Котов Виталий Евгеньевич

**Система учета студентов из числа иностранных граждан**

Дипломная *работа* по специальности

1-31-03-4 «Информатика»

Научный руководитель

Доцент кафедры информатика

Платонов Александр Сергеевич

Дипломная работа допущена к защите

Заведующий кафедрой информатики

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ С. Н. Батан

\_\_ \_\_\_\_\_\_\_\_\_\_\_ 2015

# **РЕФЕРАТ**

Котов Виталий Евгеньевич. Тема дипломной работы: система учета студентов из числа иностранных граждан.

Java, REST-сервисы, MySql, Hibernate, Intellij Idea, TomCat, Maven, AngularJS.

Разработанное приложение позволяет вести учет студентов из числа иностранных граждан и составление статистики.

# **ОГЛАВЛЕНИЕ**

[РЕФЕРАТ 2](#_Toc417556189)

[ОГЛАВЛЕНИЕ 3](#_Toc417556190)

[ВВЕДЕНИЕ 4](#_Toc417556191)

[ГЛАВА 1 ТЕОРИТИЧЕСКАЯ ЧАСТЬ 6](#_Toc417556192)

[1.1 Сведения о MySQL 6](#_Toc417556193)

[1.2 Сведения о REST 6](#_Toc417556194)

[1.3 Сведения о Java 8](#_Toc417556195)

[1.4 Сведения о AngularJS 9](#_Toc417556196)

[ГЛАВА 2 ПРЕДПРОЕКТНОЕ ИССЛЕДОВАНИЕ 10](#_Toc417556197)

[2.1 Анализ предметной области и объекта исследования 10](#_Toc417556198)

[2.2 Постановка задачи 10](#_Toc417556199)

[2.3 Входные и выходные документы 11](#_Toc417556200)

[2.4 Анализ бизнес-процессов 11](#_Toc417556201)

[ГЛАВА 3 ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ 12](#_Toc417556202)

[3.1 Построение инфологической модели 12](#_Toc417556203)

[3.2 Требования к разрабатываемой системе 16](#_Toc417556204)

[3.3 Логическая модель данных 16](#_Toc417556205)

[3.4 Физическое проектирование 17](#_Toc417556206)

[ГЛАВА 4 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ 18](#_Toc417556207)

[4.1 Обоснование выбора ПО для разработки 18](#_Toc417556208)

[4.2 Структура ИС и функционирование каждого блока 18](#_Toc417556209)

[4.3 Структура модели данных 19](#_Toc417556210)

[4.4 Результаты работы ИС 20](#_Toc417556211)

[ЗАКЛЮЧЕНИЕ 23](#_Toc417556212)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 24](#_Toc417556213)

[ПРИЛОЖЕНИЯ 25](#_Toc417556214)

# ВВЕДЕНИЕ

Была поставлена задача разработать приложение для международного отдела университета, позволяющее вести учет студентов из числа иностранных граждан. Функционал приложения должен включать возможно аутентифицированного доступа, составления списка студентов по специальностям, по странам, подсчет статистики, фильтрацию и сортировку данных, экспорт выбранных данных в Microsoft Excel.

Для реализации поставленной задачи была выбрана REST(сокр. от англ. Representational State Transfer — «передача состояния представления») архитектура, которая представляет собой метод взаимодействия компонентов распределённого приложения в сети Интернет, при котором вызов удаленной процедуры представляет собой обычный HTTP-запрос (обычно GET или POST; такой запрос называют REST-запрос), а необходимые данные передаются в качестве параметров запроса. Самое главное достоинство REST сервисов в том, что с ними работать может какая угодно система, будь то сайт, flash, программа и др. так как методы парсинга XML(JSON) и выполнения запросов HTTP присутствуют почти везде.

Для хранения информация была выбрана MySQL, свободная реляционная система управления базами данных. Для реализации REST сервисов использовался кроссплатформенный язык программирования JAVA и IDE (интегрированная среда разработки) IntelliJ IDEA. Доступ к данным был реализован с помощью библиотеки Hibernate, которая не только решает задачу связи классов Java с таблицами базы данных, но и также предоставляет средства для автоматической генерации и обновления набора таблиц, построения запросов и обработки полученных данных и может значительно уменьшить время разработки, которое обычно тратится на ручное написание SQL- и JDBC-кода.

Интерфейс приложения был разработан с использования JavaScript библиотеки AngularJS и стилей Material Design от компании Google. AngularJS предназначен для разработки одностраничных приложений. Его цель — расширение браузерных приложений на основе MVC шаблона, а также упрощение тестирования и разработки. Angular придерживается MVC-шаблона проектирования и осуществляет слабую связь между представлением, данными и логикой компонентов. Используя внедрение зависимости, Angular переносит на клиентскую сторону такие классические серверные службы, как видозависимые контроллеры. Следовательно, уменьшается нагрузка на сервер и веб-приложение становится легче. А также AngularJS очень легко позволят работать с REST API.

**Тема актуальна** и может представлять интерес для широкого круга пользователей. Так как автоматизация процесса учета студентов, составление отчетов значительно экономит время на составление необходимой документации вручную.

Для проектируемой системы предметной областью является отдел университета, который занимается учетом студентов из числа иностранных граждан. Необходимо спроектировать базу данных, информация которой будет использоваться для учета студентов и составления статистики.

**Объектом исследования** является реальный экономический объект: отдел университета, который отвечает за учет студентов.

**Предметом исследования:** является организация процесса учета студентов из числа иностранных граждан, а также расчет статистических данных.

**Цель исследования:** разработать автоматизированную систему для оптимизации учету студентов из числа иностранных граждан. Разрабатываемый прототип будет предоставлять возможность вести автоматизированный учет студентов и расчет статистических данных.

1. ТЕОРИТИЧЕСКАЯ ЧАСТЬ

## Сведения о MySQL

MySQL — свободная реляционная система управления базами данных. 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.

MySQL является решением для малых и средних приложений. Обычно MySQL используется в качестве сервера, к которому обращаются локальные или удалённые клиенты, однако в дистрибутив входит библиотека внутреннего сервера, позволяющая включать MySQL в автономные программы.

MySQL портирована на большое количество платформ: Linux, Mac OS, Solaris, SunOS, Windows. MySQL имеет API для языков Delphi, C, C++, Java, PHP, Python, библиотеки для языков платформы .NET, а также обеспечивает поддержку для ODBC посредством ODBC-драйвера MyODBC. MyODBC работает на всех системах Microsoft Windows и на большинстве платформ Unix.

В реализуемом приложении используется MySQL Server версии 5.6.

## Сведения о REST

REST (сокр. от англ. Representational State Transfer — «передача состояния представления») — метод взаимодействия компонентов распределённого приложения в сети Интернет, при котором вызов удаленной процедуры представляет собой обычный HTTP-запрос (обычно GET или POST), а необходимые данные передаются в качестве параметров запроса. Этот способ является альтернативой более сложным методам, таким как SOAP и CORBA.

В широком смысле REST означает концепцию построения распределённого приложения, при которой компоненты взаимодействуют наподобие взаимодействия клиентов и серверов во Всемирной паутине.

Хотя данная концепция лежит в самой основе Всемирной паутины, термин REST был введён Роем Филдингом, одним из создателей протокола HTTP, лишь в 2000 году. В своей диссертации в Калифорнийском университете в Ирвайне он подвёл теоретическую основу под метод взаимодействия клиентов и серверов во Всемирной паутине, абстрагировав его и назвав «передачей репрезентативного состояния». Филдинг описал концепцию построения распределённого приложения, при которой каждый запрос (REST-запрос) клиента к серверу содержит в себе исчерпывающую информацию о желаемом ответе сервера (желаемом репрезентативном состоянии), и сервер не обязан сохранять информацию о состоянии клиента («клиентской сессии»).

В качестве необходимых условий для построения распределенных REST-приложений Филдинг перечислил следующие:

* Клиент-серверная архитектура.
* Сервер не обязан сохранять информацию о состоянии клиента.
* В каждом запросе клиента должно явно содержаться указание о возможности кэширования ответа и получения ответа из существующего кэша.
* Клиент может взаимодействовать не напрямую с сервером, а с произвольным количеством промежуточных узлов. При этом клиент может не знать о существовании промежуточных узлов, за исключением случаев передачи конфиденциальной информации.
* Унифицированный программный интерфейс сервера. Филдинг приводил URI в качестве примера формата запросов к серверу, а в качестве примера ответа сервера форматы HTML, XML и JSON, различаемые с использованием идентификаторов MIME.

Филдинг указывал, что приложения, не соответствующие приведённым условиям, не могут называться REST-приложениями. Если же все условия соблюдены, то, по его мнению, приложение получит следующие преимущества:

* надёжность (за счет отсутствия необходимости сохранять информацию о состоянии клиента, которая может быть утеряна);
* производительность (за счет использования кэша);
* масштабируемость;
* прозрачность системы взаимодействия, особенно необходимая для приложений обслуживания сети;
* простота интерфейсов;
* портативность компонентов;
* легкость внесения изменений;
* способность эволюционировать, приспосабливаясь к новым требованиям (на примере Всемирной паутины).

## Сведения о Java

Java — объектно-ориентированный язык программирования, разработанный компанией Sun Microsystems (в последующем приобретённой компанией Oracle). Приложения Java обычно транслируются в специальный байт-код, поэтому они могут работать на любой виртуальной Java-машине вне зависимости от компьютерной архитектуры.

Достоинством подобного способа выполнения программ является полная независимость байт-кода от операционной системы и оборудования, что позволяет выполнять Java-приложения на любом устройстве, для которого существует соответствующая виртуальная машина. Другой важной особенностью технологии Java является гибкая система безопасности, в рамках которой исполнение программы полностью контролируется виртуальной машиной. Любые операции, которые превышают установленные полномочия программы (например, попытка несанкционированного доступа к данным или соединения с другим компьютером), вызывают немедленное прерывание.

Технология Java протестирована, усовершенствована, расширена и проверена участниками сообщества разработчиков Java, архитекторов и энтузиастов. Java позволяет разрабатывать высокопроизводительные портативные приложения практически на всех компьютерных платформах.

Java стала незаменимым инструментом для разработчиков и открыла для них следующие возможности:

* написание программного обеспечения на одной платформе и его запуск практически на любой другой платформе
* создание программ, работающих в веб-браузере и имеющих доступ к веб-службам;
* разработка приложений на стороне сервера для форумов в Интернете, магазинов, опросов, обработки форм HTML и много другого;
* объединение приложений или служб с использованием языка Java для создания высокоспециализированных приложений или служб
* создание многофункциональных и эффективных приложений для мобильных телефонов, удаленных процессоров, микроконтроллеров, беспроводных модулей, датчиков, шлюзов, потребительских продуктов и практически любых других категорий электронных устройств;

В реализуемом приложении используется Java 8, релиз версии состоялся 19 марта 2014 года.

## Сведения о AngularJS

AngularJS — JavaScript-фреймворк с открытым исходным кодом. Предназначен для разработки одностраничных приложений. Его цель — расширение браузерных приложений на основе MVC шаблона, а также упрощение тестирования и разработки.

Фреймворк работает с HTML, содержащим дополнительные пользовательские атрибуты, которые описываются директивами, и связывает ввод или вывод области страницы с моделью, представляющей собой обычные переменные JavaScript. Значения этих переменных задаются вручную или извлекаются из статических или динамических JSON-данных.

Angular придерживается MVC-шаблона проектирования и поощряет слабую связь между представлением, данными и логикой компонентов. Используя внедрение зависимости, Angular переносит на клиентскую сторону такие классические серверные службы, как видозависимые контроллеры. Следовательно, уменьшается нагрузка на сервер и веб-приложение становится легче.

1. ПРЕДПРОЕКТНОЕ ИССЛЕДОВАНИЕ

## Анализ предметной области и объекта исследования

Для проектируемой системы предметной областью является процесс учета студентов. В учебном отделе собрана информация о студентах, которые обучаются на различных факультетах, а также в определенных группах, на определенных специальностях. При этом, необходим учет их срока регистрации, места проживания, формы обучения, номера приказа, даты поступления, а также личной информации, информации о медицинской страховке и т. д. Упрощение процесса учета студентов, подсчет их общего количества, а также составление статистики представляет интерес для данного исследования.

База данных студентов создается для экономии времени, которое тратится на составление документации (отчетов) об учащихся студентах. При работе с такой базой, добавлять, изменять информацию в отчете гораздо проще и быстрее.

## Постановка задачи

Ставится задача разработать программный комплекса, который позволит упростить работу отдела университета, отвечающего за учет студентов. Описываемый прототип позволит хранить информацию о факультетах, специальностях, группах, форме обучения, дате поступления, личную информацию, а также сведения о сроке регистрации и места проживания.

Системы должна иметь возможность авторизованного доступа к ресурсам, наделенного различными правами для каждой группы пользователей.

Также функционал прототипа будет включать возможность различных видов подсчетов (общего количества студентов по специальностям, по странам, по общежитиям и т.д.).

## Входные и выходные документы

После проведения необходимых исследований и анализа предметной области, было выявлено, что необходимыми для разработки базы данных являются следующие входные документы:

1. Информация о студентах
2. Информация о факультетах
3. Информация о специальностях
4. Информация о группах
5. Информация о курсах
6. Информация о странах
7. Информация о общежитиях

Выходными документами будут списки студентов по специальностям, по странам, по общежитиям, списки с информацией о регистрации. В которых будет представлена информация о количестве студентов по курсам, странам, а также их общее количество.

## Анализ бизнес-процессов

В отделе университета ведется учет студентов из числа иностранных граждан. Каждый студент имеет личную информацию: дата рождения, место рождения, паспортные данные, сведения о медицинской страховке, сведения о визе и информацию об образовании: дату поступления, форму обучения, номер приказа, дату и причину отчисления.

В университете если несколько факультетов на которых обучают ряду специальностей. Студенты сформированы в группы, каждая группа учится на определенной специальности. Также группы студентов сформированы, в зависимости от продолжительности пройденного ими обучения, в курсы.

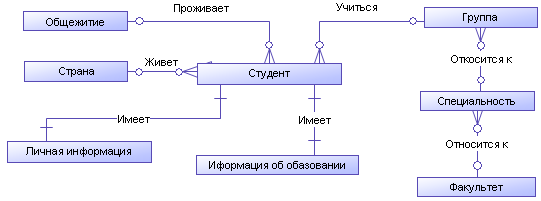
Для составления статистики, отчетов, требуется хранить всю перечисленную выше информацию.

1. ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ

## Построение инфологической модели

Информация, предоставляемая в базе данных, в первую очередь должна отображать реальные объекты прикладной области и связи между ними.

Результатом инфологического проектирования является концептуальная модель, которая представляет структуру данных не зависимую от любой физической реализации. Инфологическую модель исследуемой области можно представить следующим образом:

1. 

* Студент – сущность, в которой храниться информация о студенте: идентификатор, имя, фамилия, группа и т.д.
* Группа – сущность, содержащая информацию о группе в которой учатся студенты.
* Специальность – сущность, которая содержит данные о специальности которой обучаются группы студентов.
* Факультет – сущность, в которой содержится информация о факультете.
* Общежитие – сущность, содержащая информацию о общежитии в которой проживают студенты.
* Страна – сущность, которая содержит данные о стране в которой живут студенты.
* Личная информация – сущность, в которой храниться личная информация студента: дата рождения, место рождения, паспортные данные и т.д.
* Информация об образовании – сущность, содержащая информацию об образовании определенного студента: дата поступления, форма обучения, номер приказа и т. д.

Связи между сущностями следующие:

* В группе учатся студенты, причем студент связан только с одной группой, а группа хотя бы с одним студентом. Получаем, что сущности связаны между собой связью один ко многим (1:М);
* Специальность связана с несколькими группами, а группа с одной специальностью. Получаем, что сущности связаны между собой связью один ко многим (1:М). Аналогичным образом связаны сущности факультет и специальность, общежитие и студент, а также страна и студент;
* Сущность студент относится только к одной сущности личная информация, а личная информация относится только к одному конкретному студенту. Аналогичным образом связаны сущности студент и информацию об образовании. Эти сущности связаны между собой связью один к одному(1:1);

Выявим все атрибуты, описывающие сущности ER-модели.

Сущность Факультет (faculty) имеет атрибуты Идентификатор факультета (faculty\_id) и Имя факультета (faculty\_name).

|  |  |  |  |
| --- | --- | --- | --- |
| **Название атрибута** | **Первичный ключ** | **Внешний ключ** | **Тип** |
| Идентификатор факультета | + | - | числовой |
| Имя факультета | - | - | строковый |

Сущность Специальность (speciality) имеет атрибуты Идентификатор специальности (speciality\_id), Имя специальности (speciality\_name) и Идентификатор факультета (faculty\_id).

|  |  |  |  |
| --- | --- | --- | --- |
| **Название атрибута** | **Первичный ключ** | **Внешний ключ** | **Тип** |
| Идентификатор специальности | + | - | числовой |
| Имя специальности | - | - | строковый |
| Идентификатор факультета | - | + | числовой |

Сущность Группа (group) имеет атрибуты Идентификатор группы (group\_id), Имя группы (group\_name) и Идентификатор специальности (speciality\_id).

|  |  |  |  |
| --- | --- | --- | --- |
| **Название атрибута** | **Первичный ключ** | **Внешний ключ** | **Тип** |
| Идентификатор группы | + | - | числовой |
| Имя группы | - | - | строковый |
| Идентификатор специальности | - | + | числовой |

Сущность Студент (student) имеет атрибуты Идентификатор студента (student\_id), Имя (first\_name), Фамилия (last\_name), Отчество (middle\_name), Идентификатор группы (group\_id), Идентификатор страны (country\_id), Идентификатор общежития (hostel\_id), Идентификатор личной информации (personal\_info\_id), Идентификатор информации об образовании (education\_info\_id), Курс (course) , Адрес регистрации (registration\_address), Срок регистрации (registration\_end\_date), Дата начала каникул (vacation\_start\_date) и Дата окончания каникул (vacation\_end\_date).

|  |  |  |  |
| --- | --- | --- | --- |
| **Название атрибута** | **Первичный ключ** | **Внешний ключ** | **Тип** |
| Идентификатор студента | + | - | числовой |
| Имя | - | - | строковый |
| Фамилия | - | - | строковый |
| Отчество | - | - | строковый |
| Идентификатор группы | - | + | числовой |
| Идентификатор страны | - | + | числовой |
| Идентификатор общежития | - | + | числовой |
| Идентификатор личной информации | - | + | числовой |
| Идентификатор информации об образовании | - | + | Числовой |
| Курс | - | - | строковый |
| Адрес регистрации | - | - | строковый |
| Срок регистрации | - | - | дата |
| Дата начала каникул | - | - | дата |
| Дата окончания каникул | - | - | дата |

Сущность Личная информация (personal\_info) имеет атрибуты Идентификатор личной информации (personal\_info\_id), Дата рождения (birth\_date), Место рождения (birth\_place), Номер паспорта (pasport\_number), Срок действия паспорта (pasport\_validity), Пол (gender), Номер визы (visa\_number), Срок визы (visa\_end\_date), Номер страховки (insurance\_number) и срок страховки (insurance\_end\_date).

|  |  |  |  |
| --- | --- | --- | --- |
| **Название атрибута** | **Первичный ключ** | **Внешний ключ** | **Тип** |
| Идентификатор личной информации | + | - | числовой |
| Дата рождения | - | - | дата |
| Место рождения | - | - | строковый |
| Номер паспорта | - | - | строковый |
| Срок действия паспорта | - | - | дата |
| Номер визы | - | - | строковый |
| Срок визы | - | - | дата |
| Номер страховки | - | - | строковый |
| Срок страховки | - | - | дата |

Все атрибуты во всех сущностях не имеют значений по умолчанию и не могут принимать значение Null.

## Требования к разрабатываемой системе

Учитывая особенности предметной области, разрабатываемый прототип должен поддерживать следующие возможности:

* Добавление, изменение и удаление студентов
* Перевод студента на следующий курс
* Отчисление студента

При этом функционал прототипа должен включать возможность просмотра списка всех студентов и механизм аутентификации / авторизации пользователей. А также в просмотр отчетов по курсам, по специальностям, по странам и т. д.

## Логическая модель данных

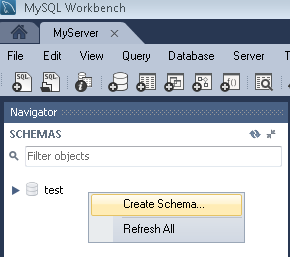
Была выбрана, как чаще всего и поступают, реляционная модель данных в связи с наглядностью табличного представления данных и удобства работы с ними. Для каждой сущности ER-модели создадим таблицу. Имя сущности – имя таблицы.

А так же определим связи между таблицами посредством механизма первичных и внешних ключей.

Таблица Факультет связана с таблицей Специальность с помощью внешнего ключа Идентификатор факультета из таблицы Специальность. Таблица Специальность связана с таблицей Группа посредством внешнего ключа Идентификатор специальности из таблицы Группа. Таблица Группа связана с таблицей Студент посредством внешнего ключа Идентификатор группы из таблицы Студент. Таблицы Студент и Страна, Студент и Общежитие, Студент и Личная информация, а также Студент и Информация об образовании связаны аналогичным образом. В Приложении А изображена схема связей между таблицами.

## Физическое проектирование

Для начала создадим новую базу данных на сервере баз данных. Для этого в MySQL Workbench откроем Navigator и вызвав контекстное меню кликните по кнопке CreateSchema (рисунок 3.2).

1. 

В появившемся окне вводим название базы данных. Назовем ее interstudents (Иностранные студенты).

В базе данных нам нужно семь таблиц. Для начала добавим таблицу faculty. Для этого создадим файл с sql-скриптом:

CREATE TABLE faculty

(

faculty\_id int not null,

faculty\_name char(60),

primary key (faculty\_id)

);

Для добавления таблицы в базу данных в меню выберем команду File за тем Open SQL Script и укажем место на жестком диске файла sql-скрипт, затем выполним его. В результате в базе данных interstudents создалась таблица faculty. Аналогичным образом создадим оставшиеся таблицы. Полный sql-скрипт находится в Приложении Б.

1. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

## Обоснование выбора ПО для разработки

Для создания программного комплекса была выбрана интегрированная среда разработки программного обеспечения Intellij Idea 14, которая содержит все необходимое для начала работы, в том числе инструменты для тестирования, компилятор и отладчик (дебаггер).

Также была использована библиотека Hibernate для отображения объектно-ориентированной модели данных в традиционные реляционные базы данных. Это библиотека для языка программирования Java, предназначенная для решения задач объектно-реляционного отображения. Она представляет собой свободное программное обеспечение с открытым исходным кодом

Для визуального проектирования баз данных был выбран MySQL Workbench — инструмент, интегрирующий проектирование, моделирование, создание и эксплуатацию БД в единое окружение для системы баз данных MySQL.

Для запуска веб-приложения используется Tomcat — контейнер сервлетов с открытым исходным кодом, разрабатываемый Apache Software Foundation. Реализует спецификацию сервлетов и спецификацию JavaServer Pages (JSP) и JavaServer Faces (JSF). Написан на языке Java. Tomcat используется в качестве самостоятельного веб-сервера, в качестве сервера контента в сочетании с веб-сервером Apache HTTP Server.

## Структура ИС и функционирование каждого блока

Созданная информационная система является трёхуровневой. Трёхуровневая архитектура программного комплекса предполагает наличие в нём трёх компонентов: клиентского приложения (обычно называемого «тонким клиентом» или интерфейсом), сервера приложения (логики приложения), к которому подключено клиентское приложение, и сервера базы данных (доступ к базе данных), с которым работает сервер приложений.

Сервер базы данных представлен MySQL-сервером; сервер приложения реализован на языке Java и web-сервером TomCat; интерфейс представлен в виде web-приложения.

В проект решения входит три части:

* модуль base
* модуль rest
* модуль angularUI

Библиотека классов base включает в себя определение и реализацию интерфейсов доступа к источнику данных. Также и реализацию источника данных (Entity Domain Models).

В состав библиотеки классов rest входят классы, описывающие логику, и реализующие rest-сервисы.

Модуль angularUI представляет интерфейс приложения реализованный с использования javascript библиотеки AngularJS.

## Структура модели данных

С уже имеющейся базы данных Hibernate может автоматически создать модель данных, состоящую из классов и свойств, соответствующих объектам базы данных (таким, как таблицы и столбцы).

Сущность faculty (факультет) представлена в листинге 4.1.

@Entity

@Table(name = "faculty")

@XmlAccessorType(value = XmlAccessType.FIELD)

public class Faculty extends AbstractEntity{

@Id

@GeneratedValue

@Column(name = "faculty\_id")

private Integer id;

@Column(name = "faculty\_name")

private String facultyName;

@Override

public Integer getId() {

return id;

}

public void setId(Integer id) {

this.id = id;

}

public String getFacultyName() {

return facultyName;

}

public void setFacultyName(String facultyName) {

this.facultyName = facultyName;

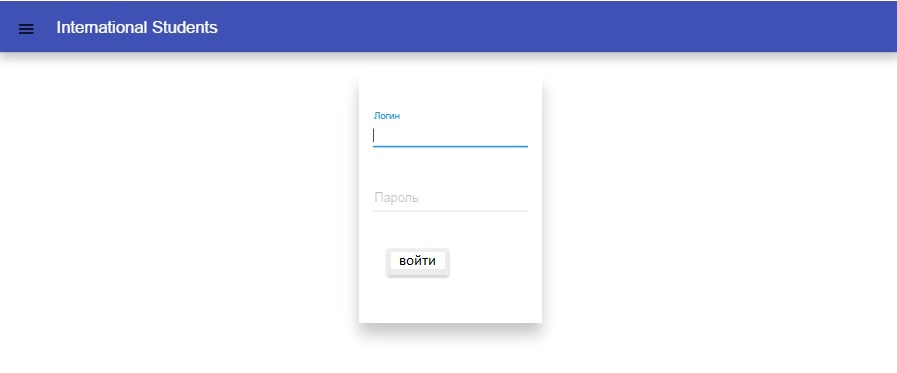
}

}

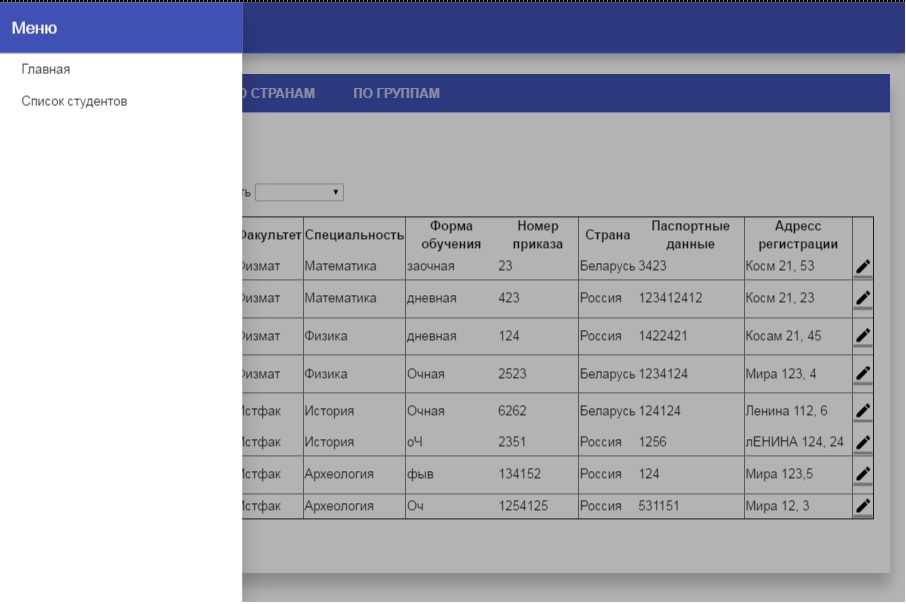
Сущность speciality (модели), group (группа), student (студент), country (страна), hostel (общежитие), personal\_info (личная информация) и education\_info (информация об образовании) описаны в Приложении В.

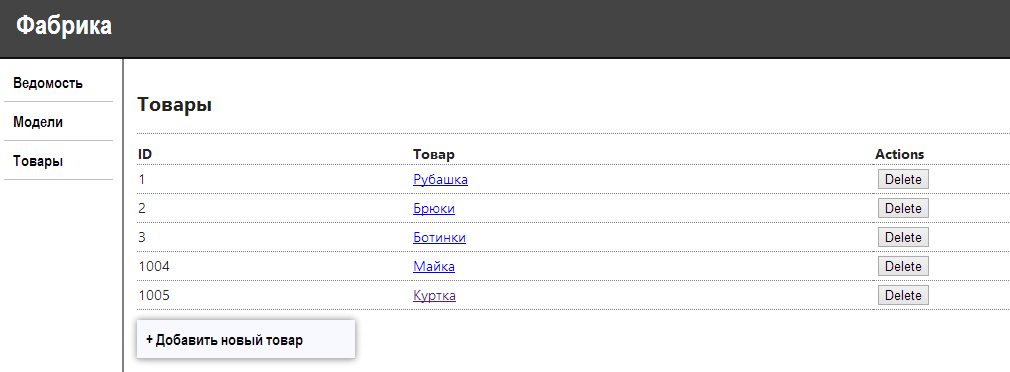
## Результаты работы ИС

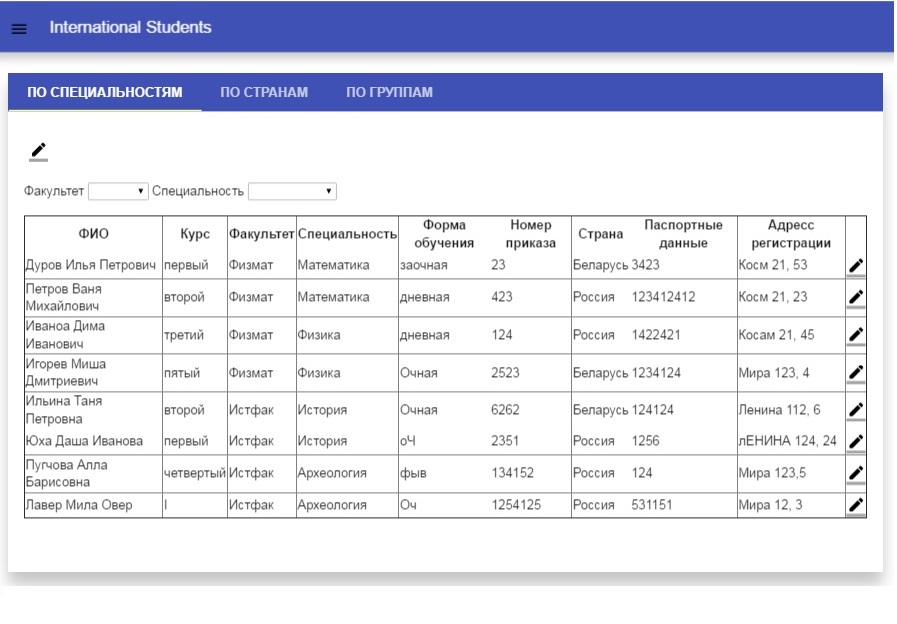
При запуске приложения страница авторизации, где пользователю предлагают ввести авторизационные данные (Рисунок 4.1).

1. 

После того как пользователь успешно прошел процесс авторизации он попадает на страницу на которой описаны возможности приложения. Также для работы он может использовать вылетающее меню (Рисунок 4.2).

1. 

При выборе в меню списка студентов пользователь переходит на страницу, где может увидеть всю необходимую информацию по специальностям, по группам, по странам и т. д. Также реализована возможность фильтрации и сортировки выбранных данных (Рисунок 4.3).

1. 

При этом, пользователи наделенные правами администратора имеют возможность редактировать и добавлять студентов в группы.

# ЗАКЛЮЧЕНИЕ

В данной дипломной работе была рассмотрена разработка и реализация автоматизированной системы учета студентов. В процессе разработки освоена технология доступа к данным Hibernate и MySQL Server, язык программирования Java, интегрированная среда разработки Intellij Idea.

Приложение разрабатывалось в соответствии с основополагающими принципами ООП: наследование, инкапсуляция, полиморфизм, а также в соответствии с подходами, описываемыми принципами SOLID и DRY.

В ходе дипломной работы были изучены принципы проектирования приложений с Rest архитектурой, а также разработка и реализация систем управления базой данных.

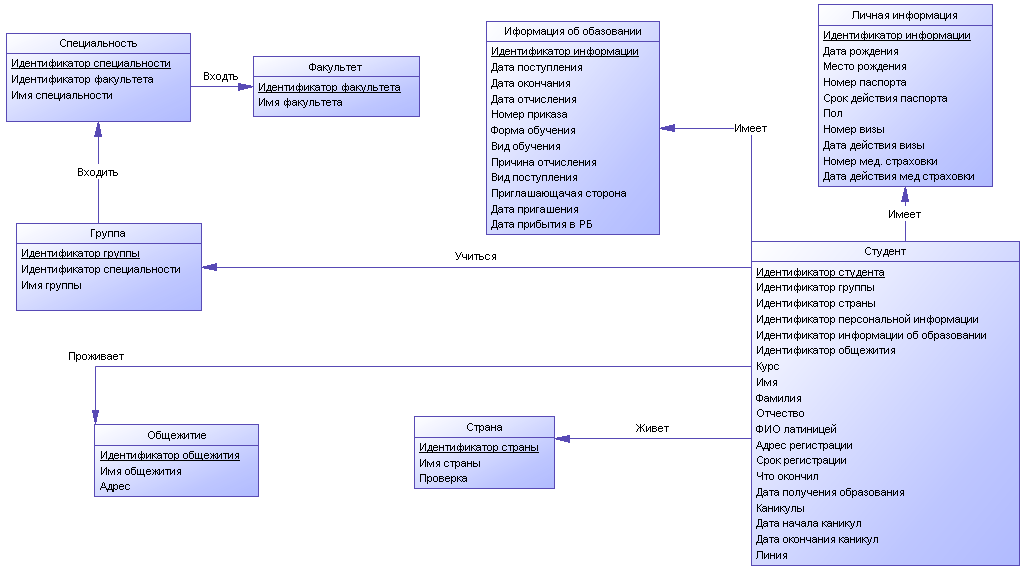
Был разработан компактный пользовательский интерфейс и использования JavaScript фреймворка AngularJS, который предоставляет возможность добавления, изменения и удаления данных. Дизайн основан на принципах построения дизайна Material desing от компании Google. Функционирование приложения можно расширить возможностью выгрузки данных в Excel.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Эккель Бр., Философия JAVA, 4-e издание, 209. – 658 с.
2. Шилдт Г., Гибкая разработка приложений на Java с помощью Spring, Hibernate и Eclipse, 2008. – 256 с.
3. Бауэр К., Hibernate в действии, 2005. – 312 с.
4. Флэнаган Д., JavaScript. Полное руководство, 6-е издание, 2012. – 711 с.
5. Microsoft Developer Network [Электронный ресурс] Режим доступа: http://msdn.microsoft.com/ru-ru/library/system.threading.aspx

Дата доступа: 10.05.2014.

# ПРИЛОЖЕНИЯ

Приложение А

Приложение Б

drop table if exists country;

drop table if exists education\_info;

drop table if exists faculty;

drop table if exists groupe;

drop table if exists hostel;

drop table if exists personal\_info;

drop table if exists speciality;

drop table if exists student;

drop table if exists user;

/\*==============================================================\*/

/\* Table: country \*/

/\*==============================================================\*/

create table country

(

country\_id int not null,

country\_name char(60),

is\_bad bool,

primary key (country\_id)

);

/\*==============================================================\*/

/\* Table: education\_info \*/

/\*==============================================================\*/

create table education\_info

(

education\_info\_id int not null,

incoming\_date date,

graduated\_date date,

dismis\_date date,

order\_number char(10),

study\_form char(60),

study\_type char(60),

reason\_dismis char(60),

income\_type char(60),

inviting\_part char(60),

invite\_date date,

incoming\_rb\_date date,

primary key (education\_info\_id)

);

/\*==============================================================\*/

/\* Table: faculty \*/

/\*==============================================================\*/

create table faculty

(

faculty\_id int not null,

faculty\_name char(60),

primary key (faculty\_id)

);

/\*==============================================================\*/

/\* Table: groupe \*/

/\*==============================================================\*/

create table groupe

(

group\_id int not null,

speciality\_id int,

group\_name char(60),

primary key (group\_id)

);

/\*==============================================================\*/

/\* Table: hostel \*/

/\*==============================================================\*/

create table hostel

(

hostel\_id int not null,

hostel\_name char(60),

address char(255),

primary key (hostel\_id)

);

/\*==============================================================\*/

/\* Table: personal\_info \*/

/\*==============================================================\*/

create table personal\_info

(

personal\_info\_id int not null,

birth\_date date,

birth\_place char(255),

pasport\_number char(60),

pasport\_validity date,

gender char(10),

visa\_number char(60),

visa\_end\_date date,

insurance\_number char(60),

insurance\_end\_date date,

primary key (personal\_info\_id)

);

/\*==============================================================\*/

/\* Table: speciality \*/

/\*==============================================================\*/

create table speciality

(

speciality\_id int not null,

faculty\_id int,

speciality\_name char(60),

primary key (speciality\_id)

);

/\*==============================================================\*/

/\* Table: student \*/

/\*==============================================================\*/

create table student

(

student\_id int not null,

group\_id int,

country\_id int,

personal\_info\_id int,

education\_info\_id int,

hostel\_id int,

course char(10),

first\_name char(60),

last\_name char(60),

middle\_name char(60),

fio\_latin char(255),

registration\_address char(255),

registration\_end\_date date,

graduated char(60),

receipt\_date date,

vacation char(10),

vacation\_start\_date date,

vacation\_end\_date date,

line char(60),

primary key (student\_id)

);

/\*==============================================================\*/

/\* Table: user \*/

/\*==============================================================\*/

create table user

(

user\_id int not null,

login char(255),

password char(255),

is\_admin bool,

primary key (user\_id)

);

alter table groupe add constraint FK\_group\_speciality foreign key (speciality\_id)

references speciality (speciality\_id) on delete restrict on update restrict;

alter table speciality add constraint FK\_speciality\_faculty foreign key (faculty\_id)

references faculty (faculty\_id) on delete restrict on update restrict;

alter table student add constraint FK\_student\_counrty foreign key (country\_id)

references country (country\_id) on delete restrict on update restrict;

alter table student add constraint FK\_student\_education\_info foreign key (education\_info\_id)

references education\_info (education\_info\_id) on delete restrict on update restrict;

alter table student add constraint FK\_student\_group foreign key (group\_id)

references groupe (group\_id) on delete restrict on update restrict;

alter table student add constraint FK\_student\_hostel foreign key (hostel\_id)

references hostel (hostel\_id) on delete restrict on update restrict;

alter table student add constraint FK\_student\_personal\_info foreign key (personal\_info\_id)

references personal\_info (personal\_info\_id) on delete restrict on update restrict;

Приложение В

@Entity

@Table(name = "speciality")

@XmlAccessorType(value = XmlAccessType.FIELD)

public class Speciality extends AbstractEntity{

@Id

@GeneratedValue

@Column(name = "speciality\_id")

private Integer id;

@Column(name = "speciality\_name")

private String specialityName;

@ManyToOne

@JoinColumn(name = "faculty\_id")

private Faculty faculty;

@Override

public Integer getId() {

return id;

}

public void setId(Integer id) {

this.id = id;

}

public String getSpecialityName() {

return specialityName;

}

public void setSpecialityName(String specialityName) {

this.specialityName = specialityName;

}

public Faculty getFaculty() {

return faculty;

}

public void setFaculty(Faculty faculty) {

this.faculty = faculty;

}

}

@Entity()

@Table(name = "groupe")

public class Group extends AbstractEntity{

@Id

@GeneratedValue

@Column(name = "group\_id")

private Integer id;

@Column(name = "group\_name")

private String groupName;

@ManyToOne

@JoinColumn(name = "speciality\_id")

private Speciality speciality;

@Override

public Integer getId() {

return id;

}

public void setId(Integer id) {

this.id = id;

}

public String getGroupName() {

return groupName;

}

public void setGroupName(String groupName) {

this.groupName = groupName;

}

public Speciality getSpeciality() {

return speciality;

}

public void setSpeciality(Speciality speciality) {

this.speciality = speciality;

}

}