внутренних процессов при помощи построения операционных диаграмм стандарта IDEF0
* Спроектирована и реализована структура базы данных
* Проведен анализ различных СУБД
* Построены общие алгоритмы и реализован заявленный набор функций при помощи платформы grails на java подобном языке.
* Рассмотрены возможности и особенности программирования на платформе Grails.

Для решения поставленной задачи была изучена набирающая популярность перспективная платформа разработки веб-приложений Grails. Также такие технологии как html, css, javascript, ajax; прикладная программа визуального моделирования BPwin, язык структурированных запросов sql и работа с системой управления базами данных MySQL.

Разработанная автоматизированная система позволит обеспечивать общение между студентами и преподавателями за рамками учебных занятий, что будет способствовать повышению эффективности учебного процесса. Система ориентирована по большей части на эксплуатацию в таких учреждениях как кафедры вузов, но также может быть адаптирована и для других образовательных учреждений.

# СПИСОК ЛИТЕРАТУРЫ

При написании данной дипломной работы использовались следующие книги и интернет-ресурсы:

1. Уроки, справочники [Электронный ресурс]: Методологии моделирования предметной области. Режим доступа к ресурсу: <http://do.gendocs.ru/docs/index-101511.html>
2. Автоматизация управления компаниями [Электронный ресурс]: Описание стандарта IDEF0. Режим доступа к ресурсу: <http://www.insapov.ru/idef0-standard-description.html>
3. Лунаев Е.А. Информационные технологии управления персоналом [Электронный ресурс]: Моделирование бизнес-процессов. Режим доступа к ресурсу: [http://www.e-college.ru/xbooks/xbook106/book/index/index.html?go=part-007\*page.htm](http://www.e-college.ru/xbooks/xbook106/book/index/index.html?go=part-007*page.htm)
4. Карпова И.П."Проектирование реляционных баз данных".– М.: Московский Государственный институт электроники и математики, 2010. – 32 с.
5. Карпова И.П. "Основы баз данных. Учебное пособие". – М.: Московский Государственный институт электроники и математики, 2007. – 75 с.
6. Microsoft® SQL Server™ 2005.Реализация и обслуживание. Учебный курс Microsoft (Экзамен 70-431). — М.: «Питер», 2007. — С. 767
7. ABACUS Корпоративные информационные системы [Электронный ресурс]: Характеристика СУБД Oracle. Режим доступа к ресурсу: <http://www.omega.ru/oracleinfo.html>
8. Балдин Е.М. История о PostgreSQL [Электронный ресурс]: Возможности PostgreSQL. Режим доступа к ресурсу: <http://www.inp.nsk.su/~baldin/PostgreSQL/postgresql-possibility.pdf>
9. Кузнецов Максим, Симдянов Игорь. Самоучитель MySQL 5. — Спб.: «БХВ-Петербург», 2006. — С. 560.
10. Пол Макфедрис. «Создание веб-страниц». – М.: АСТ., 2005 - 387 с.
11. Мержевич В.В. «Справочник по HTML. Версия 7.0». – 2011
12. Мержевич В.В. «Справочник CSS v 6.0», – 2011
13. Глен Смит, Питер Ледбрук «Grails. Гибкость Groovy и надежность Java». – Санкт-Петербург, Москва., 2010
14. Нечаев А.М. Создание программ для компьютера. Учебное пособие - Московский государственный институт электроники и математики. М., 2003
15. Майкл Хартл. [Электронный ресурс]: Изучение Rails на Примерах, 2010. Режим доступа к ресурсу: <http://railstutorial.ru/chapters/beginning?version=3.0#sec:mvc>

# ПРИЛОЖЕНИЕ

**Объекты предметной области**

package soc

class Badge {

String name

byte[] picture

long weight = 1

Date dateCreated

Date lastUpdated

static constraints = {

name(blank: false)

picture(maxSize: 1024)

}

}

package soc

class Comment {

static belongsTo = Post

String content

User master

Date dateCreated

Date lastUpdated

static constraints = {

content(nullable: false, minSize: 6, maxSize: 3000)

master(nullable: true)

}

static mapping = {

autoTimestamp true

master(lazy: false)

}

}

package soc

class Community {

String type

String privateType

String shortName // Абривиатура

String c\_describe // Описание

String name // Название

Date dateCreated

Date lastUpdated

TrainingDepartment studyingDivision

static constraints = {

name(blank: false, maxSize: 100, unique: true)

shortName(blank: false, maxSize: 30)

c\_describe(blank: false, maxSize: 300)

master(nullable: true)

type(inList: ["научное", "университет", "факультет", "кафедра", "группа", "по интересам"])

privateType(inList: ["закрытое", "открытое"])

studyingDivision(nullable: true)

}

static mapping = {

posts(lazy: false)

members(lazy: false)

master(lazy: false)

studyingDivision(lazy: false)

}

static belongsTo = [master: User]

static hasMany = [posts: Post, members: Member]

boolean userIsMember(User user) {

if (!user) return false

return members.contains(user)

}

boolean getIsPrivate() {

return privateType == "закрытое"

}

}

package soc

class Contact {

String info

Date dateCreated

Date lastUpdated

static constraints = {

info(blank: false, maxSize: 300)

}

static belongsTo = TrainingDepartment // Объект Contact принадлежит TrainingDepartment (учебное отделение)

}

package soc

class Department {

String shortName // Абривиатура

String c\_describe // Описание

String name // Название

Date dateCreated

Date lastUpdated

static constraints = {

shortName(nullable: false, maxSize: 30)

c\_describe(nullable: false, maxSize: 300)

name(nullable: false, maxSize: 100)

}

}

package soc

class Discipline {

String name

Date dateCreated

Date lastUpdated

TrainingCourse course

static constraints = {

name(blank: false, minSize: 3)

}

}

package soc

class DisciplineTeacher {

static belongsTo = [teacher: User, discipline: Discipline]

}

package soc

class Group {

String shortName // Абривиатура

String c\_describe // Описание

String name // Название

Date dateCreated

Date lastUpdated

static hasMany = [students: User]

static constraints = {

shortName(nullable: false, maxSize: 30, unique: true)

c\_describe(nullable: false, maxSize: 300)

name(nullable: false, maxSize: 100, unique: true)

students(nullable: true)

}

static mapping = {

table name: '`group`'

}

}

package soc

class Member {

Date dateCreated

Date lastUpdated

static belongsTo = [user: User, community: Community]

}

package soc

class Post {

String name

Member author

Date dateCreated

Date lastUpdated

static constraints = {

name(nullable: false, size: 3..30)

author(nullable: true)

comments(nullable: true)

}

static mapping = {

autoTimestamp true

comments(lazy: false)

}

static belongsTo = Community

static hasMany = [comments: Comment]

}

package soc

/\*\*

\* Request Map domain class.

\*/

class Requestmap {

String url

String configAttribute

static constraints = {

url(blank: false, unique: true)

configAttribute(blank: false)

}

}

package soc

/\*\*

\* Authority domain class.

\*/

class Role {

static hasMany = [people: User]

/\*\* description \*/

String description

/\*\* ROLE String \*/

String authority

String name

static constraints = {

authority(nullable: false, blank: false, unique: true)

description(nullable: true, blank: true)

name(nullable: true, blank: true)

people(nullable: true)

}

@Override

String toString() {

return name ?: description

}

}

package soc

class StudentTask {

Date dateCreated

Date lastUpdated

static belongsTo = [student: User, task: Task]

}

package soc

class Task {

String type

Date dateCreated

Date lastUpdated

String content

static constraints = {

content(blank: false, minSize: 6)

type(inList: ["домашняя работа", "лабораторная работа", "курсовая работа", "самостоятельная работа", "другое"])

}

static hasMany = [matireals: TrainingMaterial]

}

package soc

class TrainingCourse {

int numLectures

int numRecords

Date dateCreated

Date lastUpdated

static constraints = {

numLectures(min: 1)

numRecords(min: 1)

}

static belongsTo = [department: TrainingDepartment]

}

package soc

// Модель учебного отделения

class TrainingDepartment {

String shortName // Абривиатура

String c\_describe // Описание

String name // Название

Date dateCreated

Date lastUpdated

Contact contacts

static constraints = {

name(nullable: false, maxSize: 100)

shortName(nullable: false, maxSize: 30)

c\_describe(nullable: false, maxSize: 300)

contacts(nullable: false)

communities(nullable: true)

members(nullable: true)

}

String toString() {

return name

}

static hasMany = [communities: Community, members: TrainingMember]

}

package soc

class TrainingFile {

byte [] content

Date dateCreated

Date lastUpdated

static constraints = {

content(maxSize: 10240)

}

}

package soc

class TrainingMaterial {

String name

Date dateCreated

Date lastUpdated

static constraints = {

name(blank: false, minSize: 6)

}

static belongsTo = [discipline: Discipline]

}

package soc

class TrainingMember {

static belongsTo = [division: TrainingDepartment, user: User]

Date dateCreated

Date lastUpdated

}

package soc

class User {

static transients = ['pass']

static hasOne = [group: Group]

static hasMany = [authorities: Role]

static belongsTo = Role

String login

String password

UserContact contacts

UserProfile profile

Date dateCreated

Date lastUpdated

// For Spring Security plugin's user registration.

String email

String userRealName = "man"

boolean emailShow = false

boolean enabled = true

/\*\* description \*/

String description = ''

/\*\* plain password to newComm a MD5 password \*/

String pass = '[secret]'

static constraints = {

login(blank: false, unique: true, size: 3..10)

email(blank: false, unique: true, email: true)

password(blank: false, minSize: 6)

enabled()

dateCreated()

lastUpdated()

role()

profile(nullable: true)

contacts(nullable: true)

group(nullable: true)

description(blank: true)

userRealName(blank: true)

}

static mapping = {

profile(lazy: false)

role(lazy: false)

contacts(lazy: false)

authorities(lazy: false)

}

Role getRole() {

if (!authorities) return null

for (role in authorities) {

return role

}

return null

}

}

package soc

class UserBadge {

static constraints = {

}

Date dateCreated

Date lastUpdated

static belongsTo = [badge: Badge, user: User]

}

package soc

class UserContact {

static belongsTo = User

String content

Date dateCreated

Date lastUpdated

static constraints = {

content(minSize: 6)

}

}

package soc

class UserProfile {

static belongsTo = User

String aboutMe

String state

String surname // фамилия

String name // имя

String middle\_name // отчество

String work

Date birthday

Group group

Department department

UserRating rating

Boolean editable = true

Date dateCreated

Date lastUpdated

byte[] photo

static constraints = {

aboutMe(blank: true)

editable()

state(blank: true)

work(blank: true)

birthday(nullable: true)

group(nullable: true)

department(nullable: true)

badges(nullable: true)

photo(nullable: true, maxSize: 1024)

surname(blank: false, minSize: 3)

name(blank: false, minSize: 3)

middle\_name(blank: false, minSize: 3)

rating(nullable: false)

}

static mapping = {

group(lazy: false)

department(lazy: false)

}

static hasMany = [badges: UserBadge]

}

package soc

class UserRating {

static belongsTo = UserProfile

Date dateCreated

Date lastUpdated

int value = 0

static constraints = {

value(min: 0)

}

}

**Контроллеры**

package soc

class BadgeController {

def index = { redirect(action: "allComm", params: params) }

// the delete, saveComm and update actions only accept POST requests

static allowedMethods = [save: "POST", update: "POST", delete: "POST"]

def list = {

params.max = Math.min(params.max ? params.max.toInteger() : 10, 100)

[badgeInstanceList: Badge.list(params), badgeInstanceTotal: Badge.count()]

}

def create = {

def badgeInstance = new Badge()

badgeInstance.properties = params

return [badgeInstance: badgeInstance]

}

def save = {

def badgeInstance = new Badge(params)

if (!badgeInstance.hasErrors() && badgeInstance.save()) {

flash.message = "badge.created"

flash.args = [badgeInstance.id]

flash.defaultMessage = "Badge ${badgeInstance.id} created"

redirect(action: "showComm", id: badgeInstance.id)

}

else {

render(view: "newComm", model: [badgeInstance: badgeInstance])

}

}

def show = {

def badgeInstance = Badge.get(params.id)

if (!badgeInstance) {

flash.message = "badge.not.found"

flash.args = [params.id]

flash.defaultMessage = "Badge not found with id ${params.id}"

redirect(action: "list")

}

else {

return [badgeInstance: badgeInstance]

}

}

def edit = {

def badgeInstance = Badge.get(params.id)

if (!badgeInstance) {

flash.message = "badge.not.found"

flash.args = [params.id]

flash.defaultMessage = "Badge not found with id ${params.id}"

redirect(action: "list")

}

else {

return [badgeInstance: badgeInstance]

}

}

def update = {

def badgeInstance = Badge.get(params.id)

if (badgeInstance) {

if (params.version) {

def version = params.version.toLong()

if (badgeInstance.version > version) {

badgeInstance.errors.rejectValue("version", "badge.optimistic.locking.failure", "Another user has updated this Badge while you were editing")

render(view: "edit", model: [badgeInstance: badgeInstance])

return

}

}

badgeInstance.properties = params

if (!badgeInstance.hasErrors() && badgeInstance.save()) {

flash.message = "badge.updated"

flash.args = [params.id]

flash.defaultMessage = "Badge ${params.id} updated"

redirect(action: "showComm", id: badgeInstance.id)

}

else {

render(view: "edit", model: [badgeInstance: badgeInstance])

}

}

else {

flash.message = "badge.not.found"

flash.args = [params.id]

flash.defaultMessage = "Badge not found with id ${params.id}"

redirect(action: "edit", id: params.id)

}

}

def delete = {

def badgeInstance = Badge.get(params.id)

if (badgeInstance) {

try {

badgeInstance.delete()

flash.message = "badge.deleted"

flash.args = [params.id]

flash.defaultMessage = "Badge ${params.id} deleted"

redirect(action: "list")

}

catch (org.springframework.dao.DataIntegrityViolationException e) {

flash.message = "badge.not.deleted"

flash.args = [params.id]

flash.defaultMessage = "Badge ${params.id} could not be deleted"

redirect(action: "showComm", id: params.id)

}

}

else {

flash.message = "badge.not.found"

flash.args = [params.id]

flash.defaultMessage = "Badge not found with id ${params.id}"

redirect(action: "list")

}

}

}

package soc

class BasicController {

def authenticateService

def userService

def index() {

User user = authenticateService.userDomain() as User

[user: user]

}

def aboutInstitute() {

User user = authenticateService.userDomain() as User

[user: user]

}

def aboutCathedra() {

User user = authenticateService.userDomain() as User

[user: user]

}

def contacts() {

User user = authenticateService.userDomain() as User

[user: user]

}

def doLogin() {

def login = params.j\_username

def unhashedPassword = params.j\_password

def password = authenticateService.encodePassword(params.j\_password)

User user = login ? User.findByLoginAndPassword(login, password) : null

session.user = null

if (user) {

user = userService.deleteSecretFields(user)

session.user = user

}

redirect(action: "index")

}

def logout = {

User user = authenticateService.userDomain() as User

session.user = null

redirect(uri: "/")

}

def toMiem() {

redirect(url: "http://miem.edu.ru/%D0%9E%D0%B1-%D1%83%D0%BD%D0%B8%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D1%82%D0%B5%D1%82%D0%B5.html")

}

def toHome() {

redirect(url: "http://fe.miem.edu.ru/tree/?id=55")

}

}

package soc

import java.awt.Color

import java.awt.Font

import java.awt.Graphics2D

import java.awt.RenderingHints

import java.awt.geom.Rectangle2D

import java.awt.image.BufferedImage

import javax.imageio.ImageIO

class CaptchaController {

private static final String SOURCECHARS = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'

def index = {

response.setContentType('image/png')

response.setHeader('Cache-control', 'no-cache')

// Generate and remember the Source Character string (6 characters)

int l = SOURCECHARS.length()

StringBuilder b = new StringBuilder()

6.times {

int r = (int) (Math.random() \* l)

b.append(SOURCECHARS.charAt(r))

}

final int height = 200

final int width = 200

final int space = 8

System.setProperty('java.awt.headless', 'true')

BufferedImage bufferedImage = new BufferedImage(width, height, BufferedImage.TYPE\_INT\_RGB)

Graphics2D g2d = bufferedImage.createGraphics()

Font font = new Font('Serif', Font.BOLD, 18)

g2d.setFont(font)

g2d.setRenderingHint(RenderingHints.KEY\_ANTIALIASING, RenderingHints.VALUE\_ANTIALIAS\_ON)

Rectangle2D fontRect = font.getStringBounds(b.toString(), g2d.getFontRenderContext())

// Now, newComm a graphic 'space' pixels wider and taller than the the font

bufferedImage = new BufferedImage((int) fontRect.getWidth() + space,

(int) fontRect.getHeight() + space,

BufferedImage.TYPE\_INT\_RGB)

g2d = bufferedImage.createGraphics()

g2d.setRenderingHint(RenderingHints.KEY\_ANTIALIASING, RenderingHints.VALUE\_ANTIALIAS\_ON)

g2d.setFont(font)

// Draw the background

g2d.setColor(Color.WHITE)

g2d.fillRect(0, 0, width, height)

// Draw the lines

g2d.setColor(Color.GRAY)

int x1

int y1

int x2

int y2

final int step = 10

x1 = 0

y1 = step

x2 = step

y2 = 0

while (x1 < width || x2 < width || y1 < height || y2 < height) {

g2d.drawLine(x1, y1, x2, y2)

if (y1 < height) {

x1 = 0

y1 += step

}

else if (x1 < width) {

y1 = height

x1 += step

}

else {

x1 = width

y1 = height

}

if (x2 < width) {

y2 = 0

x2 += step

}

else if (y2 < height) {

x2 = width

y2 += step

}

else {

y2 = height

x2 = width

}

}

// Draw the String

g2d.setColor(Color.BLACK)

g2d.drawString(b.toString(), (int) (space / 2), (int) (space / 4) + (int) fontRect.getHeight())

OutputStream out = response.getOutputStream()

ImageIO.write(bufferedImage, 'PNG', out)

out.close()

session.setAttribute('captcha', b.toString())

}

}

package soc

class CommentController {

def scaffold = true

def index = {}

}

package soc

class CommunityController {

def scaffold = true

def rolesService

def authenticateService

def communityService

def paginateService

static defaultAction = "allComm"

def newComm = {

def communityInstance = new Community()

communityInstance.properties = params

return [communityInstance: communityInstance]

}

def saveComm = {

User user = authenticateService.userDomain() as User

if (user) params.put("master", user)

def communityInstance = new Community(params)

if (!communityInstance.hasErrors() && communityInstance.save()) {

flash.message = "community.created"

flash.args = [communityInstance.id]

flash.defaultMessage = "Community ${communityInstance.id} created"

def community = Community.findById(communityInstance.id)

if (community && user) {

Member member = new Member(community: community, user: user)

if (member) community.addToMembers(member)

}

redirect(action: "showComm", id: communityInstance.id)

}

else {

render(view: "newComm", model: [communityInstance: communityInstance])

}

}

def allComm = {

User user = rolesService.authenticateService.userDomain() as User

if (user)

params.max = Math.min(params.max ? params.max.toInteger() : 7, 100)

[communityInstanceList: Community.list(params), communityInstanceTotal: Community.count()]

}

def filter = {

params.max = Math.min(params.max ? params.int('max') : 7, 100)

render(template: "listTemplate", model: [communityInstanceList: Community.list(params), communityInstanceTotal: Community.count()])

}

def ajaxPost = {

Community communityInstance = Community.get(session.comm\_id)

def point = 0

if (communityInstance) {

params.max = Math.min(params.max ? params.int('max') : 7, 100)

List posts = communityInstance.posts as List

int offset = (params.offset ?: 0) as int

int max = params.int('max')

int totalPost = posts.size()

List<Post> subList = paginateService.getPaginateList(posts, offset, max) as List<Post>

render(template: "listPosts", model: [posts: subList, totalPost: totalPost])

} else {

redirect(action: "allComm")

}

}

def getCommunties = {

[communities: Community.list(params), total: Community.count()]

}

def showComm = {

def communityInstance = Community.findByShortName(params.id)

if (communityInstance) {

communityService.community = communityInstance.id

flash.comm\_id = communityInstance.id

session.comm\_id = communityInstance.id

}

if (!communityInstance) {

flash.message = "community.not.found"

flash.args = [params.id]

flash.defaultMessage = "Community not found with id ${params.id}"

redirect(action: "allComm")

} else {

User user = authenticateService.userDomain() as User

Member member = Member.findByUserAndCommunity(user, communityInstance)

boolean isAdmin = authenticateService.ifAnyGranted(rolesService.admin)

Role rector = Role.findByAuthority(rolesService.rector)

Role teacher = Role.findByAuthority(rolesService.teacher)

Role student = Role.findByAuthority(rolesService.student)

Role userRole = Role.findByAuthority(rolesService.user)

ArrayList<Member> members = communityInstance.members

ArrayList<User> rectors = userList(members, rector)

ArrayList<User> teachers = userList(members, teacher)

ArrayList<User> students = userList(members, student)

ArrayList<User> users = userList(members, userRole)

List posts = communityInstance.posts as List

params.max = Math.min(params.max ? params.int('max') : 7, 100)

def offset = params.offset ?: 0

def totalPost = posts.size()

def max = Math.min(((params.max ?: 0) + offset - 1), totalPost - 1)

List subList = totalPost != 0 ? posts[offset..max] : null

flash.rectors = rectors

flash.teachers = teachers

flash.students = students

flash.users = users

def point = 0

if ((communityInstance.isPrivate) && ((!isAdmin) || (!member))) {

return [community: communityInstance, posts: null, totalPost: 0, rectors: null, teachers: null, students: null,

users: null, rolesService: rolesService]

}

return [community: communityInstance, posts: subList, totalPost: totalPost, rectors: rectors, teachers: teachers, students: students,

users: users, rolesService: rolesService]

}

}

private ArrayList<User> userList(ArrayList<Member> members, Role role) {

ArrayList<User> result = new ArrayList<User>()

for (Member member : members) {

User user = member.user

if (user.role == role)

result.add(user)

}

return result;

}

}

package soc

class ContactController {

def scaffold = true

def index = {}

}

package soc

class DepartmentController {

def scaffold = true

def index = {}

}

package soc

class DisciplineController {

def scaffold = true

def index = {}

}

package soc

class DisciplineTeacherController {

def scaffold = true

def index = {}

}

package soc

class FileController {

def index() { }

}

package soc

import org.springframework.security.context.SecurityContextHolder as SCH

import org.codehaus.groovy.grails.plugins.springsecurity.RedirectUtils

import org.springframework.security.AuthenticationTrustResolverImpl

import org.springframework.security.DisabledException

import org.springframework.security.ui.AbstractProcessingFilter

import org.springframework.security.ui.webapp.AuthenticationProcessingFilter

/\*\*

\* Login Controller (Example).

\*/

class LoginController {

/\*\*

\* Dependency injection for the authentication service.

\*/

def authenticateService

/\*\*

\* Dependency injection for OpenIDConsumer.

\*/

def openIDConsumer

/\*\*

\* Dependency injection for OpenIDAuthenticationProcessingFilter.

\*/

def openIDAuthenticationProcessingFilter

private final authenticationTrustResolver = new AuthenticationTrustResolverImpl()

def index = {

if (isLoggedIn()) {

redirect uri: '/'

}

else {

redirect action: auth, params: params

}

}

/\*\*

\* Show the login page.

\*/

def auth = {

nocache response

if (isLoggedIn()) {

redirect uri: '/'

return

}

String view

String postUrl

def config = authenticateService.securityConfig.security

if (config.useOpenId) {

view = 'openIdAuth'

postUrl = "${request.contextPath}/login/openIdAuthenticate"

}

else if (config.useFacebook) {

view = 'facebookAuth'

postUrl = "${request.contextPath}${config.facebook.filterProcessesUrl}"

}

else {

view = 'auth'

postUrl = "${request.contextPath}${config.filterProcessesUrl}"

}

render view: view, model: [postUrl: postUrl]

}

/\*\*

\* Form submit action to start an OpenID authentication.

\*/

def openIdAuthenticate = {

String openID = params['j\_username']

try {

String returnToURL = RedirectUtils.buildRedirectUrl(

request, response, openIDAuthenticationProcessingFilter.filterProcessesUrl)

String redirectUrl = openIDConsumer.beginConsumption(request, openID, returnToURL)

redirect url: redirectUrl

}

catch (org.springframework.security.ui.openid.OpenIDConsumerException e) {

log.error "Consumer error: $e.message", e

redirect url: openIDAuthenticationProcessingFilter.authenticationFailureUrl

}

}

// Login page (function|json) for Ajax access.

def authAjax = {

nocache(response)

//this is example:

render """

<script type='text/javascript'>

(function() {

loginForm();

})();

</script>

"""

}

/\*\*

\* The Ajax success redirect url.

\*/

def ajaxSuccess = {

nocache(response)

render '{success: true}'

}

/\*\*

\* Show denied page.

\*/

def denied = {

if (isLoggedIn() && authenticationTrustResolver.isRememberMe(SCH.context?.authentication)) {

// have cookie but the page is guarded with IS\_AUTHENTICATED\_FULLY

redirect action: full, params: params

}

}

/\*\*

\* Login page for users with a remember-me cookie but accessing a IS\_AUTHENTICATED\_FULLY page.

\*/

def full = {

render view: 'auth', params: params,

model: [hasCookie: authenticationTrustResolver.isRememberMe(SCH.context?.authentication)]

}

// Denial page (data|view|json) for Ajax access.

def deniedAjax = {

//this is example:

render "{error: 'access denied'}"

}

/\*\*

\* login failed

\*/

def authfail = {

def username = session[AuthenticationProcessingFilter.SPRING\_SECURITY\_LAST\_USERNAME\_KEY]

def msg = ''

def exception = session[AbstractProcessingFilter.SPRING\_SECURITY\_LAST\_EXCEPTION\_KEY]

if (exception) {

if (exception instanceof DisabledException) {

msg = "[$username] не поддтвержден"

}

else {

msg = "[$username] неверные логин/пароль"

}

}

if (isAjax()) {

render "{error: '${msg}'}"

}

else {

flash.message = msg

redirect uri: "/", params: params

}

}

/\*\*

\* Check if logged in.

\*/

private boolean isLoggedIn() {

return authenticateService.isLoggedIn()

}

private boolean isAjax() {

return authenticateService.isAjax(request)

}

/\*\* cache controls \*/

private void nocache(response) {

response.setHeader('Cache-Control', 'no-cache') // HTTP 1.1

response.addDateHeader('Expires', 0)

response.setDateHeader('max-age', 0)

response.setIntHeader('Expires', -1) //prevents caching at the proxy server

response.addHeader('cache-Control', 'private') //IE5.x only

}

}

package soc

/\*\*

\* Logout Controller (Example).

\*/

class LogoutController {

/\*\*

\* Index action. Redirects to the Spring security logout uri.

\*/

def index = {

// TODO put any pre-logout code here

redirect(uri: '/j\_spring\_security\_logout')

}

}

package soc

class PostController {

def authenticateService

def scaffold = true

def index = {}

def createPost() {

Community community = Community.get(params.community\_id)

if (!community) redirect(controller: "community", action: "allComm")

[post: new Post(params), community\_id: community.id]

}

def savePost() {

Community community = Community.get(params.community\_id)

User user = authenticateService.userDomain() as User

if (!community || !user) redirect(controller: "community", action: "allComm")

Post post = new Post(params)

if (post.validate()) {

String content = params.content

Member member = Member.findByUser(user)

post.author = member

Comment comment = new Comment(content: content, master: user).save()

if (!comment) {

render(view: "createPost", model: [post: post])

}

post = post.addToComments(comment).save()

community.addToPosts(post)

redirect(controller: "community", action: "showComm", id: community.shortName)

} else {

render(view: "createPost", model: [post: post])

}

}

def newComment() {

Post post = Post.get(params.id)

User user = authenticateService.userDomain() as User

def g = params

String content = params.coment

Comment comment = new Comment(content: content, master: user)

if (comment.validate()) {

post.addToComments(comment)

post.save()

} else {

flash.error\_message = "Ошибка добавления поста!"

}

redirect(action: "showPost", id: "${params.id}")

}

def showPost = {

Community communityInstance = Community.get(session.comm\_id)

Post postInstance = Post.get(params.id)

if (!communityInstance) redirect(controller: "community", action: "allComm")

if (!postInstance) {

redirect(controller: "community", action: "showComm", id: "${communityInstance.shortName}")

} else {

List<Comment> comments = postInstance.comments as List<Comment>

List rectors = flash.rectors

List teachers = flash.teachers

List students = flash.students

List users = flash.users

flash.rectors = rectors

flash.teachers = teachers

flash.students = students

flash.users = users

List sortComments = comments.sort {comm1, comm2 ->

comm1.lastUpdated <=> comm2.lastUpdated

}

Comment firstComment = sortComments[0]

session.comments = sortComments

session.firstComment = firstComment

[post: postInstance, firstComment: firstComment, comments: sortComments[1..<sortComments.size()], community: communityInstance, rectors: rectors, teachers: teachers, students: students,

users: users]

}

}

def ajaxPost = {

Community communityInstance = Community.get(session.comm\_id)

def point = 0

if (communityInstance) {

params.max = Math.min(params.max ? params.int('max') : 7, 100)

List posts = communityInstance.posts as List

int offset = (params.offset ?: 0) as int

def totalPost = posts.size()

int newTotalPosts = totalPost.toInteger() - 1

int newMax = params.int('max') + offset.toInteger() - 1

def max = Math.min(newTotalPosts, newMax)

List subList = posts[offset..max]

def meow = 0

render(template: "listPosts", model: [posts: subList, totalPost: totalPost])

} else {

redirect(action: "allComm")

}

}

}

package soc

import org.springframework.security.providers.UsernamePasswordAuthenticationToken as AuthToken

import org.springframework.security.context.SecurityContextHolder as SCH

/\*\*

\* Registration controller.

\*/

class RegisterController {

def authenticateService

def daoAuthenticationProvider

def emailerService

def rolesService

static Map allowedMethods = [save: 'POST', update: 'POST']

/\*\*

\* User Registration Top page.

\*/

def index = {

// skip if already logged in

if (authenticateService.isLoggedIn()) {

redirect action: show

return

}

if (session.id) {

def person = new User()

person.properties = params

return [person: person]

}

redirect uri: '/'

}

/\*\*

\* User Information page for current user.

\*/

def show = {

// get user id from session's domain class.

User user = authenticateService.userDomain() as User

if (user) {

render view: 'show', model: [person: User.get(user.id)]

}

else {

redirect action: index

}

}

/\*\*

\* Edit page for current user.

\*/

def edit = {

User person

User user = authenticateService.userDomain() as User

if (user) {

person = User.get(user.id)

}

if (!person) {

flash.message = "[Illegal Access] User not found with id ${params.id}"

redirect action: index

return

}

[person: person]

}

/\*\*

\* update action for current user's edit page

\*/

def update = {

def person

User user = authenticateService.userDomain() as User

if (user) {

person = User.get(user.id)

}

else {

redirect action: index

return

}

if (!person) {

flash.message = "[Illegal Access] User not found with id ${params.id}"

redirect action: index, id: params.id

return

}

// if user want to change password. leave passwd field blank, passwd will not change.

if (params.password && params.password.length() > 0

&& params.repassword && params.repassword.length() > 0) {

if (params.password == params.repassword) {

person.password = authenticateService.encodePassword(params.passwd)

}

else {

person.password = ''

flash.message = 'The passwords you entered do not match.'

render view: 'edit', model: [person: person]

return

}

}

person.userRealName = params.userRealName

person.email = params.email

if (params.emailShow) {

person.emailShow = true

}

else {

person.emailShow = false

}

if (person.save()) {

redirect action: show, id: person.id

}

else {

render view: 'edit', model: [person: person]

}

}

/\*\*

\* Person save action.

\*/

def save = {

// skip if already logged in

if (authenticateService.isLoggedIn()) {

redirect action: show

return

}

def person = new User()

person.properties = params

def config = authenticateService.securityConfig

def defaultRole = config.security.defaultRole

def role = Role.findByAuthority(defaultRole)

if (!role) {

person.password = ''

flash.message = 'Default Role not found.'

render view: 'index', model: [person: person]

return

}

if (params.captcha.toUpperCase() != session.captcha) {

person.password = ''

flash.message = 'Код введён неверно!.'

render view: 'index', model: [person: person]

return

}

if (params.password != params.repassword) {

person.password = ''

flash.message = 'The passwords you entered do not match.'

render view: 'index', model: [person: person]

return

}

def pass = authenticateService.encodePassword(params.password)

person.password = pass

person.enabled = true

person.emailShow = true

person.description = ''

person = person.save()

if (person) {

Role newRole = Role.findByAuthority(rolesService.user)

// now add the User to the role

newRole.addToPeople(person)

// Записываем всё в базу

newRole.save()

if (config.security.useMail) {

String emailContent = """You have signed up for an account at:

${request.scheme}://${request.serverName}:${request.serverPort}${request.contextPath}

Here are the details of your account:

-------------------------------------

LoginName: ${person.login}

Email: ${person.email}

Full Name: ${person.userRealName}

Password: ${params.password}

"""

def email = [

to: [person.email], // 'to' expects a List, NOT a single email address

subject: "[${request.contextPath}] Account Signed Up",

text: emailContent // 'text' is the email body

]

emailerService.sendEmails([email])

}

person.save(flush: true)

def auth = new AuthToken(person.login, params.password)

def authtoken = daoAuthenticationProvider.authenticate(auth)

SCH.context.authentication = authtoken

redirect uri: '/'

}

else {

person.password = ''

render view: 'index', model: [person: person]

}

}

}

package soc

/\*\*

\* soc.Requestmap controller.

\*/

class RequestmapController {

def authenticateService

// the delete, saveComm and update actions only accept POST requests

static Map allowedMethods = [delete: 'POST', save: 'POST', update: 'POST']

def index = {

redirect action: list, params: params

}

def list = {

if (!params.max) {

params.max = 10

}

[requestmapList: Requestmap.list(params)]

}

def show = {

def requestmap = Requestmap.get(params.id)

if (!requestmap) {

flash.message = "soc.Requestmap not found with id $params.id"

redirect action: list

return

}

[requestmap: requestmap]

}

def delete = {

def requestmap = Requestmap.get(params.id)

if (!requestmap) {

flash.message = "soc.Requestmap not found with id $params.id"

redirect action: list

return

}

requestmap.delete()

authenticateService.clearCachedRequestmaps()

flash.message = "soc.Requestmap $params.id deleted."

redirect(action: list)

}

def edit = {

def requestmap = Requestmap.get(params.id)

if (!requestmap) {

flash.message = "soc.Requestmap not found with id $params.id"

redirect(action: list)

return

}

[requestmap: requestmap]

}

/\*\*

\* Update action, called when an existing soc.Requestmap is updated.

\*/

def update = {

def requestmap = Requestmap.get(params.id)

if (!requestmap) {

flash.message = "soc.Requestmap not found with id $params.id"

redirect(action: edit, id: params.id)

return

}

long version = params.version.toLong()

if (requestmap.version > version) {

requestmap.errors.rejectValue 'version', "requestmap.optimistic.locking.failure",

"Another user has updated this soc.Requestmap while you were editing."

render view: 'edit', model: [requestmap: requestmap]

return

}

requestmap.properties = params

if (requestmap.save()) {

authenticateService.clearCachedRequestmaps()

redirect action: show, id: requestmap.id

}

else {

render view: 'edit', model: [requestmap: requestmap]

}

}

def create = {

[requestmap: new Requestmap(params)]

}

/\*\*

\* Save action, called when a new soc.Requestmap is created.

\*/

def save = {

def requestmap = new Requestmap(params)

if (requestmap.save()) {

authenticateService.clearCachedRequestmaps()

redirect action: show, id: requestmap.id

}

else {

render view: 'create', model: [requestmap: requestmap]

}

}

}

package soc

import soc.Role

/\*\*

\* Authority Controller.

\*/

class RoleController {

// the delete, saveComm and update actions only accept POST requests

static Map allowedMethods = [delete: 'POST', save: 'POST', update: 'POST']

def authenticateService

def index = {

redirect action: list, params: params

}

/\*\*

\* Display the allComm authority page.

\*/

def list = {

if (!params.max) {

params.max = 10

}

[authorityList: Role.list(params)]

}

/\*\*

\* Display the showComm authority page.

\*/

def show = {

def authority = Role.get(params.id)

if (!authority) {

flash.message = "soc.Role not found with id $params.id"

redirect action: list

return

}

[authority: authority]

}

/\*\*

\* Delete an authority.

\*/

def delete = {

def authority = Role.get(params.id)

if (!authority) {

flash.message = "soc.Role not found with id $params.id"

redirect action: list

return

}

authenticateService.deleteRole(authority)

flash.message = "soc.Role $params.id deleted."

redirect action: list

}

/\*\*

\* Display the edit authority page.

\*/

def edit = {

def authority = Role.get(params.id)

if (!authority) {

flash.message = "soc.Role not found with id $params.id"

redirect action: list

return

}

[authority: authority]

}

/\*\*

\* Authority update action.

\*/

def update = {

def authority = Role.get(params.id)

if (!authority) {

flash.message = "soc.Role not found with id $params.id"

redirect action: edit, id: params.id

return

}

long version = params.version.toLong()

if (authority.version > version) {

authority.errors.rejectValue 'version', 'authority.optimistic.locking.failure',

'Another user has updated this soc.Role while you were editing.'

render view: 'edit', model: [authority: authority]

return

}

if (authenticateService.updateRole(authority, params)) {

authenticateService.clearCachedRequestmaps()

redirect action: show, id: authority.id

}

else {

render view: 'edit', model: [authority: authority]

}

}

/\*\*

\* Display the newComm new authority page.

\*/

def create = {

[authority: new Role()]

}

/\*\*

\* Save a new authority.

\*/

def save = {

def authority = new Role()

authority.properties = params

if (authority.save()) {

redirect action: show, id: authority.id

}

else {

render view: 'create', model: [authority: authority]

}

}

}

package soc

class SecretController {

def index() { }

}

package soc

import javax.servlet.http.HttpServletResponse

class UserController {

def scaffold = true

def authenticateService

def rolesService

def secureService

def register = {

if (params) {

if (params.password) {

params.password = authenticateService.encodePassword(params.password)

}

def user = new User(params)

if (user.validate()) {

if (params.captcha?.toUpperCase() != session.captcha) {

user.password = ""

flash.message = "Неправильный код"

return [user: user]

}

user = user.save()

Role role = Role.findById(rolesService.USER)

role.addToPeople(user)

role.save()

flash.message = "Вы успешно зарегистрировались"

redirect(uri: '/success')

} else {

user.password = ""

flash.message = "Ошибка регистрации"

return [user: user]

}

}

}

def showProfile = {

User user = User.get(params.id)

if (!user) {

response.sendError(HttpServletResponse.SC\_NOT\_FOUND); // 404

return

}

User myUser = authenticateService.userDomain() as User

int userRating = user?.profile?.rating?.value ?: 0

userRating /= 1000000 / 100

List<Comment> comments = Comment.findAllByMaster(user, [offset: 0, max: 10])

[user: user, myUser: myUser, userRating: userRating]

}

def changeRole = {

List users = User.findAll()

List roles = Role.findAll()

[users: users, roles: roles]

}

def saveRole = {

def p = params

String role\_id = params.role\_id == "null" ? "" : params.role\_id

if (role\_id.isEmpty()) {

flash.error\_message = "Роль #${params.role\_id} отсутствует!"

redirect(action: "changeRole")

return

}

User user = User.get(params.user\_id)

Role role = Role.get(params.role\_id)

if (!user) {

flash.error\_message = "Пользователь #${params.user\_id} не найден!"

redirect(action: "changeRole")

return

}

if (!role) {

flash.error\_message = "Роль #${params.role\_id} отсутствует!"

redirect(action: "changeRole")

return

}

User myUser = authenticateService.userDomain() as User

if (user.id == myUser.id) {

flash.error\_message = "Вы не можете изменить роль самому себе!"

redirect(action: "changeRole")

return

}

user.save()

Role.findAll().each { it.removeFromPeople(user) }

Role.findByAuthority(role.authority).addToPeople(user)

user = User.get(params.user\_id)

if (!user.profile) {

UserRating \_rating = new UserRating().save()

UserProfile prof = new UserProfile(aboutMe: "Новый пользователь", birthday: new Date(0, 0, 0),

name: "Новый", surname: "Пользователь", middle\_name: "Очень", rating: \_rating,

work: "Новый пользователь", state: "Здравствуйте!").save()

user.profile = prof

} else {

user.profile.editable = false

}

user.save()

flash.success\_message = "Роль пользователя #${user.id} успешно изменена на ${role.toString()}!"

redirect(action: "changeRole")

}

def test = {

ArrayList<String> keys = secureService.getGeneratedKeys(10, 10)

keys.clear()

}

}

**Представления**

<nav>

<table width="100%" align="center">

<tr>

<td><a href="<g:createLink action="index"/>">О проекте</a></td>

<td><a href="<g:createLink action="aboutInstitute"/>">Об институте</a></td>

<td><a class="active" href="<g:createLink action="aboutCathedra"/>">О кафедре</a></td>

<td><a href="<g:createLink controller="community" action="allComm"/>">Сообщества</a></td>

<td><a href="<g:createLink action="contacts"/>">Контакты</a></td>

</tr>

</table>

</nav>

<nav>

<table width="100%" align="center">

<tr>

<td><a href="<g:createLink controller="basic"/>">О проекте</a></td>

<td><a href="<g:createLink controller="basic" action="aboutInstitute"/>">Об институте</a></td>

<td><a href="<g:createLink controller="basic" action="aboutCathedra"/>">О кафедре</a></td>

<td><a class="active" href="<g:createLink controller="community" action="allComm"/>">Сообщества</a></td>

<td><a href="<g:createLink controller="basic" action="contacts"/>">Контакты</a></td>

</tr>

</table>

</nav>

<nav>

<table width="100%" align="center">

<tr>

<td><a href="<g:createLink action="index"/>">О проекте</a></td>

<td><a href="<g:createLink action="aboutInstitute"/>">Об институте</a></td>

<td><a href="<g:createLink action="aboutCathedra"/>">О кафедре</a></td>

<td><a href="<g:createLink controller="community" action="allComm"/>">Сообщества</a></td>

<td><a class="active" href="<g:createLink action="contacts"/>">Контакты</a></td>

</tr>

</table>

</nav>

<nav>

<table width="100%" align="center">

<tr>

<td><a href="<g:createLink action="index"/>">О проекте</a></td>

<td><a class="active" href="<g:createLink action="aboutInstitute"/>">Об институте</a>

</td>

<td><a href="<g:createLink action="aboutCathedra"/>">О кафедре</a></td>

<td><a href="<g:createLink controller="community" action="allComm"/>">Сообщества</a></td>

<td><a href="<g:createLink action="contacts"/>">Контакты</a></td>

</tr>

</table>

</nav>

<nav>

<table width="100%" align="center">

<tr>

<td><a href="<g:createLink controller="basic" action="index"/>">О проекте</a></td>

<td><a href="<g:createLink controller="basic" action="aboutInstitute"/>">Об институте</a></td>

<td><a href="<g:createLink controller="basic" action="aboutCathedra"/>">О кафедре</a></td>

<td><a href="<g:createLink controller="community" action="allComm"/>">Сообщества</a></td>

<td><a href="<g:createLink controller="basic" action="contacts"/>">Контакты</a></td>

</tr>

</table>

</nav>

<nav>

<table width="100%" align="center">

<tr>

<td><a class="active" href="<g:createLink action="index"/>">О проекте</a></td>

<td><a href="<g:createLink action="aboutInstitute"/>">Об институте</a></td>

<td><a href="<g:createLink action="aboutCathedra"/>">О кафедре</a></td>

<td><a href="<g:createLink controller="community" action="allComm"/>">Сообщества</a></td>

<td><a href="<g:createLink action="contacts"/>">Контакты</a></td>

</tr>

</table>

</nav>

<div id="registration"><b>Здравствуйте!</b><br>

<a href=""><g:img dir="images/soc" file="nophoto.png"/>

</a><g:fullName/><br>

<i><g:fullStatus/></i>

</div>

<div class="right\_menu">

<g:ifAnyGranted role="ROLE\_ADMIN">

<a href="${createLink(controller: "secret")}">Администраторская</a><br>

</g:ifAnyGranted>

<a href="">Моя страница</a><br>

<a href="">Мои сообщества</a><br>

<a href="">Редактировать профиль</a><br>

<a href="${createLink(controller: "logout")}">Выйти</a><br>

</div>

<div class="left\_menu">

<ul>

<li><a href="">Моя страница</a></li>

<li><a href="">Мои сообщества</a></li>

<li><a href="${createLink(controller: "basic", action: "logout")}">Выйти</a></li>

</ul>

</div>

<div id="registration">

<g:if test='${flash.message}'>

<div class='login\_message' style="color: #ff273d;">${flash.message}</div>

</g:if>

<form method="POST" action="${resource(file: 'j\_spring\_security\_check')}" class="cssform">

<p>

<label for="j\_username">Логин</label><br>

<g:textField id="j\_username" name="j\_username"/>

</p>

<p>

<label for="j\_password">Пароль</label><br>

<input id="j\_password" type="password" class='text\_' name='j\_password'/>

</p>

<p>

<input type="checkbox" id="remember\_me" class="chk" name="\_spring\_security\_remember\_me"

<g:if test='${hasCookie}'>checked='checked'</g:if>/>

<label for="remember\_me">Запомнить?</label>

</p>

<g:submitButton name="login" type="submit" class="sub\_button" value="Войти"/>

</form>

</div>

<script type='text/javascript'>

<!--

(function () {

document.forms['loginForm'].elements['j\_username'].focus();

})();

// -->

</script>

<div class="right\_menu">

<a href="${createLink(controller: "register", action: "index")}">Регистрация</a>

<a href="">Забыли пароль?</a>

</div>

<!doctype html>

<html>

<head>

<title>О кафедре</title>

</head>

<body>

<g:render template="navs/cathedra"/>

<article>

<div class="about">

<h1>Факультеты и кафедры</h1> <br>

<div><h2><a href="http://miem.hse.ru/edu/fet/">Факультет электроники и телекоммуникаций</a></h2>

<p class="text">Декан факультета: лауреат Государственной премии СССР, доктор технических наук, профессор <a href="http://www.hse.ru/org/persons/47632869">Львов Борис Глебович</a>.</p>

<p class="text">Телефон: (499) 235-88-84. E-mail: <a href="mailto:blvov@hse.ru">blvov@hse.ru</a>.</p>

<ul>

<li><a href="http://miem.hse.ru/edu/fet/ein/">Кафедра электроники и наноэлектроники</a></li>

<li><a href="http://miem.hse.ru/edu/fet/rt/">Кафедра радиоэлектроники и телекоммуникаций</a></li>

<li><a href="http://miem.hse.ru/edu/fet/61217518/">Кафедра микросистемной техники, материаловедения и технологий</a></li>

</ul>

<h2>&nbsp;</h2>

<h2><a href="http://miem.hse.ru/edu/itvt/">Факультет информационных технологий и вычислительной техники</a></h2>

<p class="text">Декан факультета: доктор технических наук, профессор, академик Академии проблем качества, заслуженный деятель науки и техники России, лауреат премии Правительства РФ <a href="http://www.hse.ru/org/persons/47633983">Пожидаев Евгений Дмитриевич</a>.</p>

<p class="text">Телефон: (495) 917-15-96, (495) 916-88-50. E-mail: <a href="mailto:epozhidaev@hse.ru">epozhidaev@hse.ru</a>.</p>

<ul>

<li><a href="http://miem.hse.ru/edu/itvt/vss/">Кафедра вычислительных систем и сетей</a></li>

<li><a href="http://miem.hse.ru/edu/itvt/ikt/">Кафедра информационно-коммуникационных технологий</a></li>

<li><a href="http://miem.hse.ru/edu/itvt/itas/">Кафедра информационных технологий и автоматизированных систем</a></li>

</ul>

<h2>&nbsp;</h2>

<h2><a href="http://miem.hse.ru/edu/pmk/">Факультет прикладной математики и кибернетики</a></h2>

<p class="text">Декан факультета: кандидат технических наук, доцент, <a href="http://www.hse.ru/org/persons/47634735">Белов Александр Владимирович</a>.</p>

<p class="text">Телефон: (495) 916-24-47, (495) 916-88-97. E-mail: <a href="mailto:avbelov@hse.ru">avbelov@hse.ru</a>.</p>

<ul>

<li><a href="http://miem.hse.ru/edu/pmk/pm/">Кафедра прикладной математики</a></li>

<li><a href="http://miem.hse.ru/edu/pmk/61217880/">Кафедра кибернетики</a></li>

<li><a href="http://miem.hse.ru/edu/pmk/mm/">Кафедра механики и математического моделирования</a></li>

<li><a href="http://miem.hse.ru/edu/pmk/cosec/">Кафедра компьютерной безопасности</a></li>

</ul>

<h2>&nbsp;</h2>

<h2><a href="http://design.hse.ru/">Отделение дизайн</a></h2>

<p class="text">Заведующая отделением: доктор педагогических, доцент, член Творческого союза художников России, Аристова Ульяна Викторовна.</p>

<p class="text">E-mail: <a href="mailto:uaristova@hse.ru">uaristova@hse.ru</a>.</p>

<h2>&nbsp;</h2>

<h2>Общеинститутские кафедры</h2>

<ul>

<li><a href="http://miem.hse.ru/edu/oik/higher\_math/">Кафедра высшей математики</a></li>

<li><a href="http://miem.hse.ru/edu/oik/physics/">Кафедра физики</a></li>

<li><a href="http://pce.miem.hse.ru/">Кафедра физической химии и экологии</a></li>

<li><a href="http://miem.hse.ru/edu/oik/lang/">Кафедра иностранных языков</a></li>

</ul>

<h2>&nbsp;</h2>

<h2>Общеинститутские учебно-исследовательские лаборатории</h2>

<ul>

<li>Лаборатория высокопроизводительных аппаратно-программных комплексов и локальных вычислительных сетей</li>

<li><a href="http://astro.miem.hse.ru/">Лаборатория космических систем и технологий</a></li>

<li><a href="http://miem.hse.ru/uvpmiem/lab/internetekcon/">Лаборатория интернет-технологий и сервисов</a></li>

<li><a href="http://miem.hse.ru/uvpmiem/lab/ka/">Лаборатория функциональной безопасности космических аппаратов и систем</a></li>

<li>Мультивендорный центр</li>

</ul>

<p><a href="<g:createLink action="toHome"/>">Источник</a></p>

</div>

</article>

</body>

</html>

</html>

<%@ page import="soc.Community" %>

<div class="list">

<g:each var="post" in="${posts}">

<div class="tema">

<g:link class="tema\_name" controller="post" action="showPost" id="${post.id}">${post.name}</g:link><br>

${post.comments.size()} Последнее от <a href="">Игоря Петрова</a><br>

<i>вчера</i>

</div>

</g:each>

</div>

<div class="paginateButtons">

<util:remotePaginate total="${totalPost}"

update="filteredList" action="ajaxPost"

alwaysShowPageSizes="false"/>

</div>

<%@ page import="soc.Community" %>

<div class="list">

<g:each var="community" status="i" in="${communityInstanceList}">

<div class="tema">

<a style="text-decoration: none !important; color: #333; font-weight: bold">${community.shortName} (${community.type})</a><br>

Всего пользователей: ${community.members.size()}<br>

Описание:

<i>${community.c\_describe}</i><br>

<g:link action="showComm" id="${community.shortName}">Перейти на страницу сообщества</g:link>

</div>

</g:each>

</div>

<div class="paginateButtons">

<util:remotePaginate total="${communityInstanceTotal}"

update="filteredList" action="filter"

alwaysShowPageSizes="false"/>

</div>

<div class="right\_block">

<g:if test="${rectors}">

<div class="user\_list">

<h2>Управляющие:</h2>

<g:each var="rector" in="${rectors}">

<a href=""><img

src="${resource(dir: 'images/soc', file: 'nophoto.png')}"

alt="Logo"/><g:fullName user="${rector}"/></a><br>

</g:each>

</div>

</g:if>

<g:if test="${teachers}">

<div class="user\_list">

<h2>Преподаватели:</h2>

<g:each var="teacher" in="${teachers}">

<a href=""><img

src="${resource(dir: 'images/soc', file: 'nophoto.png')}"

alt="Logo"/><g:fullName user="${teacher}"/></a><br>

</g:each>

</div>

</g:if>

<g:if test="${students}">

<div class="user\_list">

<h2>Студенты:</h2>

<g:each var="student" in="${students}">

<a href=""><img

src="${resource(dir: 'images/soc', file: 'nophoto.png')}"

alt="Logo"/><g:fullName user="${student}"/></a><br>

</g:each>

</div>

</g:if>

<g:if test="${users}">

<div class="user\_list">

<h2>Пользователи:</h2>

<g:each var="user" in="${users}">

<a href=""><img

src="${resource(dir: 'images/soc', file: 'nophoto.png')}"

alt="Logo"/><g:fullName user="${user}"/></a><br>

</g:each>

</div>

</g:if>

</div>

<%@ page import="soc.Community" %>

<!doctype html>

<html>

<head>

<meta name="layout" content="basic">

<title>Список сообществ</title>

</head>

<body>

<g:render template="/basic/navs/comms"/>

<g:isNotLoggedIn>

<article>

<div class="about">

<h1>Страница доступна только <a

href="${createLink(controller: "user", action: "register")}">зарегистированным</a> пользователям.

</h1>

<p>Пожалуйста, воспользуйтесь формой входа.</p>

<p>Она находится в правом верхнем углу.</p>

</div>

</article>

</g:isNotLoggedIn>

<g:isLoggedIn>

<article>

<div class="title">

<h1>Список сообществ</h1>

</div>

<div class="wrapper">

<div class="main\_block">

<h2>Всего сообществ: ${communityInstanceTotal}</h2><br>

<g:ifNotGranted role="ROLE\_USER">

<a href="${createLink(action: "newComm")}"><b>Создать сообщество</b></a><br><br>

</g:ifNotGranted>

<div id="filteredList">

<g:render template="listTemplate"/>

</div>

</div>

</div>

<g:render template="/basic/leftMenu"/>

<div class="clear"></div>

</article>

</g:isLoggedIn>

</body>

</html>

<%@ page import="soc.Community" %>

<!doctype html>

<html>

<head>

<meta name="layout" content="basic">

<g:set var="entityName" value="${message(code: 'community.label', default: 'Community')}"/>

<title><g:message code="default.list.label" args="[entityName]"/></title>

</head>

<body>

<nav>

<table width="100%" align="center">

<tr>

<td><a href="<g:createLink controller='basic' action='index'/>">О проекте</a></td>

<td><a href="<g:createLink controller="basic" action="aboutInstitute"/>">Об институте</a></td>

<td><a href="<g:createLink controller="basic" action="aboutCathedra"/>">О кафедре</a></td>

<td><a class="active" href="<g:createLink controller="community" action="allComm"/>">Сообщества</a></td>

<td><a href="<g:createLink controller="basic" action="contacts"/>">Контакты</a></td>

</tr>

</table>

</nav>

<article>

<div class="title">

<h1>Создание нового сообщества</h1>

</div>

<div class="wrapper">

<div class="main\_block">

<g:form action="saveComm">

<dl>

<dt>Название\*</dt>

<dt><g:textField name="name" value="${communityInstance?.name}"/></dt>

<g:hasErrors bean="${communityInstance}" field="name">

<g:eachError bean="${communityInstance}" field="name">

<p style="color: red;">

<g:message error="${it}"/>

</p>

</g:eachError>

</g:hasErrors>

<dt>Абривиатура\*</dt>

<dt><g:textField name="shortName" value="${communityInstance?.shortName}"/></dt>

<g:hasErrors bean="${communityInstance}" field="shortName">

<g:eachError bean="${communityInstance}" field="shortName">

<p style="color: red;">

<g:message error="${it}"/>

</p>

</g:eachError>

</g:hasErrors>

<dt>Описание\*</dt>

<dt><g:textArea cols="25" rows="10" name="c\_describe"

value="${communityInstance?.c\_describe}"/></dt>

<g:hasErrors bean="${communityInstance}" field="c\_describe">

<g:eachError bean="${communityInstance}" field="c\_describe">

<p style="color: red;">

<g:message error="${it}"/>

</p>

</g:eachError>

</g:hasErrors>

<dt>Тип сообщества\*</dt>

<dt>

<g:select name="type" from="${communityInstance.constraints.type.inList}"

value="${communityInstance.type}" valueMessagePrefix="go.type"/>

</dt>

<dt>Анонимность сообщества\*</dt>

<dt>

<g:select name="privateType" from="${communityInstance.constraints.privateType.inList}"

value="${communityInstance.privateType}" valueMessagePrefix="go.privateType"/>

</dt>

<dt>Учебное отделение(если есть)</dt>

<dt>

<g:select id="studyingDivision" name="studyingDivision.id"

from="${soc.TrainingDepartment.list()}"

optionKey="id"

value="${communityInstance?.studyingDivision?.id}" class="many-to-one"

noSelection="['null': 'выберите ...']"/>

</dt>

<br>

<dt><g:submitButton name="save" value="Создать"/></dt>

</dl>

</g:form>

</div>

</div>

<div class="left\_menu">

<ul>

<li><a href="">Моя страница</a></li>

<li><a href="">Мои сообщества</a></li>

<li><a href="${createLink(controller: "basic", action: "logout")}">Выйти</a></li>

</ul>

</div>

<div class="clear"></div>

</article>

</body>

</html>

<%@ page import="soc.Community" %>

<!doctype html>

<html>

<head>

<meta name="layout" content="basic">

<title>${community.name}</title>

</head>

<body>

<g:render template="/basic/navs/stable"/>

<article>

<div class="title">

<h1>"${community.name}"</h1>

</div>

<div class="wrapper">

<div class="main\_block">

<div class="tema">

<b>Описание:</b>

${community.c\_describe}

<br><br>

<b>Тип:</b>

${community.type}

</div>

<h2>Темы насущные:</h2>

<br>

<g:if test="${true}">

<a href="${createLink(controller: "post", action: "createPost", params: [community\_id: community.id])}"><b>Создать тему</b>

</a><br><br>

</g:if>

<div id="filteredList">

<g:render template="listPosts"/>

</div>

</div>

</div>

<g:render template="/basic/leftMenu"/>

<g:render template="rightBlock"

model="${[rectors: rectors, teachers: teachers, students: students, users: users]}"/>

<div class="clear"></div>

</article>

</body>

</html>

<!doctype html>

<!--[if lt IE 7 ]> <html lang="en" class="no-js ie6"> <![endif]-->

<!--[if IE 7 ]> <html lang="en" class="no-js ie7"> <![endif]-->

<!--[if IE 8 ]> <html lang="en" class="no-js ie8"> <![endif]-->

<!--[if IE 9 ]> <html lang="en" class="no-js ie9"> <![endif]-->

<!--[if (gt IE 9)|!(IE)]><!--> <html lang="en" class="no-js"><!--<![endif]-->

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">

<title>Social Learning :: <g:layoutTitle default="Welcome"/></title>

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<link rel="stylesheet" href="${resource(dir: 'css', file: 'style.css')}" type="text/css">

<g:javascript library="jquery" plugin="jquery"/>

<g:setProvider library="jquery"/>

<g:javascript library='application'/>

<g:layoutHead/>

<r:layoutResources/>

</head>

<body>

<div id="main">

<header>

<div id="logo"><a href="<g:createLink controller='basic'/>"><img

src="${resource(dir: 'images/soc', file: 'logo.png')}" alt="Logo"/></a></div>

<g:isNotLoggedIn>

<g:render template="/basic/login"/>

<g:render template="/basic/rightMenu"/>

</g:isNotLoggedIn>

<g:isLoggedIn>

<g:render template="/basic/fullInfo"/>

</g:isLoggedIn>

</header>

<g:layoutBody/>

<footer>

Social Learning MIEM 2012

</footer>

<g:javascript library="application"/>

<r:layoutResources/>

</div>

</body>

</html>

<!doctype html>

<!--[if lt IE 7 ]> <html lang="en" class="no-js ie6"> <![endif]-->

<!--[if IE 7 ]> <html lang="en" class="no-js ie7"> <![endif]-->

<!--[if IE 8 ]> <html lang="en" class="no-js ie8"> <![endif]-->

<!--[if IE 9 ]> <html lang="en" class="no-js ie9"> <![endif]-->

<!--[if (gt IE 9)|!(IE)]><!--> <html lang="en" class="no-js"><!--<![endif]-->

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">

<title>Social Learning <g:layoutTitle default="Welcome"/></title>

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<link rel="stylesheet" href="${resource(dir: 'css', file: 'style.css')}" type="text/css">

<g:javascript library='jquery'/>

<g:javascript library='application'/>

<g:javascript library='scriptaculous'/>

<g:layoutHead/>

<r:layoutResources/>

</head>

<body>

<div id="main">

<header>

<div id="logo"><a href="#"><img

src="${resource(dir: 'images/soc', file: 'logo.png')}" alt="Logo"/></a></div>

<div id="registration"><b>Форма авторизации</b><br>

<label for="login">Логин</label><br>

<input type="text" id="login" name="login"/><br>

<label for="pass">Пароль</label><br>

<input type="password" id="pass" name="pass"/><br>

<input type="checkbox" id="remember\_me" name="remember\_me"/><label for="remember\_me">Запомнить?</label>

<input class="sub\_button" type="submit" value="Войти"/>

</div>

<div class="right\_menu">

<a href="">Регистрация</a>

<a href="">Забыли пароль?</a>

</div>

</header>

<g:layoutBody/>

<g:javascript library="application"/>

<r:layoutResources/>

</div>

</body>

</html>

<!doctype html>

<!--[if lt IE 7 ]> <html lang="en" class="no-js ie6"> <![endif]-->

<!--[if IE 7 ]> <html lang="en" class="no-js ie7"> <![endif]-->

<!--[if IE 8 ]> <html lang="en" class="no-js ie8"> <![endif]-->

<!--[if IE 9 ]> <html lang="en" class="no-js ie9"> <![endif]-->

<!--[if (gt IE 9)|!(IE)]><!--> <html lang="en" class="no-js"><!--<![endif]-->

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">

<title><g:layoutTitle default="Grails"/></title>

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<link rel="shortcut icon" href="${resource(dir: 'images', file: 'favicon.ico')}" type="image/x-icon">

<link rel="apple-touch-icon" href="${resource(dir: 'images', file: 'apple-touch-icon.png')}">

<link rel="apple-touch-icon" sizes="114x114" href="${resource(dir: 'images', file: 'apple-touch-icon-retina.png')}">

<link rel="stylesheet" href="${resource(dir: 'css', file: 'main.css')}" type="text/css">

<link rel="stylesheet" href="${resource(dir: 'css', file: 'mobile.css')}" type="text/css">

<g:layoutHead/>

<r:layoutResources/>

</head>

<body>

<div id="grailsLogo" role="banner"><a href="${createLink(controller: "basic")}"><img

src="${resource(dir: 'images/soc', file: 'logo.png')}" alt="Logo"/></a></div>

<g:layoutBody/>

<div class="footer" role="contentinfo"></div>

<div id="spinner" class="spinner" style="display:none;"><g:message code="spinner.alt" default="Loading&hellip;"/></div>

<g:javascript library="application"/>

<r:layoutResources/>

</body>

</html>

<%@ page contentType="text/html;charset=UTF-8" %>

<html>

<head>

<title>QOTD &raquo; <g:layoutTitle/></title>

<link rel="stylesheet" href="<g:createLinkTo dir='css' file='snazzy.css'/>"/>

<g:javascript library="jquery"/>

<r:layoutResources/>

<g:layoutHead/>

</head>

<body>

<div id="header">

<img src="<g:createLinkTo dir='images' file='logo.png'/>" alt="logo"/>

</div>

<g:layoutBody/>

</body>

</html>

<head>

<meta name='layout' content='main' />

<title>Login</title>

<style type='text/css' media='screen'>

#login {

margin:15px 0px; padding:0px;

text-align:center;

}

#login .inner {

width:260px;

margin:0px auto;

text-align:left;

padding:10px;

border-top:1px dashed #499ede;

border-bottom:1px dashed #499ede;

background-color:#EEF;

}

#login .inner .fheader {

padding:4px;margin:3px 0px 3px 0;color:#2e3741;font-size:14px;font-weight:bold;

}

#login .inner .cssform p {

clear: left;

margin: 0;

padding: 5px 0 8px 0;

padding-left: 105px;

border-top: 1px dashed gray;

margin-bottom: 10px;

height: 1%;

}

#login .inner .cssform input[type='text'] {

width: 120px;

}

#login .inner .cssform label {

font-weight: bold;

float: left;

margin-left: -105px;

width: 100px;

}

#login .inner .login\_message {color:red;}

#login .inner .text\_ {width:120px;}

#login .inner .chk {height:12px;}

</style>

</head>

<body>

<div id='login'>

<div class='inner'>

<g:if test='${flash.message}'>

<div class='login\_message'>${flash.message}</div>

</g:if>

<div class='fheader'>Please Login..</div>

<form action='${postUrl}' method='POST' id='loginForm' class='cssform'>

<p>

<label for='j\_username'>Login ID</label>

<input type='text' class='text\_' name='j\_username' id='j\_username' value='${request.remoteUser}' />

</p>

<p>

<label for='j\_password'>Password</label>

<input type='password' class='text\_' name='j\_password' id='j\_password' />

</p>

<p>

<label for='remember\_me'>Remember me</label>

<input type='checkbox' class='chk' name='\_spring\_security\_remember\_me' id='remember\_me'

<g:if test='${hasCookie}'>checked='checked'</g:if> />

</p>

<p>

<input type='submit' value='Login' />

</p>

</form>

</div>

</div>

<script type='text/javascript'>

<!--

(function(){

document.forms['loginForm'].elements['j\_username'].focus();

})();

// -->

</script>

</body>

<%@ page contentType="text/html;charset=UTF-8" %>

<html>

<head>

<title>Ошибка доступа 403!</title>

<meta name="layout" content="basic">

</head>

<body>

<g:render template="/basic/navs/stable"/>

<article>

<div class="about" style="margin: 0 30% 0 30%;">

<h1>К сожалению, эта страница вам не доступна!</h1>

<p>Ошибка доступа (403).</p>

<p><img src="${resource(dir: 'images/soc', file: 'error403.jpg')}" alt="error403"/></p>

</div>

</article>

</body>

</html>

<head>

<meta name='layout' content='main' />

<title>Login</title>

<style type='text/css' media='screen'>

#login {

margin:15px 0px; padding:0px;

text-align:center;

}

#login .inner {

width:260px;

margin:0px auto;

text-align:left;

padding:10px;

border-top:1px dashed #499ede;

border-bottom:1px dashed #499ede;

background-color:#EEF;

}

#login .inner .fheader {

padding:4px;margin:3px 0px 3px 0;color:#2e3741;font-size:14px;font-weight:bold;

}

#login .inner .cssform p{

clear: left;

margin: 0;

padding: 5px 0 8px 0;

padding-left: 105px;

border-top: 1px dashed gray;

margin-bottom: 10px;

height: 1%;

}

#login .inner .cssform input[type='text']{

width: 120px;

}

#login .inner .cssform label{

font-weight: bold;

float: left;

margin-left: -105px;

width: 100px;

}

#login .inner .login\_message {color:red;}

#login .inner .text\_ {width:120px;}

</style>

</head>

<body>

<div id='login'>

<div class='inner'>

<g:if test='${flash.message}'>

<div class='login\_message'>${flash.message}</div>

</g:if>

<div class='fheader'>Please Login..</div>

<form action='${postUrl}' method='POST' id='loginForm' class='cssform'>

<p>

<label for='j\_username'>OpenID Identity</label>

<input type='text' class='text\_' name='j\_username' />

</p>

<p>

<input type='submit' value='Login' />

</p>

</form>

</div>

</div>

<script type='text/javascript'>

(function(){

document.forms['loginForm'].elements['j\_username'].focus();

})();

</script>

</body>

<%@ page contentType="text/html;charset=UTF-8" %>

<!doctype html>

<html>

<head>

<meta name="layout" content="basic">

<title>Создание новой темы</title>

</head>

<body>

<g:render template="/basic/navs/comms"/>

<article>

<div class="title">

<h1>Создание новой темы</h1>

</div>

<div class="wrapper">

<div class="main\_block">

<g:form action="savePost" params="${[community\_id: community\_id]}">

<dl>

<dt>Заголовок\*</dt>

<dt><g:textArea cols="25" rows="1" name="name" value="${post?.name}"/></dt>

<g:hasErrors bean="${post}" field="name">

<g:eachError bean="${post}" field="name">

<p style="color: red;">

<g:message error="${it}"/>

</p>

</g:eachError>

</g:hasErrors>

<dt>Сообщение\*</dt>

<dt><g:textArea cols="25" rows="10" name="content"/></dt>

<br>

<dt><g:submitButton name="save" value="Создать"/></dt>

</dl>

</g:form>

</div>

</div>

<div class="left\_menu">

<ul>

<li><a href="">Моя страница</a></li>

<li><a href="">Мои сообщества</a></li>

<li><a href="${createLink(controller: "basic", action: "logout")}">Выйти</a></li>

</ul>

</div>

<div class="clear"></div>

</article>

</body>

</html>

<%@ page import="soc.Community" %>

<!doctype html>

<html>

<head>

<meta name="layout" content="basic">

<title>${community.name}</title>

</head>

<body>

<g:render template="/basic/navs/stable"/>

<article>

<div class="title">

<h1><a style="color: #ffffff;"

href="${createLink(controller: "community", action: "showComm", id: "${community.shortName}")}">

${community.name}</a> - ${post.name}

</h1>

</div>

<div class="wrapper">

<div class="main\_block">

<h2><a href="">${post?.author?.user?.profile?.surname} ${post?.author?.user?.profile?.name} ${post?.author?.user?.profile?.middle\_name}</a>

</h2>

<span class="date">${post.lastUpdated.dateString} / ${post.lastUpdated.timeString}</span><br><br>

<p>${firstComment.content}</p>

<br><b>Коментарии к обсуждению:</b>

<g:each var="comment" in="${comments}">

<div class="tweets">

<a href="">${comment?.master?.profile?.surname} ${comment?.master?.profile?.name} ${comment?.master?.profile?.middle\_name}</a>:<br><br>

${comment.content}

<br><br><span

class="date">${comment.lastUpdated.dateString} / ${comment.lastUpdated.timeString}</span>

</div>

</g:each>

<g:if test="${flash.error\_message}">

<div class="message"

style="color: red; padding-top: 10px; padding-bottom: 10px;">${flash.error\_message}</div>

</g:if>

<g:form action="newComment" id="${post.id}">

<g:textArea id="coment" name="coment" style="width:100%;height:50px;"

cols="25" rows="10" onfocus="this.value = '';">Оставить коментарий...</g:textArea><br>

<g:submitButton name="save" value="Отправить коментарий"/>

</g:form>

%{-- <div id="filteredList">

<g:render template="listPosts"/>

</div>--}%

</div>

</div>

<g:render template="/basic/leftMenu"/>

<g:render template="/community/rightBlock"

model="${[rectors: rectors, teachers: teachers, students: students, users: users]}"/>

<div class="clear"></div>

</article>

</body>

</html>

<head>

<meta name="layout" content="main" />

<title>Edit Profile</title>

</head>

<body>

<div class="nav">

<span class="menuButton"><a class='home' href="${createLinkTo(dir:'')}">Home</a></span>

</div>

<div class="body">

<h1>Edit Profile</h1>

<g:if test="${flash.message}">

<div class="message">${flash.message}</div>

</g:if>

<g:hasErrors bean="${person}">

<div class="errors">

<g:renderErrors bean="${person}" as="list" />

</div>

</g:hasErrors>

<g:form>

<input type="hidden" name="id" value="${person.id}" />

<input type="hidden" name="version" value="${person.version}" />

<div class="dialog">

<table>

<tbody>

<tr class='prop'>

<td valign='top' class='name'><label for='username'>Login Name:</label></td>

<td valign='top' class='value ${hasErrors(bean:person,field:'username','errors')}'>

<input type="hidden" name='username' value="${person.username?.encodeAsHTML()}"/>

<div style="margin:3px">${person.username?.encodeAsHTML()}</div>

</td>

</tr>

<tr class='prop'>

<td valign='top' class='name'><label for='userRealName'>Full Name:</label></td>

<td valign='top' class='value ${hasErrors(bean:person,field:'userRealName','errors')}'>

<input type="text" name='userRealName' value="${person.userRealName?.encodeAsHTML()}"/>

</td>

</tr>

<tr class='prop'>

<td valign='top' class='name'><label for='passwd'>Password:</label></td>

<td valign='top' class='value ${hasErrors(bean:person,field:'passwd','errors')}'>

<input type="password" name='passwd' value=""/>

</td>

</tr>

<tr class='prop'>

<td valign='top' class='name'><label for='enabled'>Confirm Password:</label></td>

<td valign='top' class='value ${hasErrors(bean:person,field:'passwd','errors')}'>

<input type="password" name='repasswd' />

</td>

</tr>

<tr class='prop'>

<td valign='top' class='name'><label for='email'>Email:</label></td>

<td valign='top' class='value ${hasErrors(bean:person,field:'email','errors')}'>

<input type="text" name='email' value="${person.email?.encodeAsHTML()}"/>

</td>

</tr>

<tr class='prop'>

<td valign='top' class='name'><label for='emailShow'>Show Email:</label></td>

<td valign='top' class='value ${hasErrors(bean:person,field:'emailShow','errors')}'>

<g:checkBox name='emailShow' value="${person.emailShow}" ></g:checkBox>

</td>

</tr>

</tbody>

</table>

</div>

<div class="buttons">

<span class="button"><g:actionSubmit class='save' value="Update" /></span>

</div>

</g:form>

</div>

</body>

<html>

<head>

<title>Регистрация нового пользователя</title>

<meta name="layout" content="basic">

<style>

dd {

text-align: left;

margin-left: 80px;

margin-top: 5px;

}

</style>

</head>

<body>

<g:render template="/basic/navs/stable"/>

<article>

<div class="about">

<h1>Регистрация нового пользователя</h1>

<g:hasErrors>

<div class="errors" style="padding-left: 5%;">

<g:renderErrors bean="${person}" as="list"/>

</div>

</g:hasErrors>

<g:if test="${flash.message}">

<div class="message" style="color: red; padding-top: 10px; padding-bottom: 10px;">${flash.message}</div>

</g:if>

<g:form action="save">

<dl>

<dt>Логин</dt>

<dt><input id='login' type="text" name='login' value="${person?.login?.encodeAsHTML()}"/></dt>

<dt>Пароль</dt>

<dt><input id='password' type="password" name='password' value="${person?.password?.encodeAsHTML()}"/>

</dt>

<dt>Ещё раз</dt>

<dt><input id='enabled' type="password" name='repassword' value="${person?.password?.encodeAsHTML()}"/>

</dt>

<dt>Почтовый ящик</dt>

<dt><input id='email' type="text" name='email' value="${person?.email?.encodeAsHTML()}"/></dt>

<dt>Введите код</dt>

<dt><input id='code' type="text" name="captcha" size="8"/>

<img src="${createLink(controller: 'captcha', action: 'index')}" align="absmiddle"/>

</dt>

<br>

<dt><g:submitButton name="register" class='save' type="submit" value="Зарегистироваться"/></dt>

</dl>

</g:form>

</div>

</article>

</body>

</html>

<head>

<meta name="layout" content="main" />

<title>User Profile</title>

</head>

<body>

<div class="nav">

<span class="menuButton"><a class='home' href="${createLinkTo(dir:'')}">Home</a></span>

</div>

<div class="body">

<h1>User Profile</h1>

<g:if test="${flash.message}">

<div class="message">${flash.message}</div>

</g:if>

<div class="dialog">

<table>

<tbody>

<tr class="prop">

<td valign="top" class="name">Login Name:</td>

<td valign="top" class="value">${person.username?.encodeAsHTML()}</td>

</tr>

<tr class="prop">

<td valign="top" class="name">Full Name:</td>

<td valign="top" class="value">${person.userRealName?.encodeAsHTML()}</td>

</tr>

<tr class="prop">

<td valign="top" class="name">Enabled:</td>

<td valign="top" class="value">${person.enabled}</td>

</tr>

<tr class="prop">

<td valign="top" class="name">Email:</td>

<td valign="top" class="value">${person.email?.encodeAsHTML()}</td>

</tr>

<tr class="prop">

<td valign="top" class="name">Show Email:</td>

<td valign="top" class="value">${person.emailShow}</td>

</tr>

<tr class="prop">

<td valign="top" class="name">Roles:</td>

<td valign="top" class="value">

<ul>

<g:each var='authority' in="${person.authorities}">

<li>${authority.authority}</li>

</g:each>

</ul>

</td>

</tr>

</tbody>

</table>

</div>

<div class="buttons">

<g:form>

<input type="hidden" name="id" value="${person.id}" />

<span class="button"><g:actionSubmit class='edit' value="Edit" /></span>

</g:form>

</div>

</div>

</body>

<head>

<meta name="layout" content="main" />

<title>Create soc.Requestmap</title>

</head>

<body>

<div class="nav">

<span class="menuButton"><a class="home" href="${createLinkTo(dir:'')}">Home</a></span>

<span class="menuButton"><g:link class="list" action="list">soc.Requestmap List</g:link></span>

</div>

<div class="body">

<h1>Create soc.Requestmap</h1>

<g:if test="${flash.message}">

<div class="message">${flash.message}</div>

</g:if>

<g:hasErrors bean="${requestmap}">

<div class="errors">

<g:renderErrors bean="${requestmap}" as="list" />

</div>

</g:hasErrors>

<g:form action="save">

<div class="dialog">

<table>

<tbody>

<tr class="prop">

<td valign="top" class="name"><label for="url">URL Pattern:</label></td>

<td valign="top" class="value ${hasErrors(bean:requestmap,field:'url','errors')}">

<input type="text" id="url" name="url" value="${requestmap.url?.encodeAsHTML()}"/>

</td>

</tr>

<tr class="prop">

<td valign="top" class="name"><label for="configAttribute">soc.Role (comma-delimited):</label></td>

<td valign="top" class="value ${hasErrors(bean:requestmap,field:'configAttribute','errors')}">

<input type="text" id="configAttribute" name="configAttribute" value="${requestmap.configAttribute?.encodeAsHTML()}"/>

</td>

</tr>

</tbody>

</table>

</div>

<div class="buttons">

<span class="button"><input class="save" type="submit" value="Create" /></span>

</div>

</g:form>

</div>

</body>

<head>

<meta name="layout" content="main" />

<title>Edit soc.Requestmap</title>

</head>

<body>

<div class="nav">

<span class="menuButton"><a class="home" href="${createLinkTo(dir:'')}">Home</a></span>

<span class="menuButton"><g:link class="list" action="list">soc.Requestmap List</g:link></span>

<span class="menuButton"><g:link class="create" action="create">New soc.Requestmap</g:link></span>

</div>

<div class="body">

<h1>Edit soc.Requestmap</h1>

<g:if test="${flash.message}">

<div class="message">${flash.message}</div>

</g:if>

<g:hasErrors bean="${requestmap}">

<div class="errors">

<g:renderErrors bean="${requestmap}" as="list" />

</div>

</g:hasErrors>

<div class="prop">

<span class="name">ID:</span>

<span class="value">${requestmap.id}</span>

</div>

<g:form>

<input type="hidden" name="id" value="${requestmap.id}" />

<input type="hidden" name="version" value="${requestmap.version}" />

<div class="dialog">

<table>

<tbody>

<tr class="prop">

<td valign="top" class="name"><label for="url">URL Pattern:</label></td>

<td valign="top" class="value ${hasErrors(bean:requestmap,field:'url','errors')}">

<input type="text" id="url" name="url" value="${requestmap.url?.encodeAsHTML()}"/>

</td>

</tr>

<tr class="prop">

<td valign="top" class="name"><label for="configAttribute">Roles (comma-delimited):</label></td>

<td valign="top" class="value ${hasErrors(bean:requestmap,field:'configAttribute','errors')}">

<input type="text" name='configAttribute' value="${requestmap.configAttribute}"/>

</td>

</tr>

</tbody>

</table>

</div>

<div class="buttons">

<span class="button"><g:actionSubmit class="save" value="Update" /></span>

<span class="button"><g:actionSubmit class="delete" onclick="return confirm('Are you sure?');" value="Delete" /></span>

</div>

</g:form>

</div>

</body>

<%@ page import="soc.Requestmap" %>

<head>

<meta name="layout" content="main" />

<title>soc.Requestmap List</title>

</head>

<body>

<div class="nav">

<span class="menuButton"><a class="home" href="${createLinkTo(dir:'')}">Home</a></span>

<span class="menuButton"><g:link class="create" action="create">New soc.Requestmap</g:link></span>

</div>

<div class="body">

<h1>soc.Requestmap List</h1>

<g:if test="${flash.message}">

<div class="message">${flash.message}</div>

</g:if>

<div class="list">

<table>

<thead>

<tr>

<g:sortableColumn property="id" title="ID" />

<g:sortableColumn property="url" title="URL Pattern" />

<g:sortableColumn property="configAttribute" title="Roles" />

<th>&nbsp;</th>

</tr>

</thead>

<tbody>

<g:each in="${requestmapList}" status="i" var="requestmap">

<tr class="${(i % 2) == 0 ? 'odd' : 'even'}">

<td>${requestmap.id}</td>

<td>${requestmap.url?.encodeAsHTML()}</td>

<td>${requestmap.configAttribute}</td>

<td class="actionButtons">

<span class="actionButton">

<g:link action="show" id="${requestmap.id}">Show</g:link>

</span>

</td>

</tr>

</g:each>

</tbody>

</table>

</div>

<div class="paginateButtons">

<g:paginate total="${Requestmap.count()}" />

</div>

</div>

</body>

<head>

<meta name="layout" content="main" />

<title>Show soc.Requestmap</title>

</head>

<body>

<div class="nav">

<span class="menuButton"><a class="home" href="${createLinkTo(dir:'')}">Home</a></span>

<span class="menuButton"><g:link class="list" action="list">soc.Requestmap List</g:link></span>

<span class="menuButton"><g:link class="create" action="create">New soc.Requestmap</g:link></span>

</div>

<div class="body">

<h1>Show soc.Requestmap</h1>

<g:if test="${flash.message}">

<div class="message">${flash.message}</div>

</g:if>

<div class="dialog">

<table>

<tbody>

<tr class="prop">

<td valign="top" class="name">ID:</td>

<td valign="top" class="value">${requestmap.id}</td>

</tr>

<tr class="prop">

<td valign="top" class="name">URL:</td>

<td valign="top" class="value">${requestmap.url}</td>

</tr>

<tr class="prop">

<td valign="top" class="name">Roles:</td>

<td valign="top" class="value">${requestmap.configAttribute}</td>

</tr>

</tbody>

</table>

</div>

<div class="buttons">

<g:form>

<input type="hidden" name="id" value="${requestmap.id}" />

<span class="button"><g:actionSubmit class="edit" value="Edit" /></span>

<span class="button"><g:actionSubmit class="delete" onclick="return confirm('Are you sure?');" value="Delete" /></span>

</g:form>

</div>

</div>

</body>

<head>

<meta name="layout" content="old\_main" />

<title>Create Role</title>

</head>

<body>

<div class="nav">

<span class="menuButton"><a class="home" href="${createLinkTo(dir:'/secret')}">Home</a></span>

<span class="menuButton"><g:link class="list" action="list">Role List</g:link></span>

</div>

<div class="body">

<h1>Create Role</h1>

<g:if test="${flash.message}">

<div class="message">${flash.message}</div>

</g:if>

<g:hasErrors bean="${authority}">

<div class="errors">

<g:renderErrors bean="${authority}" as="list" />

</div>

</g:hasErrors>

<g:form action="save">

<div class="dialog">

<table>

<tbody>

<tr class="prop">

<td valign="top" class="name"><label for="authority">soc.Role Name:</label></td>

<td valign="top" class="value ${hasErrors(bean:authority,field:'authority','errors')}">

<input type="text" id="authority" name="authority" value="${authority?.authority?.encodeAsHTML()}"/>

</td>

</tr>

<tr class="prop">

<td valign="top" class="name"><label for="description">Description:</label></td>

<td valign="top" class="value ${hasErrors(bean:authority,field:'description','errors')}">

<input type="text" id="description" name="description" value="${authority?.description?.encodeAsHTML()}"/>

</td>

</tr>

</tbody>

</table>

</div>

<div class="buttons">

<span class="button"><input class="save" type="submit" value="Create" /></span>

</div>

</g:form>

</div>

</body>

<head>

<meta name="layout" content="old\_main" />

<title>Edit Role</title>

</head>

<body>

<div class="nav">

<span class="menuButton"><a class="home" href="${createLinkTo(dir:'/secret')}">Home</a></span>

<span class="menuButton"><g:link class="list" action="list">Role List</g:link></span>

<span class="menuButton"><g:link class="create" action="create">New Role</g:link></span>

</div>

<div class="body">

<h1>Edit Role</h1>

<g:if test="${flash.message}">

<div class="message">${flash.message}</div>

</g:if>

<g:hasErrors bean="${authority}">

<div class="errors">

<g:renderErrors bean="${authority}" as="list" />

</div>

</g:hasErrors>

<div class="prop">

<span class="name">ID:</span>

<span class="value">${authority.id}</span>

</div>

<g:form>

<input type="hidden" name="id" value="${authority.id}" />

<input type="hidden" name="version" value="${authority.version}" />

<div class="dialog">

<table>

<tbody>

<tr class="prop">

<td valign="top" class="name"><label for="authority">soc.Role Name:</label></td>

<td valign="top" class="value ${hasErrors(bean:authority,field:'authority','errors')}">

<input type="text" id="authority" name="authority" value="${authority.authority?.encodeAsHTML()}"/>

</td>

</tr>

<tr class="prop">

<td valign="top" class="name"><label for="description">Description:</label></td>

<td valign="top" class="value ${hasErrors(bean:authority,field:'description','errors')}">

<input type="text" id="description" name="description" value="${authority.description?.encodeAsHTML()}"/>

</td>

</tr>

<tr class="prop">

<td valign="top" class="name"><label for="people">People:</label></td>

<td valign="top" class="value ${hasErrors(bean:authority,field:'people','errors')}">

<ul>

<g:each var="p" in="${authority.people?}">

<li>${p}</li>

</g:each>

</ul>

</td>

</tr>

</tbody>

</table>

</div>

<div class="buttons">

<span class="button"><g:actionSubmit class="save" value="Update" /></span>

<span class="button"><g:actionSubmit class="delete" onclick="return confirm('Are you sure?');" value="Delete" /></span>

</div>

</g:form>

</div>

</body>

<%@ page import="soc.Role" %>

<head>

<meta name="layout" content="old\_main" />

<title>Role List</title>

</head>

<body>

<div class="nav">

<span class="menuButton"><a class="home" href="${createLinkTo(dir:'/secret')}">Home</a></span>

<span class="menuButton"><g:link class="create" action="create">New Role</g:link></span>

</div>

<div class="body">

<h1>Role List</h1>

<g:if test="${flash.message}">

<div class="message">${flash.message}</div>

</g:if>

<div class="list">

<table>

<thead>

<tr>

<g:sortableColumn property="id" title="ID" />

<g:sortableColumn property="authority" title="soc.Role Name" />

<g:sortableColumn property="description" title="Description" />

<th>&nbsp;</th>

</tr>

</thead>

<tbody>

<g:each in="${authorityList}" status="i" var="authority">

<tr class="${(i % 2) == 0 ? 'odd' : 'even'}">

<td>${authority.id}</td>

<td>${authority.authority?.encodeAsHTML()}</td>

<td>${authority.description?.encodeAsHTML()}</td>

<td class="actionButtons">

<span class="actionButton">

<g:link action="show" id="${authority.id}">Show</g:link>

</span>

</td>

</tr>

</g:each>

</tbody>

</table>

</div>

<div class="paginateButtons">

<g:paginate total="${Role.count()}" />

</div>

</div>

</body>

<head>

<meta name="layout" content="old\_main"/>

<title>Show Role</title>

</head>

<body>

<div class="nav">

<span class="menuButton"><a class="home" href="${createLinkTo(dir: '/secret')}">Home</a></span>

<span class="menuButton"><g:link class="list" action="list">Role List</g:link></span>

<span class="menuButton"><g:link class="create" action="create">New Role</g:link></span>

</div>

<div class="body">

<h1>Show Role</h1>

<g:if test="${flash.message}">

<div class="message">${flash.message}</div>

</g:if>

<div class="dialog">

<table>

<tbody>

<tr class="prop">

<td valign="top" class="name">ID:</td>

<td valign="top" class="value">${authority.id}</td>

</tr>

<tr class="prop">

<td valign="top" class="name">soc.Role Name:</td>

<td valign="top" class="value">${authority.authority}</td>

</tr>

<tr class="prop">

<td valign="top" class="name">Description:</td>

<td valign="top" class="value">${authority.description}</td>

</tr>

<tr class="prop">

<td valign="top" class="name">People:</td>

<td valign="top" class="value">${authority.people}</td>

</tr>

</tbody>

</table>

</div>

<div class="buttons">

<g:form>

<input type="hidden" name="id" value="${authority?.id}"/>

<span class="button"><g:actionSubmit class="edit" value="Edit"/></span>

<span class="button"><g:actionSubmit class="delete" onclick="return confirm('Are you sure?');"

value="Delete"/></span>

</g:form>

</div>

</div>

</body>

<!doctype html>

<html>

<head>

<meta name="layout" content="basic">

<title>Администраторская</title>

</head>

<body>

<article>

<div class="title">

<h1>Ресурсы для управления</h1>

</div>

<div class="wrapper">

<div class="main\_block">

<h2><g:link controller="user" action="list">Пользователи</g:link></h2><br>

<h2><g:link controller="role" action="list">Роли</g:link></h2><br>

<h2><g:link controller="user" action="changeRole">Сменить роль</g:link></h2><br>

<h2><g:link controller="community" action="list">Сообщества</g:link></h2><br>

</div>

</div>

<div class="clear"></div>

</article>

</body>

</html>

<!doctype html>

<html>

<head>

<title>Grails Runtime Exception</title>

<meta name="layout" content="main">

<link rel="stylesheet" href="${resource(dir: 'css', file: 'errors.css')}" type="text/css">

</head>

<body>

<g:renderException exception="${exception}" />

</body>

</html>

<!doctype html>

<html>

<head>

<meta name="layout" content="main"/>

</head>

<%@ page contentType="text/html;charset=UTF-8" %>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Успешная регистрация!</title>

<link rel="stylesheet" href="${resource(dir: 'css', file: 'style.css')}" type="text/css">

<g:javascript library='application'/>

<g:javascript library='scriptaculous'/>

</head>

<body>

<div id="success">

<h1>${flash.message}</h1><br>

<a href="<g:createLink controller="basic" action="index"/>">На главную страницу</a>

</div>

</body>

</html>

**Службы**

import javax.mail.MessagingException

import org.springframework.mail.MailException

import org.springframework.mail.SimpleMailMessage

/\*\*

\* Simple service for sending emails.

\*

\* Work is planned in the Grails roadmap to implement first-class email

\* support, so there's no point in making this code any more sophisticated.

\*

\* @author Haotian Sun

\*/

class EmailerService {

boolean transactional = false

def mailSender

def mailMessage // a "prototype" email instance

/\*\*

\* Send a allComm of emails.

\*

\* @param mails a allComm of maps

\*/

def sendEmails(mails) {

// Build the mail messages

def messages = []

for (mail in mails) {

// newComm a copy of the default message

def message = new SimpleMailMessage(mailMessage)

message.to = mail.to

message.text = mail.text

message.subject = mail.subject

messages << message

}

// Send them all together

try {

mailSender.send(messages as SimpleMailMessage[])

}

catch (MailException e) {

log.error "Failed to send emails: $e.message", e

}

catch (MessagingException e) {

log.error "Failed to send emails: $e.message", e

}

}

}

package soc

class CommunityService {

private long communityID

public void setCommunity(def num) {

communityID = num as long

}

public Community getCommunity() {

return Community.get(communityID)

}

}

package soc

class QuoteService {

boolean transactional = false

def getRandomQuote() {

null

/\*

def allQuotes = Quote.list()

def randomQuote = null

if (allQuotes.size() > 0) {

def randomIdx = new Random().nextInt(allQuotes.size())

randomQuote = allQuotes[randomIdx]

} else {

randomQuote = getStaticQuote()

}

return randomQuote

\*/

}

def getStaticQuote() {

/\*

return new Quote(author: "Anonymous",

content: "Real Programmers Don’t eat quiche")

\*/

null

}

def serviceMethod() {}

}

package soc

class RolesService {

def authenticateService

private static final String ROLE\_ADMIN = "ROLE\_ADMIN"

private static final String ROLE\_RECTOR = "ROLE\_RECTOR"

private static final String ROLE\_TEACHER = "ROLE\_TEACHER"

private static final String ROLE\_STUDENT = "ROLE\_STUDENT"

private static final String ROLE\_USER = "ROLE\_USER"

private static final String ADMIN\_NAME = "Администратор"

private static final String RECTOR\_NAME = "Управляющий"

private static final String TEACHER\_NAME = "Преподаватель"

private static final String STUDENT\_NAME = "Студент"

private static final String USER\_NAME = "Пользователь"

String getAdminName() {

return ADMIN\_NAME

}

String getRectorName() {

return RECTOR\_NAME

}

String getTeacherName() {

return TEACHER\_NAME

}

String getStudentName() {

return STUDENT\_NAME

}

String getUserName() {

return USER\_NAME

}

String getAdmin() {

return ROLE\_ADMIN

}

String getRector() {

return ROLE\_RECTOR

}

String getTeacher() {

return ROLE\_TEACHER

}

String getStudent() {

return ROLE\_STUDENT

}

String getUser() {

return ROLE\_USER

}

boolean isMember(User user) { // исправить

if (!user) return false

User findUser = Member.findByUser(user) as User

return findUser as boolean

}

Role getRole(String name) {

return Role.findByAuthority(name)

}

}

package soc

import java.security.SecureRandom

class SecureService {

private char[] dictionary = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'j', 'k', 'l', 'q', 'w', 'e', 'r', 't', 'y', 'u', 'i', 'o', 'p',

'z', 'x', 'c', 'v', 'b', 'n', 'm', '1', '2', '3', '4', '5', '6', '7', '8', '9', '0', '-', '=', '+']

String generateKey(int size) {

int length = dictionary.length

String key = ""

for (int i = 0; i < size; i++) {

int index = new SecureRandom().nextInt(length)

char newSymbol = dictionary[index]

boolean upCase = new SecureRandom().nextBoolean()

if (upCase) newSymbol = newSymbol.toUpperCase()

key += newSymbol

}

return key

}

ArrayList<String> getGeneratedKeys(int count, int keyLength) {

ArrayList<String> keys = new ArrayList<String>();

while (count != 0) {

String currentKey = generateKey(keyLength);

if (keys.indexOf(currentKey) == -1) {

keys.add(currentKey);

count--;

}

}

return keys;

}

}

package soc

class UserService {

def deleteSecretFields(User user) {

if (!user) return null

user.password = ""

return user

}

}

**Конфигурации**

import soc.\*

class BootStrap {

def authenticateService

def rolesService

def init = { servletContext ->

createAdminUserIfRequired()

}

def createAdminUserIfRequired() {

if (!User.findByLogin("admin")) {

def userRole = Role.findByAuthority(rolesService.user) ?: new Role(authority: rolesService.user, name: rolesService.userName).save() // failOnError: true

def studentRole = Role.findByAuthority(rolesService.student) ?: new Role(authority: rolesService.student, name: rolesService.studentName).save()

def teacherRole = Role.findByAuthority(rolesService.teacher) ?: new Role(authority: rolesService.teacher, name: rolesService.teacherName).save()

def rectorRole = Role.findByAuthority(rolesService.rector) ?: new Role(authority: rolesService.rector, name: rolesService.rectorName).save()

def adminRole = Role.findByAuthority(rolesService.admin) ?: new Role(authority: rolesService.admin, name: rolesService.adminName).save()

println "Fresh Database. Creating ADMIN user."

String pass = authenticateService.passwordEncoder("password")

UserRating \_rating = new UserRating(value: 1000000).save()

UserProfile prof = new UserProfile(aboutMe: "Smart admin", birthday: new Date(3000, 10, 23),

name: "Fox", surname: "Smart", middle\_name: "Server", rating: \_rating,

work: "Full Master", state: "I will kill you!").save()

UserContact \_contacts = new UserContact(content: "public@miem.edu.ru").save()

User superAdmin = new User(login: "admin", password: pass, email: "gotowhitesnow@gmail.com",

profile: prof, contacts: \_contacts).save()

Role role = Role.findByAuthority(rolesService.admin)

// now add the User to the role

role.addToPeople(superAdmin)

// Записываем всё в базу

role.save()

addDepartments()

} else {

println "Existing admin user, skipping creation"

}

}

def addDepartments() {

Contact contact = new Contact(info: "Москва, ул. Пионерская Малая, 12\nМосква, Большой Трехсвятительский пер., 3").save()

TrainingDepartment department = new TrainingDepartment(shortName: "Миэм", contacts: contact,

name: "Московский государсвтенный университет электроники и математики", c\_describe: "Чудесный технический университет").save()

}

def destroy = {

}

}

class UrlMappings {

static mappings = {

"/$controller/$action?/$id?" {

constraints {

}

}

"/comm/edit/$id?"(controller: "community", action: "edit")

"/comm/show/$id?"(controller: "community", action: "show")

"/comm/delete/$id?"(controller: "community", action: "delete")

"/comm/list"(controller: "community", action: "list")

"/communities"(controller: "community", action: "allComm")

"/community/create\_community"(controller: "community", action: "newComm")

"/community/save\_new"(controller: "community", action: "saveComm")

"/community/comm$id?"(controller: "community", action: "showComm")

"/"(controller: "basic")

"/institution"(controller: "basic", action: "aboutInstitute")

"/department"(controller: "basic", action: "aboutCathedra")

"/contacts"(controller: "basic", action: "contacts")

"/secret"(controller: "secret")

"/success"(view: "success")

"500"(view: '/error')

}

}

package soc

class SecurityFilters {

def authenticateService

def rolesService

def filters = {

commChanges(controller: "community", action: "edit") {

before = {

}

}

enterSecret(controller: "secret", action: "\*") {

before = {

User user = authenticateService.userDomain() as User

}

}

}

}

modules = {

application {

resource url:'js/application.js'

}

}

grails.servlet.version = "2.5" // Change depending on target container compliance (2.5 or 3.0)

grails.project.class.dir = "target/classes"

grails.project.test.class.dir = "target/test-classes"

grails.project.test.reports.dir = "target/test-reports"

grails.project.target.level = 1.6

grails.project.source.level = 1.6

//grails.project.war.file = "target/${appName}-${appVersion}.war"

grails.server.port.http = 9090

grails.project.dependency.resolution = {

// inherit Grails' default dependencies

inherits("global") {

// uncomment to disable ehcache

// excludes 'ehcache'

}

log "error" // log level of Ivy resolver, either 'error', 'warn', 'info', 'debug' or 'verbose'

checksums true // Whether to verify checksums on resolve

repositories {

inherits true // Whether to inherit repository definitions from plugins

grailsPlugins()

grailsHome()

grailsCentral()

mavenCentral()

// uncomment these to enable remote dependency resolution from public Maven repositories

//mavenCentral()

//mavenLocal()

//mavenRepo "http://snapshots.repository.codehaus.org"

//mavenRepo "http://repository.codehaus.org"

//mavenRepo "http://download.java.net/maven/2/"

//mavenRepo "http://repository.jboss.com/maven2/"

}

dependencies {

// specify dependencies here under either 'build', 'compile', 'runtime', 'test' or 'provided' scopes eg.

runtime 'mysql:mysql-connector-java:5.1.16'

}

plugins {

runtime ":hibernate:$grailsVersion"

runtime ":jquery:1.7.1"

runtime ":resources:1.1.6"

// Uncomment these (or add new ones) to enable additional resources capabilities

//runtime ":zipped-resources:1.0"

//runtime ":cached-resources:1.0"

//runtime ":yui-minify-resources:0.1.4"

build ":tomcat:$grailsVersion"

}

}

// locations to search for config files that get merged into the main config

// config files can either be Java properties files or ConfigSlurper scripts

// grails.config.locations = [ "classpath:${appName}-config.properties",

// "classpath:${appName}-config.groovy",

// "file:${userHome}/.grails/${appName}-config.properties",

// "file:${userHome}/.grails/${appName}-config.groovy"]

// if (System.properties["${appName}.config.location"]) {

// grails.config.locations << "file:" + System.properties["${appName}.config.location"]

// }

grails.project.groupId = appName // change this to alter the default package name and Maven publishing destination

grails.mime.file.extensions = true // enables the parsing of file extensions from URLs into the request format

grails.mime.use.accept.header = false

grails.mime.types = [ html: ['text/html','application/xhtml+xml'],

xml: ['text/xml', 'application/xml'],

text: 'text/plain',

js: 'text/javascript',

rss: 'application/rss+xml',

atom: 'application/atom+xml',

css: 'text/css',

csv: 'text/csv',

all: '\*/\*',

json: ['application/json','text/json'],

form: 'application/x-www-form-urlencoded',

multipartForm: 'multipart/form-data'

]

// URL Mapping Cache Max Size, defaults to 5000

//grails.urlmapping.cache.maxsize = 1000

// What URL patterns should be processed by the resources plugin

grails.resources.adhoc.patterns = ['/images/\*', '/css/\*', '/js/\*', '/plugins/\*']

// The default codec used to encode data with ${}

grails.views.default.codec = "html" // none, html, base64

grails.views.gsp.encoding = "UTF-8"

grails.converters.encoding = "UTF-8"

// enable Sitemesh preprocessing of GSP pages

grails.views.gsp.sitemesh.preprocess = true

// scaffolding templates configuration

grails.scaffolding.templates.domainSuffix = 'Instance'

//enable jquery

//grails.views.javascript.library="jquery"

//grails.views.javascript.library="prototype"

// Set to false to use the new Grails 1.2 JSONBuilder in the render method

grails.json.legacy.builder = false

// enabled native2ascii conversion of i18n properties files

grails.enable.native2ascii = true

// packages to include in Spring bean scanning

grails.spring.bean.packages = []

// whether to disable processing of multi part requests

grails.web.disable.multipart=false

// request parameters to mask when logging exceptions

grails.exceptionresolver.params.exclude = ['password']

// enable query caching by default

grails.hibernate.cache.queries = true

// set per-environment serverURL stem for creating absolute links

environments {

development {

grails.logging.jul.usebridge = true

}

production {

grails.logging.jul.usebridge = false

// TODO: grails.serverURL = "http://www.changeme.com"

}

}

// log4j configuration

log4j = {

// Example of changing the log pattern for the default console

// appender:

//

//appenders {

// console name:'stdout', layout:pattern(conversionPattern: '%c{2} %m%n')

//}

error 'org.codehaus.groovy.grails.web.servlet', // controllers

'org.codehaus.groovy.grails.web.pages', // GSP

'org.codehaus.groovy.grails.web.sitemesh', // layouts

'org.codehaus.groovy.grails.web.mapping.filter', // URL mapping

'org.codehaus.groovy.grails.web.mapping', // URL mapping

'org.codehaus.groovy.grails.commons', // core / classloading

'org.codehaus.groovy.grails.plugins', // plugins

'org.codehaus.groovy.grails.orm.hibernate', // hibernate integration

'org.springframework',

'org.hibernate',

'net.sf.ehcache.hibernate'

}

security.defaultRole = "ROLE\_USER"

//log4j.logger.org.springframework.security='off,stdout'

dataSource {

pooled = true

driverClassName = "com.mysql.jdbc.Driver"

username = "root"

password = "1807"

}

hibernate {

cache.use\_second\_level\_cache = true

cache.use\_query\_cache = true

cache.provider\_class = "net.sf.ehcache.hibernate.EhCacheProvider"

}

/\*log4j = {

// ... whatever

debug 'org.hibernate.SQL',

'org.hibernate.transaction' // optionally

}\*/

// environment specific settings

environments {

development {

dataSource {

dbCreate = "update" // one of 'newComm', 'newComm-drop','update'

url = "jdbc:mysql://localhost:3306/base"

}

}

test {

dataSource {

dbCreate = "update"

url = "jdbc:mysql://localhost:3306/base"

}

}

production {

dataSource {

dbCreate = "update"

url = "jdbc:mysql://localhost:3306/base"

}

}

}

security {

// влючает безопасность (очень важный параметр)

active = true

useRequestMapDomainClass = true

loginUserDomainClass = "soc.User"

authorityDomainClass = "soc.Role"

requestMapClass = "soc.Requestmap"

userName = "login"

password = "password"

loginFormUrl = "/"

}

package soc

class UserTagLib {

def rolesService

def authenticateService

def fullName = {attrs, body ->

User user

if (attrs.user) {

user = attrs.user

} else {

user = authenticateService.userDomain() as User

}

String result = ""

if (user?.profile) {

result += "${user.profile.surname} ${user.profile.name} ${user.profile.middle\_name}"

}

out << result.encodeAsHTML()

}

def fullStatus = {attrs, body ->

User user

if (attrs.user) {

user = attrs.user

} else {

user = authenticateService.userDomain() as User

}

String result = ""

String role = user.role.name ?: rolesService.userName

result += role

if (user?.profile) {

Department dep = user.profile.department

if (dep) result += ", ${dep.shortName}"

}

out << result.encodeAsHTML()

}

}