Министерство образования и науки Российской Федерации

ФГАОУ ВПО «Северо-Восточный федеральный университет имени М.К. Аммосова»

Институт математики и информатики

Кафедра информационных технологий

ДОПУСТИТЬ К ЗАЩИТЕ

Зав. кафедрой \_\_\_\_\_\_\_\_\_\_/С.Д.Мордовской/

Протокол №\_\_\_ от «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_г.

Разработка приложения

«Электронная анкета абитуриента»

(Модуль АРМ сотрудника. Административная часть)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

(направление: 09.03.01 «Информатика и вычислительная техника»)

Выполнил: студент IV курса

группы ИВТ-12, ИМИ СВФУ

Сергеев Владимир Сергеевич \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

Руководитель: старший преподаватель кафедры ИТ,   
к.ф.-м.н. Эверстов Владимир Васильевич

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

Якутск

2016

**Содержание**

[Введение 3](#_Toc452454342)

[1. Приемная кампания СВФУ им. М.К. Аммосова 5](#_Toc452454343)

[1.1 Процесс проведения обработки документов 5](#_Toc452454344)

[1.2 Концепция сервиса 8](#_Toc452454345)

[1.3 Обзор аналогов 9](#_Toc452454346)

[1.4 Методы разработки ПО 14](#_Toc452454347)

[1.5 Инструменты разработки web-приложений 17](#_Toc452454348)

[2 Разработка модуля АРМ сотрудника. Админимтративная часть 25](#_Toc452454349)

[2.1 Требования к модулю 25](#_Toc452454350)

[2.2 Проектирование 26](#_Toc452454351)

[2.3 Инструменты разработки 28](#_Toc452454352)

[2.4 Процесс разработки 29](#_Toc452454353)

[2.5 Описание модуля 30](#_Toc452454354)

[Заключение 40](#_Toc452454355)

[Список использованных источников и литературы 41](#_Toc452454356)

[Приложения 4](#_Toc452454357)4

Введение

Обработка данных, полученных от абитуриентов, является важным этапом проведения приемной кампании. На этом этапе проводится проверка корректности и самое важное – ранжирование абитуриентов согласно баллам, полученным на экзаменах с учетом их индивидуальных достижений во время учебы в школе. Ошибка, допущенная сотрудником приемной комиссии на данном этапе, дорого обходится абитуриенту.

Количество сотрудников приемной комиссии не увеличивается год от года. Следовательно, увеличивается нагрузка, приходящаяся на одного сотрудника комиссии. Поэтому автоматизация этого аспекта работы ВУЗа жизненно необходима. Это будет способствовать уменьшению вероятности совершения ошибок во время обработки документов, увеличит скорость работы сотрудников и в целом благоприятно скажется на проведении приемной кампании ФГАОУ ВПО Северо-восточный федеральный университет им. М.К. Аммосова.

**Цель проекта -** разработка веб-портала «Электронная анкета абитуриента».

**Цель данной работы:** разработка модуля АРМ Сотрудника для веб-портала «Электронная анкета абитуриента».

**Задачи:**

* Исследовать процесс проведения приемной кампании СВФУ им. М.К. Аммосова.
* Спроектировать модуль АРМ сотрудника. Административная часть.
* Разработать web-приложение модуль АРМ сотрудника. Административная часть.

**Объект исследования**: Процесс разработки программного обеспечения

**Предмет исследования:** процессы разработки веб-приложений.

**Структура выпускной квалификационной работы**. Работа состоит из введения, двух глав, заключения, приложений и списка использованных источников. В первой главе описывается механизм работы приемной комиссии и обзор инструментов разработки для создания ПО. Во второй главе описывается процесс разработки ПО. Выпускная квалификационная работа содержит 16 рисунков, три приложения.

# Приемная кампания СВФУ им. М.К. Аммосова

## Процесс проведения обработки документов

Алгоритм действий сотрудника приемной комиссии университета выглядит следующим образом:

1. Сотрудник проверяет полученные документы;

2. Заводит на каждого абитуриента папку с его документами;

3. Заполняет и выдает расписку о получении документов;

4. Помещает все документы в отдельный конверт;

5. Отправляет конверт в очередь на обработку.

После всех этих операций ответственный сотрудник либо секретарь, вносит данные в ИС ИИСУСС. рис.1

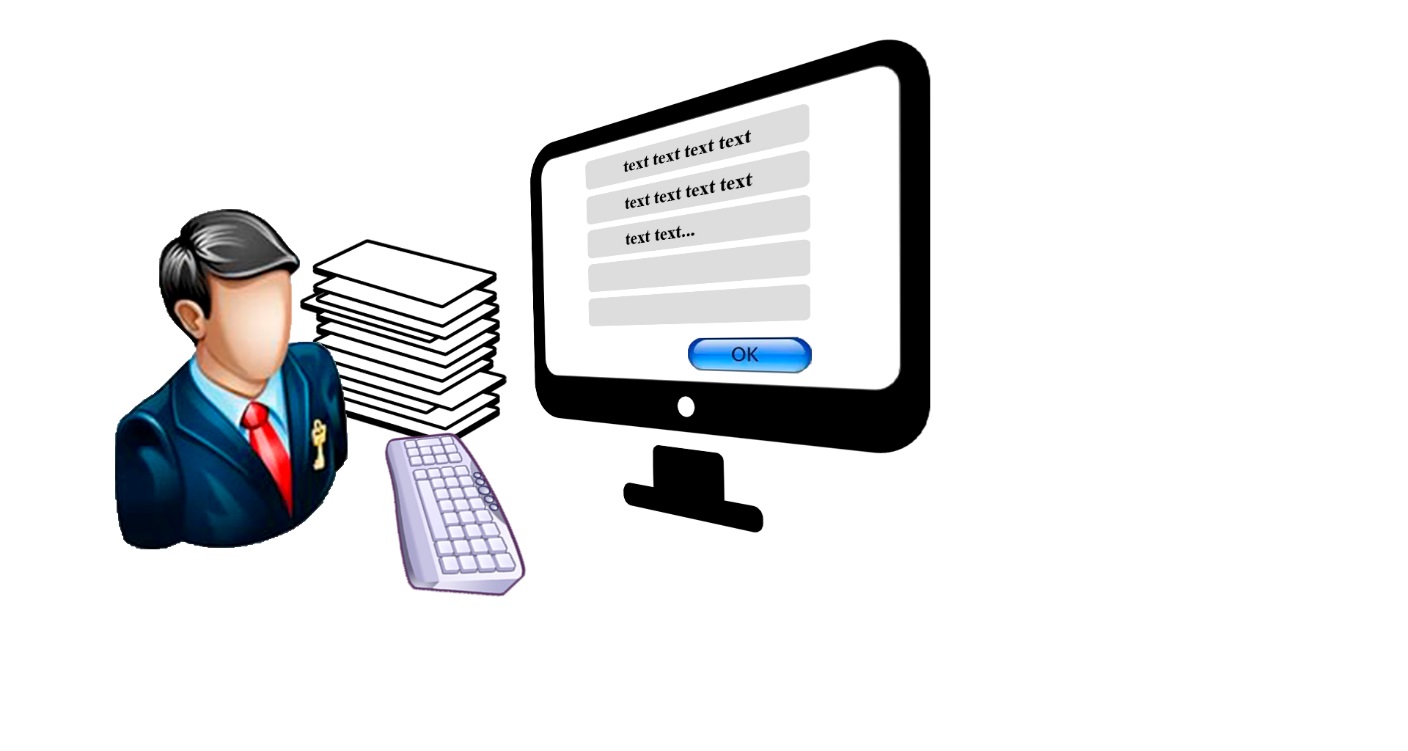


Рис.1. Процесс приема документов

В течение всего срока проведения приемной компании сотрудники комиссии должны считать количество поданных заявлений, проверять правильность заполнения документов, наличие оригиналов.

Согласно правилам приема ФГАОУ ВПО СВФУ им. М.К. Аммосова зачисление поступающих на обучение по программам бакалавриата и программам специалитета по очной форме обучения на места в рамках контрольных цифр проводится в следующие сроки [12]:

а) первый этап зачисления:

28 июля списки поступающих размещаются на официальном сайте и на информационном стенде;

30 июля завершается прием оригинала документа установленного образца от поступающих без вступительных испытаний, поступающих на места в пределах квоты приема лиц, имеющих особые права, поступающих на места в пределах квоты целевого приема (от лиц, одновременно подавших заявления о приеме в соответствии с двумя или более подпунктами пункта 45 Правил в различные организации высшего образования);

31 июля: издается и размещается на официальном сайте и на информационном стенде приказ (приказы) о зачислении поступающих без вступительных испытаний, поступающих на места в пределах квоты приема лиц, имеющих особые права, поступающих на места в пределах квоты целевого приема; количество конкурсных мест в соответствующих конкурсных списках на места в рамках контрольных цифр по общему конкурсу увеличивается на количество мест, равное числу поступающих без вступительных испытаний на места в рамках контрольных цифр, не представивших оригинал документа установленного образца, а также на количество мест, оставшихся вакантными в пределах квоты приема лиц, имеющих особые права, и квоты целевого приема; на официальном сайте и на информационном стенде размещаются конкурсные списки на места в рамках контрольных цифр по общему конкурсу с выделением списков лиц, рекомендованных к зачислению на первом этапе;

4 августа завершается прием оригинала документа установленного образца от поступающих, включенных в списки лиц, рекомендованных к зачислению на первом этапе;

5 августа издается и размещается на официальном сайте и на информационном стенде приказ (приказы) о зачислении поступающих, включенных в списки лиц, рекомендованных к зачислению на первом этапе, и представивших в установленный срок оригинал документа установленного образца;

б) второй этап зачисления (зачисление лиц, включенных в списки поступающих, до заполнения вакантных мест):

5 августа на официальном сайте и на информационном стенде размещаются конкурсные списки на места в рамках контрольных цифр по общему конкурсу с выделением в них списков лиц, рекомендованных приемной комиссией к зачислению на втором этапе с учетом оставшегося количества мест (в том числе мест, освободившихся в связи с непредставлением в установленный срок (отзывом) оригинала документа установленного образца);

8 августа завершается прием оригинала документа установленного образца от поступающих, включенных в списки лиц, рекомендованных к зачислению на втором этапе;

11 августа издается и размещается на официальном сайте и на информационном стенде приказ (приказы) о зачислении поступающих, включенных в списки лиц, рекомендованных к зачислению на втором этапе, и представивших в установленный срок оригинал документа установленного образца.

При приеме на обучение по программам бакалавриата и программам специалитета по заочной форме обучения, на обучение по договорам об оказании платных образовательных услуг, а также на обучение по программам магистратуры зачисление завершается не позднее чем за 10 дней до начала учебного года. Зачисление на места по договорам об оказании платных образовательных услуг проводится после зачисления на места в рамках контрольных цифр либо вне зависимости от сроков зачисления на места в рамках контрольных цифр [12].

Лица, включенные в список лиц, рекомендованных к зачислению на первом или втором этапе и не представившие в установленный срок (отозвавшие) оригинал документа установленного образца, выбывают из конкурса и рассматриваются как отказавшиеся от зачисления на места в рамках контрольных цифр приема. 89. Если поступающий поступает на обучение одновременно по различным условиям, указанным в подпункте 1 и (или) подпункте 2 пункта 8 Правил, то при зачислении на обучение по каким-либо условиям он исключается из списков, поступающих по тем направлениям, которые определены им в заявлении о приеме как менее приоритетные [12].

Представленные поступающим оригиналы документов возвращаются лицу, отозвавшему поданные документы, либо не поступившему на обучение, в соответствии со способом возврата поданных документов, указанным в заявлении об отзыве поданных документов или в заявлении о приеме на обучение, в течение 20 рабочих дней соответственно после отзыва поданных документов или после подведения итогов конкурса [12].

После окончания приемной компании ответственному секретарю необходимо заполнить отчет о проведенном наборе новоиспеченных студентов университета. Этот отчет содержит различную информацию о абитуриентах, например [12]:

1. Средний балл ЕГЭ по различным срезам:
   1. По подразделению,
   2. По направлению подготовки,
   3. По профилю.

## Концепция сервиса

Пользуясь нашим сервисом «Электронная анкета абитуриента» сотрудник приемной кампании не будет вводить данные абитуриента вручную. Деятельность сотрудника приемной кампании будет сводится к проверке правильности введённых данных, предварительно заполненных абитуриентом, и экспортированию их в систему учета данных студентов СВФУ ИИСУСС рис.2.

Т.е. помимо стандартных обязанностей, сотруднику необходимо еще выполнять дополнительные, которые диктуются разрабатываемым сервисом, но эти обязанности не предполагают выполнение ими каких-либо действий вручную, которых будет очень мало. Основные действия сотрудника, ответственного за прием документов, в предлагаемом нами портале:

* Сотрудник должен найти заявление подошедшего абитуриента по уникальному номеру,
* Далее он сверяет правильность заполнения заявления,
* И меняет статус заявления на «проверенный» завершая процесс сверки данных.

Обязанности ответственного сотрудника:

* Окончательная проверка на присутствие возможных ошибок,
* Экспортировать заявление в интегрированную информационную систему учета студенческого состава (ИИСУСС).



Рис.2. Концепция предлагаемой системы

Предлагаемая концепция веб-сервиса освобождает сотрудника приемной кампании от ручного ввода и бумажной волокиты, что способствует уменьшению ошибок, а также увеличению скорости работы и, в общем, положительно повлияет на приемную кампанию в целом.

## Обзор аналогов

Информационная система "ПРИЕМНАЯ КОМИССИЯ"

Программа «Приемная комиссия» (рис.3) позволяет автоматизировать работу от подготовки к приему до подведения итогов деятельности приемной комиссии, включая:

* создание электронного личного дела абитуриента;
* учет результатов вступительных испытаний;
* проверку документов в федеральной базе свидетельств ЕГЭ;
* формирование экзаменационных групп;
* создание экзаменационных ведомостей;
* создание сводной ведомости на зачисление;
* подготовку отчетов и статистики.

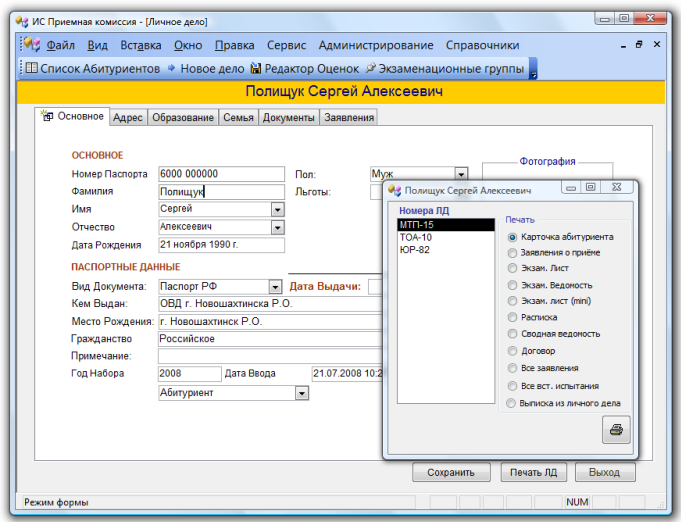


Рис.3. ИС "ПРИЕМНАЯ КОМИССИЯ"

Единое личное дело хранит информацию обо всех специальностях, на которые подает документы абитуриент. После создания электронного личного дела абитуриента можно сформировать и распечатать все необходимые документы.

В процессе подготовки к приему абитуриентов можно настроить:

* перечень специальностей, на который ведется прием;
* план приема и формы обучения;
* список дисциплин и видов вступительных испытаний;
* шкалу перевода оценок по ЕГЭ, в том числе раздельно для каждой специальности.

Пользовательский интерфейс программы (рис.3) оптимизирован для быстрого ввода данных и проверки вводимой информации на корректность. После ввода данных возможна автоматизированная проверка паспортных данных и оценок по ЕГЭ в Федеральной базе данных свидетельств ЕГЭ (ФБС).

Особое внимание в программе уделено отчетам, позволяющим упростить работу сотрудникам приемной комиссии. В программе доступны отчеты:

* Карточка абитуриента – личное дело абитуриента для выбранного заявления.
* Заявление о приеме – заполненный бланк заявления о приеме на имя ректора на выбранную специальность.
* Экзаменационный лист абитуриента.
* Вступительные испытания абитуриента определённого заявления.
* Экзаменационная ведомость для заданной группы и дисциплины.
* Сводная ведомость – сводный список абитуриентов для зачисления.
* Журнал набора – информация об абитуриентах, подавших заявления на специальность.
* Сводка по специальностям – плановый и фактический набор по специальностям.
* Отчеты для ФБС – подготовка файла для массовой проверки свидетельств ЕГЭ и аттестатов во всероссийской базе данных результатов ЦТ и ЕГЭ.

Одна из особенностей системы – это гибкая настройка программы под требования учебного заведения. Формы и отчеты программы могут изменять непосредственно пользователи программы.

ИС «Приемная комиссия» интегрирована с ИС «Деканат» и позволяет экспортировать личные дела абитуриентов в базу студентов. Программа включает в себя бесплатную поддержку, обновления и может устанавливаться на неограниченное число компьютеров в образовательном учреждении.

"БИТ. ВУЗ. Приемная комиссия"

Программа "БИТ.ВУЗ.Приемная комиссия" (рис.4), входят в линейку "БИТ.ВУЗ". Продукты предназначены для комплексной автоматизации задач приемной комиссии, подготовительных курсов, деканатов, учебно-методического отдела.

Программа "БИТ.ВУЗ.Приемная комиссия" предназначена для автоматизации всего цикла задач приемной комиссии вуза: регистрация данных абитуриентов, формирование списков, допущенных к вступительным экзаменам, экзаменационных ведомостей, плана приема, списков зачисленных, приказа о зачислении, а также предоставление отчетности по итогам работы.

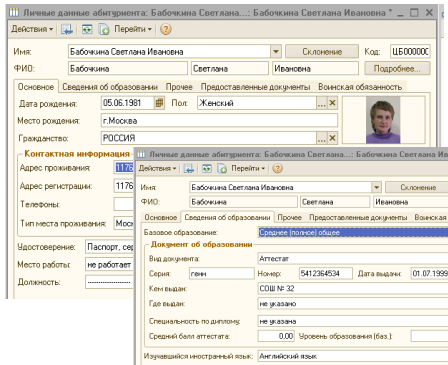


Рис.4. Интерфейс БИТ. ВУЗ

С внедрением «БИТ.ВУЗ.Приемная комиссия» вуз получит эффективный инструмент для проведения приемных кампаний и сделает первый шаг на пути к созданию полноценной комплексной автоматизированной системы учета и управления.

Программа решает задачи:

* Регистрация анкетных данных и заявлений абитуриентов (при поступлении сразу на несколько специальностей абитуриентом подается одно заявление, есть возможность указать приоритеты по направлениям);
* Формирование необходимого пакета документов для абитуриента (расписки, описи, анкета и т.д.);
* Учет результатов ЕГЭ. Реализована автоматическая проверка результатов ЕГЭ по данным, как Федеральной базы свидетельств (ФБС), так и Федеральной информационной системы (ФИС). Также учитываются целевые направления и письма направляющих организаций;
* Утверждение списка дисциплин, по которым сдаются вступительные экзамены, для каждой конкурсной группы;
* Формирование списков, рекомендованных к зачислению по результатам ЕГЭ и вступительных испытаний с учетом наличия у абитуриента льгот при поступлении в вуз;
* Формирование отчетности для анализа данных об абитуриентах, поданных заявлениях и результатах вступительных испытаний в разрезе факультетов, специальностей, форм обучения, видов финансирования и др.

Программа может быть использована при создании информационной системы персональных данных любого класса, дополнительная сертификация решения не требуется.

Результаты внедрения программы:

* Хранение данных всех абитуриентов университета в единой базе;
* Контроль хода приема в режиме онлайн;
* Формирование отчетности по итогам работе приемной комиссии в предельно сжатые сроки;
* Автоматизированная передача данных о поступивших в деканаты, нет необходимости дублирования.

## Методы разработки ПО

**Методология** — это система принципов, а также совокупность идей, понятий, методов, способов и средств, определяющих стиль разработки программного обеспечения [10].

Методология — это реализация стандарта. Сами стандарты лишь говорят о том, что должно быть, оставляя свободу выбора и адаптации [10].

Конкретные вещи реализуется через выбранную методологию. Именно она определяет, как будет выполняться разработка. Существует много успешных методологий создания программного обеспечения. Выбор конкретной методологии зависит от размера команды, от специфики и сложности проекта, от стабильности и зрелости процессов в компании и от личных качеств сотрудников [10].

Методологии представляют собой ядро теории управления разработкой программного обеспечения. К существующей классификации в зависимости от используемой в ней модели жизненного цикла (водопадные и итерационные методологии) добавилась более общая классификация на прогнозируемы и адаптивные методологии [10].

**Waterfall** - Каскадная модель процесса разработки программного обеспечения, в которой процесс разработки выглядит как поток, последовательно проходящий фазы [9]:

* Анализ требований
* Планирование
* Реализация
* Тестирование
* Установка
* Поддержка

Каскадная модель подходит для больших проектов с большими сроками, большими штатами, большим функционалом: сложные системы в банковской сфере, большие интернет-магазины, проекты с большим числом пользователей и так далее. ТЗ для таких проектов может занимать тысячи страниц, но для разработчиков такой процесс фактически идеален: все описано, всё завизировано и утверждено, сделано максимально всё, чтобы понять, как должна работать система. Водопад даёт качественный результат в силу чёткого следования порядку работы, отсутствию смены требований [9].

Минус системы: при необходимости изменений возникает большой объём "бюрократических" работ: согласование и утверждение со всеми заинтересованными лицами [9].

**Agile** - это ускоряющая методология создания проектов. Она минимизирует риски посредством коротких (2 – 3 недели) циклов, или итераций, разработки. Отдельная итерация представляет собой миниатюрный программный проект. Она включает все необходимые для выдачи мини-прироста по функциональности задачи, а именно [1]:

* проектирование;
* анализ требований;
* кодирование;
* тестирование;
* документирование.

Обычно отдельной итерации недостаточно для выпуска новой версии продукта, но подразумевается, что в конце каждой итерации гибкий программный проект готов к выпуску. То есть проект можно разделить на маленькие части, приносящие прибыль уже сразу после разработки. Каждая итерация призвана решать приоритетные на момент итерации задачи. По окончании каждой итерации командой выполняется переоценка приоритетных задач разработки [1].

**Feature Driven Development**- эта методология (кратко именуемая FDD) была разработана Джеффом Де Люка (Jeff De Luca) и признанным гуру в области объектно-ориентированных технологий Питером Коадом (Peter Coad). Как и остальные адаптивные методологии, она делает основной упор на коротких итерациях, каждая из которых служит для проработки определенной части функциональности системы. Согласно FDD, одна итерация длится две недели. FDD насчитывает пять процессов. Первые три из них относятся к началу проекта [4].

1. Разработка общей модели
2. Составление списка требуемых функций системы
3. Планирование работы над каждым свойством
4. Проектирование каждого свойства
5. Конструирование каждого свойства

Последние два шага необходимо делать во время каждой итерации. При этом каждый процесс разбивается на задачи и имеет критерии верификации.

1. Разработка общей модели.

Проект начинается с многоуровневого определения границ системы и её контекста (как я понимаю, доменной модели). Далее, детализированная модель создается для каждой области моделирования. Для удобства поддержки, они делятся на маленькие группы, объединяющиеся по областям. Эти доменные модели областей, в итоге, объединятся в общую модель системы [4].

1. Составление списка требуемых функций системы.

Знания при создании модели используются для построения списка функций системы. Это сделано при помощи функциональной декомпозиции домена по предметным областям. Они содержать бизнес-активности, а также описание, какая из бизнес-активностей формирует какой список функций системы. Функции системы являются малыми составляющими клиентский функций, например, “посчитать количество продаж”. Функции системы должны быть реализованы не более чем за 2 недели, иначе их следует делить на меньшие части [4].

1. Планирование работы над каждым свойством.

После того, как сделан список функции системы, необходимо разработать план разработки. Функции системы назначаются как классы ведущим разработчикам [4].

1. Проектирование каждого свойства.

Под дизайном здесь не понимается дизайн интерфейса. Уточняется диаграмма последовательности по группе функций системы, которые можно реализовать за 2 недели [4].

1. Конструирование каждого свойства.

Реализация выбранной функциональности [4].

## Инструменты разработки web-приложений

1. **Python**

Python — высокоуровневый язык программирования общего назначения, ориентированный на повышение производительности разработчика и читаемости кода. Синтаксис ядра Python минималистичен. В то же время стандартная библиотека включает большой объём полезных функций [6].

Python поддерживает несколько парадигм программирования, в том числе структурное, объектно-ориентированное, функциональное, императивное и аспектно-ориентированное. Основные архитектурные черты — динамическая типизация, автоматическое управление памятью, полная интроспекция, механизм обработки исключений, поддержка многопоточных вычислений и удобные высокоуровневые структуры данных. Код в Python организовывается в функции и классы, которые могут объединяться в модули (они в свою очередь могут быть объединены в пакеты) [6].

Преимущества языка программирования python [6]:

* открытая разработка;
* довольно прост в изучении, особенно на начальном этапе;
* особенности синтаксиса стимулируют программиста писать хорошо читаемый код;
* предоставляет средства быстрого прототипирования и динамической семантики;
* имеет большое сообщество, позитивно настроенное по отношению к новичкам;
* множество полезных библиотек и расширений языка можно легко использовать в своих проектах благодаря предельно унифицированному механизму импорта и программным интерфейсам;
* механизмы модульности хорошо продуманы и могут быть легко использованы;
* абсолютно всё в Python является объектами в смысле ООП, но при этом объектный подход не навязывается программисту.

Имеет следующие недостатки [6]:

* не слишком удачная поддержка многопоточности;
* на Python создано не так уж много качественных программных проектов по сравнению с другими универсальными языками программирования, например, с Java;
* отсутствие коммерческой поддержки средств разработки (хотя эта ситуация со временем меняется);
* изначальная ограниченность средств для работы с базами данных;
* бенчмарки показывают меньшую производительность Python по сравнению с основными Java VM, что создаёт этому языку репутацию медленного.

1. **Ruby**

Ruby — динамический, рефлективный, интерпретируемый высокоуровневый язык программирования для быстрого и удобного объектно-ориентированного программирования. Язык обладает независимой от операционной системы реализацией многопоточности, строгой динамической типизацией, сборщиком мусора и многими другими возможностями.

Основное назначение Ruby — создание простых и в то же время понятных программ, где важна не скорость работы программы, а малое время разработки, понятность и простота синтаксиса[7].

Основные преимущества [7]:

* Открытая разработка;
* работает на многих платформах;
* может внедряться в HTTP-разметку;
* относится к языкам программирования сверхвысокого уровня (VHLL), то есть обладает высоким уровнем абстракции и предметным подходом в реализации алгоритмов;
* реализует концептуально чистую объектно-ориентированную парадигму;
* предоставляет продвинутые методы манипуляции строками и текстом;
* легко интегрирует в свои программы высокопроизводительные серверы баз данных (DB2, MySQL, Oracle и Sybase);
* благодаря VHLL программы на Ruby хорошо масштабируются и легко сопровождаются;
* простой и чистый синтаксис значительно облегчает программистам первые шаги в обучении этому языку;
* имеется простой программный интерфейс для создания многопоточных приложений;
* имеет продвинутые средства для работы с массивами;
* возможности языка можно расширить при помощи библиотек, написанных на C или Ruby;
* зарезервированные слова могут являться идентификаторами, если это не создаёт неоднозначности для парсера;
* дополнительные возможности для обеспечения безопасности;
* встроенный отладчик.

Недостатки Ruby[7]:

* обучение языку выше начального уровня может оказаться непростым;
* информационных ресурсов, посвящённых Ruby, явно недостаточно;
* Ruby менее производителен по сравнению со многими другими языками, применяемыми в веб-разработке;
* Ruby относительно медленно разрабатывается и развивается.

1. **CMS**

Информационная система или компьютерная программа, используемая для обеспечения и организации совместного процесса создания, редактирования и управления контентом (то есть содержимым). Основные функции CMS [13]:

* Предоставление инструментов для создания содержимого, организация совместной работы над содержимым,
* Управление содержимым: хранение, контроль версий, соблюдение режима доступа, управление потоком документов и т. п.,
* Публикация содержимого,Представление информации в виде, удобном для навигации, поиска.

В системе управления содержимым могут находиться самые различные данные: документы, фильмы, фотографии, номера телефонов, научные данные и так далее. Такая система часто используется для хранения, управления, пересмотра и публикации документации. Контроль версий является одним из основных её преимуществ, когда содержимое изменяется группой лиц [13].

1. **Django**

Django — свободный фреймворк для веб-приложений на языке Python, использующий шаблон проектирования MVC. Проект поддерживается организацией Django Software Foundation[3].

Сайт на Django строится из одного или нескольких приложений, которые рекомендуется делать отчуждаемыми и подключаемыми. Это одно из существенных архитектурных отличий этого фреймворка от некоторых других (например, Ruby on Rails). Один из основных принципов фреймворка — DRY (англ. Don't repeat yourself)[3].

Также, в отличие от других фреймворков, обработчики URL в Django конфигурируются явно при помощи регулярных выражений, а не выводятся автоматически из структуры моделей контроллеров [3].

Django проектировался с использованием PostgreSQL в качестве базы данных.

В настоящее время, помимо базы данных PostgreSQL, Django может работать с другими СУБД: MySQL, SQLite, Microsoft SQL Server, DB2, Firebird, SQL Anywhere и Oracle и многими другими.

1. **Ruby on Rails**

Ruby on Rails (RoR) - Фреймворк, написанный на языке программирования Ruby, реализует архитектурный шаблон Model-View-Controller для веб-приложений, а также обеспечивает их интеграцию с веб-сервером и сервером баз данных. Является открытым программным обеспечением и распространяется под лицензией MIT[8].

Базируется на следующих принципах разработки приложений:

Максимальное использование механизмов повторного использования, позволяющих минимизировать дублирование кода в приложениях (принцип Don’t repeat yourself); по умолчанию используются соглашения по конфигурации, типичные для большинства приложений (принцип Convention over configuration) — явная спецификация конфигурации требуется только в нестандартных случаях [8].

Модель предоставляет остальным компонентам приложения объектно-ориентированное отображение данных (таких как каталог продуктов или список заказов). Объекты модели могут осуществлять загрузку и сохранение данных в реляционной базе данных, а также реализуют бизнес-логику [8].

В качестве СУБД можно использовать MySQL, PostgreSQL, SQLite, Oracle, SQL Server, DB2 или Firebird.

1. **jQuery**

jQuery — библиотека JavaScript, фокусирующаяся на взаимодействии JavaScript и HTML. Библиотека jQuery помогает легко получать доступ к любому элементу DOM, обращаться к атрибутам и содержимому элементов DOM, манипулировать ими. Также библиотека jQuery предоставляет удобный API для работы с AJAX [5].

Точно так же, как CSS отделяет визуализацию от структуры HTML, JQuery отделяет поведение от структуры HTML. Например, вместо прямого указания на обработчик события нажатия кнопки управление передаётся JQuery, которая идентифицирует кнопки и затем преобразует его в обработчик события клика. Такое разделение поведения и структуры также называется принципом ненавязчивого JavaScript[5].

Библиотека jQuery содержит функциональность, полезную для максимально широкого круга задач. Тем не менее, разработчиками библиотеки не ставилась задача совмещения в jQuery функций, которые подошли бы всюду, поскольку это привело бы к большому коду, бо́льшая часть которого не востребована. Поэтому была реализована архитектура компактного универсального ядра библиотеки и плагинов. Это позволяет собрать для ресурса именно ту JavaScript-функциональность, которая на нём была бы востребована[5].

1. **Bootstrap**

Twitter Bootstrap — свободный набор инструментов для создания сайтов и веб-приложений. Включает в себя HTML и CSS шаблоны оформления для типографики, веб-форм, кнопок, меток, блоков навигации и прочих компонентов веб-интерфейсов, включая JavaScript расширения [2].

Bootstrap использует самые современные наработки в области CSS и HTML, поэтому необходимо быть внимательным при поддержке старых браузеров[2].

Основные преимущества Twitter Bootstrap3[2]:

* Экономия времени — Bootstrap позволяет сэкономить время и усилия, используя шаблоны дизайна и классы, и сконцентрироваться на других разработках;
* Высокая скорость — динамичные макеты Bootstrap масштабируются на разные устройства и разрешения экрана без каких-либо изменений в разметке;
* Гармоничный дизайн — все компоненты платформы Bootstrap используют единый стиль и шаблоны с помощью центральной библиотеки. Дизайн и макеты веб-страниц согласуются друг с другом;
* Простота в использовании — платформа проста в использовании, пользователь с базовыми знаниями HTML и CSS может начать разработку с Twitter Bootstrap;
* Совместимость с браузерами — Twitter Bootstrap совместим с Mozilla Firefox, Yandex Browser, Google Chrome, Safari, Internet Explorer, Microsoft Edge и Opera;
* Открытое программное обеспечение — особенность Twitter Bootstrap, которая предполагает удобство использования, посредством открытости исходных кодов и бесплатной загрузки.

1. **Foundation**

Foundation - это библиотека (framework) для разработки сайтов с адаптивным (responsive) дизайном. Она включает в себя как css, так и js код. JavaScript код сделан на основе jQuery и различных плагинов, которые можно включать в код библиотеки по желанию, в зависимости от технического задания на конкретный проект [11].

Возможности и преимущества[11]:

* Создание веб-страниц с перестановкой блоков в зависимости от размера экрана пользователя и способа управления
* Простая и гибкая настройка визуального представления для разных устройств
* Использование графических акселераторов для быстрой анимации
* Готовый современный дизайн веб-форм
* Возможность использования SASS для кастомизации css кода foundation под себя

**Вывод к первой главе**

В первой главе нами был изучен процесс проведения и сбора документов приемной кампании СВФУ и проанализированы информационные системы автоматизации деятельности приемной комиссии вследствие чего нами был предложен концепт сервиса для автоматизации процесса приемной кампании «Электронная анкета абитуриента». Также проведен обзор методов разработки програмного обеспечения и основных наиболее популярных инструментов для разработки веб приложений.

# Разработка модуля АРМ сотрудника. Админимтративная часть

## Требования к модулю

Принято выделять следующие общие этапы при разработке программныого обеспечения:

* Определение требований к программе;
* Разработка технического задания;
* Разработка проекта программы;
* Кодирование программы;
* Сборка программы;
* Тестирование программных модулей;
* Ввод программы в постоянную эксплуатацию;

В самом начале мы должны выявить требования к будущему программному продукту. Разрабатываемый нами модуль должен уметь решать следующие задачи:

1. Разграничивать права доступа к функционалу и данным приложения.
2. Позволять сотрудникам
   1. Совершать поиск
      1. Заявления
      2. Абитуриента
      3. Сотрудника
      4. УЧП
      5. Справочную информацию
3. Редактировать поданные заявления
   1. Изменять,
   2. Аннулировать
4. Менять пароль
5. Управлять своими персональными данными
6. Ответственные сотрудники имеют возможность
   1. Подтверждать полноту и правильность информации в заявления абитуриента.
   2. Экспортировать данные в ИИСУСС
7. Администратор может
   1. Управлять сотрудниками
      1. Добавлять сотрудников
      2. Редактировать
      3. Деактивировать
   2. Работа со справочной информацией:
      1. Добавление
      2. Редактирование
      3. Удаление. инф. (Регионы, улусы, нас, пункты, предметы и т. д.)
8. Просматривать списки заявлений:
   1. Все
   2. Поданных
   3. Подтвержденных
   4. Экспортированных
   5. Аннулированных

## Проектирование

**Модель классов предметной области**

Для создания приложения была сконструирована модель классов (рис.5), которая состоит из 3 классов, соединенных между собой. Каждый класс содержит в себе атрибуты.

1. **Employee**

Данный класс содержит в себе информацию о сотрудниках приемной кампании. Состоит из атрибутов: *user, department, first\_name, last\_name, mid\_name, fullname, position.* Атрибуты *first\_name, last\_name, mid\_name, fullname* определяют фамилию, имя, отчество, и полное имя, *department*, *position* определяют подразделение и должность, *user* определяет аккаунт сотрудника.

1. **Contacts**

Данный класс содержит в себе информацию о контактах сотрудник приемной кампании. Состоит из атрибутов: *employee* определяет сотрудника, *contact\_type* определяет тип контакта, *value* значение контакта.

1. **Position**

Данный класс содержит в себе информацию о занимаемой должности того или иного сотрудника. Состоит из одного атрибута: *name* определяет название должности

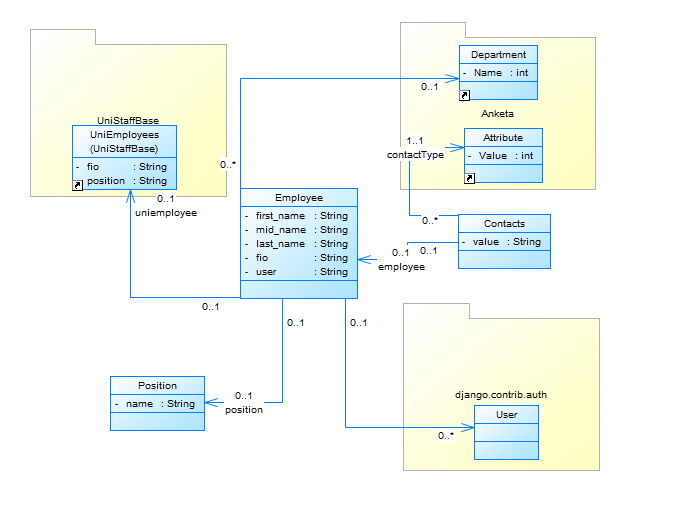
****

Рис.5. Модель классов модуля "АРМ сотрудник"

User – Это встроенный класс Django предоставляет работу с пользователями, группами пользователей, правами и сессиями пользователей работающими через куки.

UniStaffBase – Внутренняя база сотрудников СВФУ включает данные ФИО и занимаемую должность.

Anketa – это модуль сбора документов «Электронной анкеты абитуриента».

Attribute – это класс описывающий все используемые атрибуты при подачи заявлений из модели классов модуля Anketa.

Department – это класс из модели классов модуля Anketa описывающий все подразделения университета

**Архитектура модуля**

Ниже представлена архитектура web приложения, рис.6 элементов нашей разработки. Зайдя на портал, сотрудник приемной кампании авторизуется, после чего попадает на главную страницу сотрудников и имеет доступ к личному кабинету, где может редактировать личные данные, к заявлениям, где отображаются все заявления на разных этапах проверки. Сотрудник с правами администратора может добавлять, удалять, изменять сотрудников, и имеет доступ к редактированию всех справочников.

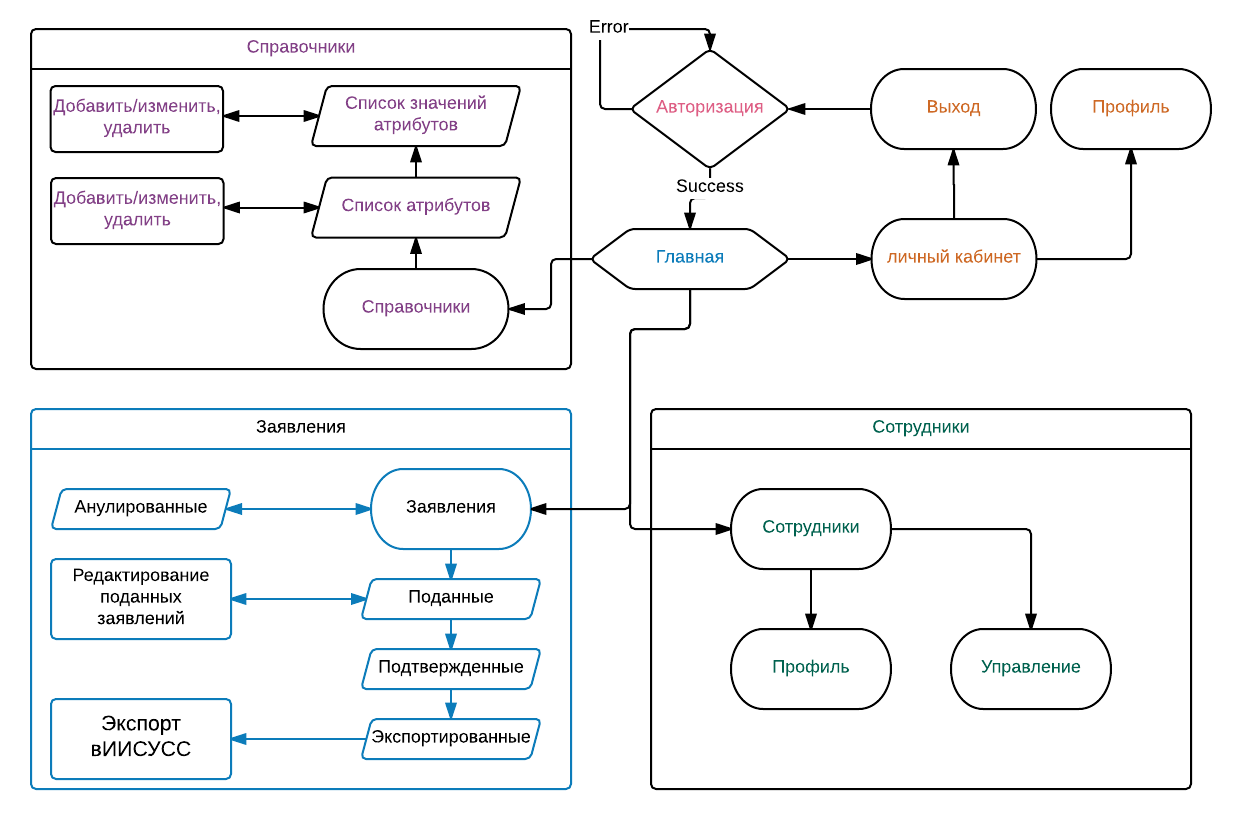


Рис.6. Карта модуля АРМ сотрудника

## Инструменты разработки

В результате анализа и сопоставления положительных и отрицательных сторон языков программирования. Нами в качестве языка программирования проекта был выбран Python, поскольку он довольно прост в изучении, по сравнению с другими языками программирования.

Выбор Django был очевиден после выбора языка программирования.

Для создания дизайна будет использоваться набор шаблонов Bootstrap. За его преимущества, bootstrap позволяет существенно сэкономить время и усилия. Имеет приятный дизайн совместим со всеми браузерами прост в использовании.

В качестве клиентского скриптового языка будет использоваться JavaScript. Но для упрощения работы мы в нашем проекте будем использовать библиотеку jQuery.

## Процесс разработки

Согласно выбранной методики разработки ПО (FDD), необходимо определиться с последовательностью реализации функциональных требований. Для этого необходимо проставить приоритеты для каждого из них. Согласно этим приоритетам нами были разработаны функции в следующем порядке:

* Администратор может
  + Управлять сотрудниками
  + Добавлять сотрудников
  + Редактировать
  + Деактивировать
* Позволять сотрудникам менять пароль
* Позволять сотрудникам управлять своими персональными данными
* Просматривать списки:
  + все
  + поданных
  + подтвержденных
  + экспортированных
  + аннулированных заявлений
* Позволять сотрудникам совершать поиск
  + заявлений
  + сотрудников
  + справочной информации
* Администратор может
  + Работать со справочной информацией:
    - добавлять
    - редактировать
    - удалять

Для начала разработки спроектированных функций необходимо начать с файла urls.py – это так сказать «содержание» все возможные пути в адресной строке. На следующем этапе нужно создать функции представления views.py (промежуточный этапп между базой данных и конкретно html разметкой), которые непосредственно пивязываются к определенным урлам за исключением функций транзакций и вспомогательных функций, вызываемых основными. После чего разрабатывается .html разметка.

В ходе разработки модуля АРМ сотрудника мы работали с такими файлами проекта Django как: urls.py, views.py, models.py, admins.py, settings.py и множество html файлов.

## Описание модуля

После авторизации сотрудник приемной кампании попадает на станицу списоков заявлений (рис.7). Где уже может пиступить к работе.

Вот так выглядит фрагмент кода автоматической пагинации с сохранением использованных фильтров.

app\_pages = Paginator(applications, 4) // по 4 элемента на страницу

page = request.GET.get('page')

try: current\_page = app\_pages.page(page)

except PageNotAnInteger: current\_page = app\_pages.page(1)

except EmptyPage: current\_page = app\_pages.page(app\_pages.num\_pages)

Разметка:

<ul class="pagination">

{% for page in pages.paginator.page\_range %}

{% if page == pages.number %}

<li class="active"><a href="{% url 'staff:application\_list' %}?page={{ page }}{% for key,value in filters.items %}{% ifnotequal key 'page' %}&{{ key }}={{ value }}{% endifnotequal %}{% endfor %}">{{ page }}</a></li>

{% else %}

<li><a href="{% url 'staff:application\_list' %}?page={{ page }}{% for key,value in filters.items %}{% ifnotequal key 'page' %}&{{ key }}={{ value }}{% endifnotequal %}{% endfor %}">{{ page }}</a></li>

{% endif %}

{% endfor %}

</ul>

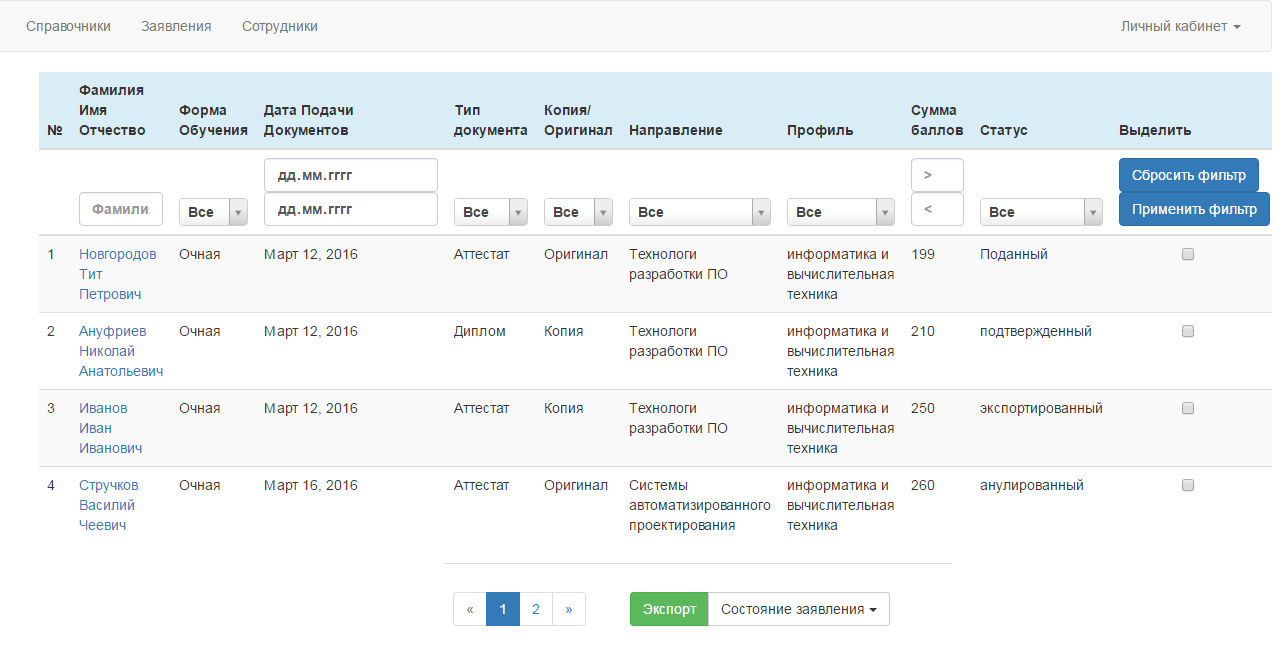


Рис.7. Список заявлений

Следующим образом выглядит отфильтрованый список (рис.8), также можно выполнить поиск абитуриента по фамилии имени и отчеству.

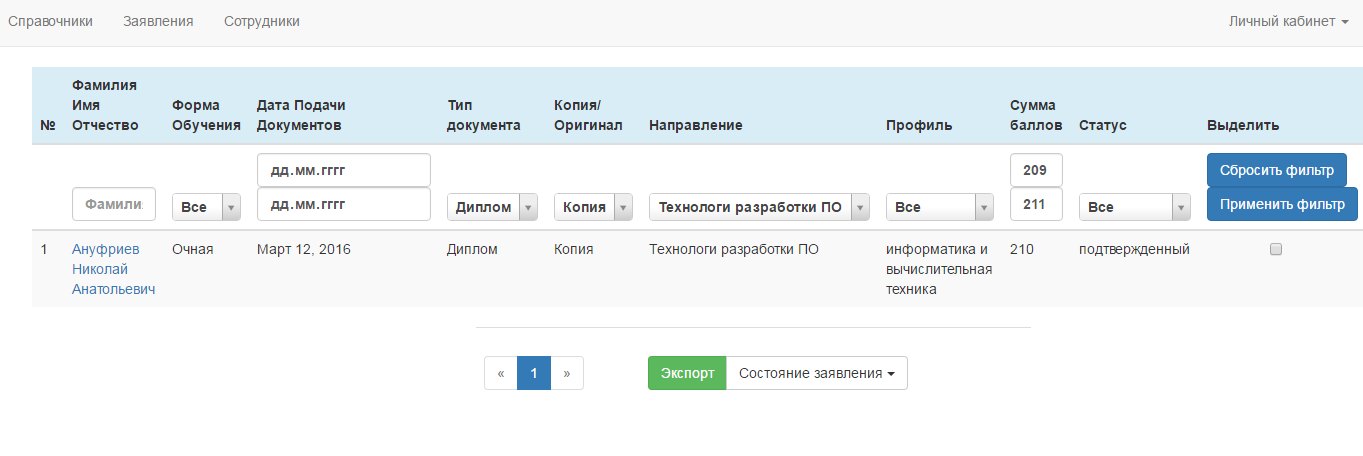


Рис.8. Отфильтрованный список заявлений

Логика фильтров, код представления (views.py)

if 'napravlenie' in request.GET and int(request.GET['napravlenie'])>0:

selectnapr = request.GET['napravlenie']

applications = applications.filter(edu\_prog\_\_edu\_prog\_\_id=selectnapr)

filters['napravlenie'] = int(selectnapr)

if 'profil' in request.GET and int(request.GET['profil'])>0:

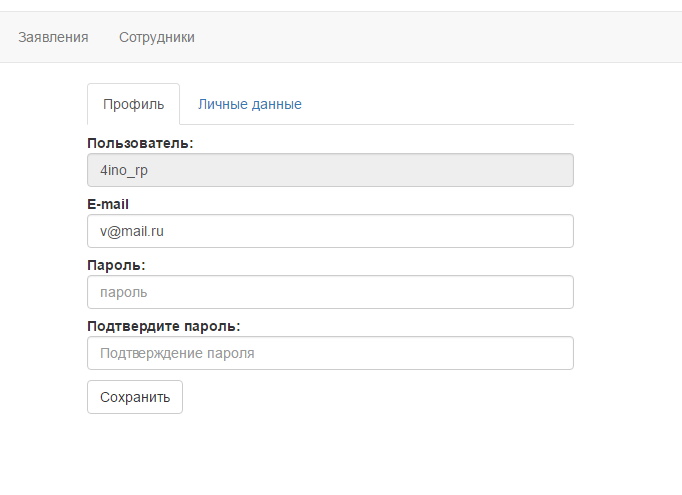
selectprof = request.GET['profil']

applications = applications.filter( edu\_prog\_\_edu\_prog\_\_qualification\_\_id=selectprof)

filters['profil'] = int(selectprof)

Таким образом оформленны все фильтры.

Так выглядит личный кабинет сотрудника (рис.9). Здесь он может изменить электронную почту, поменять пароль. При несовпадении пароля подтверждения ничего не сохранится и всплывет сообщение об ошибке.

Рис.9. Личный кабинет сотрудника (профиль)

Во вкладке личные данные (рис.10) сотудник может поменять фамилию, имя, отчество, а также денамически добавить или удалить всевозможные контактные данные.

Если пользователь вводит контактные данные в поля и нажмает на кнопку «Сохранить», то на сервер отправляется ajax-запрос на добавление контактных данных пользователя.

$(document).ready(function(){

$('.dels').on('click',function(){

var $arg=$(this).attr('value');

$.ajax({

url:"{% url 'staff:contact\_dels' %}",

method:"POST",

data:{ 'csrfmiddlewaretoken':$(

"input[name='csrfmiddlewaretoken']").val(),

'query':$arg,

},

dataType: "json",

dataProcess:true,

success:function(data)

{

if(data[0]['result']=='1'){

console.log('123'); $("tr[id='contact\_"+data[0]['id']+"']").remove();

}else{

alert(data[0]['error\_message']);

}

},

error:function()

{

alert('jsonelove');

},

});

});

});

Динамическое добавление контактных данных, запрограммировано при помощи jQuery. Если добавление прошло успешно, то на странице добавляются новые поля, которые содержат тип контакта и его значение (рис. 10).

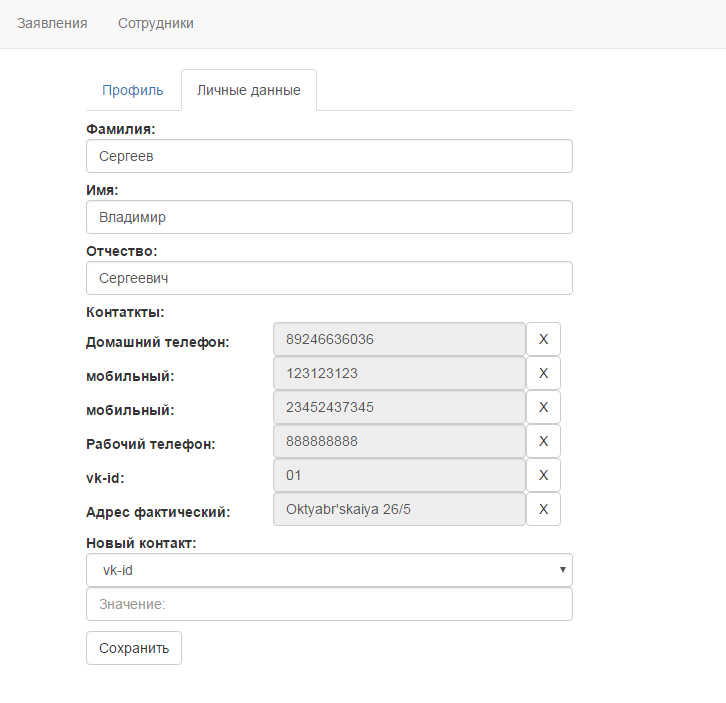


Рис.10. Личный кабинет сотрудника (личные данные)

У администратора портала есть доступ к списку сотрудников (рис.11) в котором отображаются их подразделения и должность. Здесь он может добавлять новых сотрудников и удалять уже имеющихся. Выглядит он следующим образом

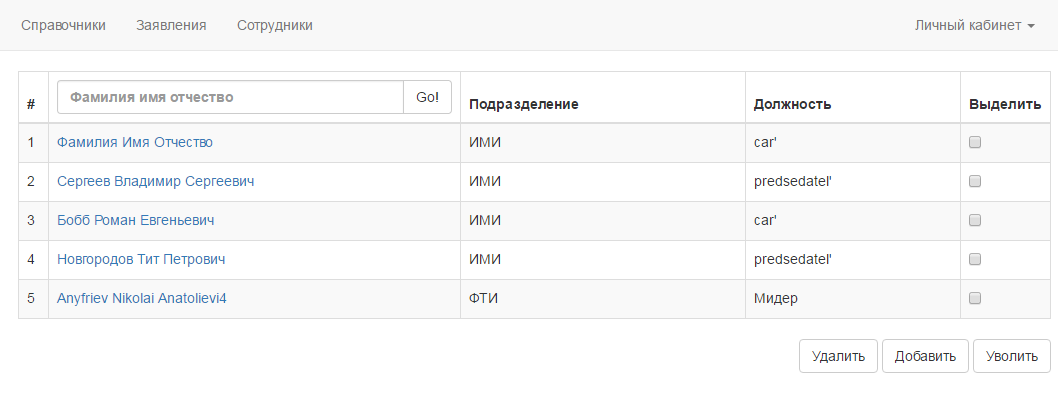


Рис.11. Список сотрудников

Реализация простого списка сотрудников

<tbody>

{% for employee in data.employee %}

<tr>

<td>{{ forloop.counter }}</td>

<td><a href="{% url 'staff:employee\_edit' employee.id %}">{{ employee.fullname }}</a></td>

<td>{{ employee.department.name }}</td>

<td>{{ employee.position }}</td>

<td><input type="checkbox" name="selected" value="{{ employee.id }}"></td>

</tr>

{% endfor %}

</tbody>

Также администратор может изменять данные сотрудников и даже сбросить пароль в случае утраты доступа к аккаунту (рис.12). Для редактирования используется тот же шаблон, что и для добавления сотрудников.

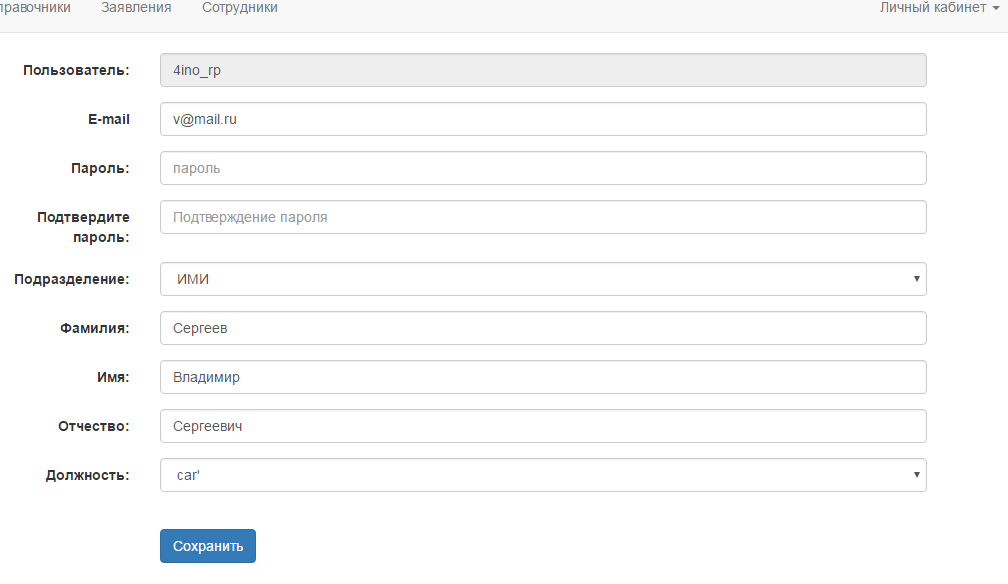


Рис.12. Добавление/изменение сотрудников

Как сотруднику, так и администратору доступен просмотр справочника всевозможных атрибутов и их значений (рис.13) с оговоркой на доступ к его изменению который доступен только администратору.

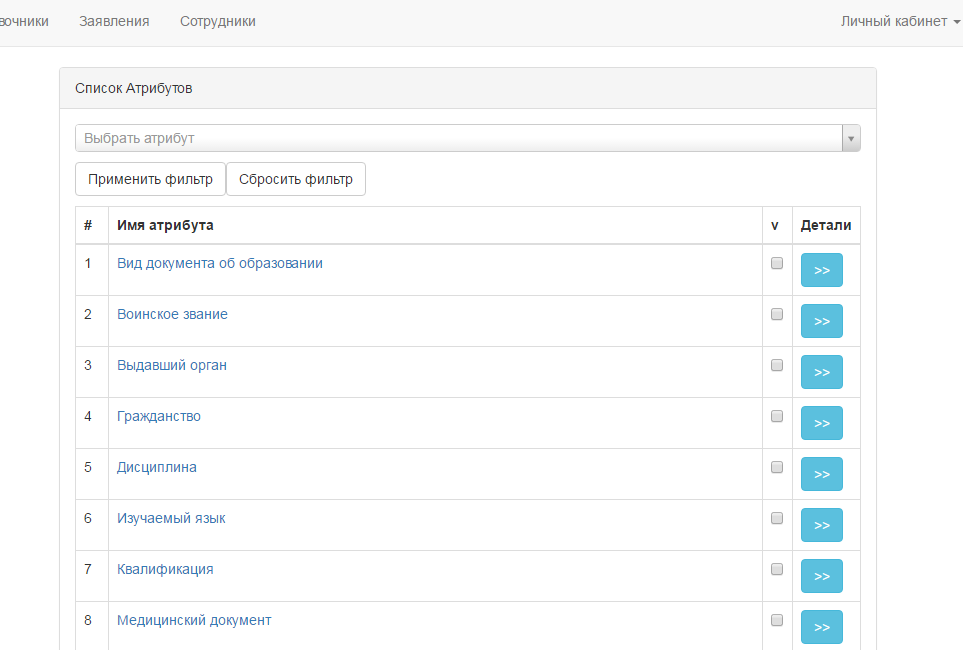


Рис.173. Список атрибутов

Также здесь реализован динамический select2 – (выпадающий список с функцией поиска), с применением технологии ajax.

$(document).ready(function(){

$('#attribute1').select2({

placeholder: "Выбрать атрибут",

language:"ru",

ajax:{

url:"{% url 'staff:get\_attrs' %}",

dataType: 'json',

delay: 250,

data:function(params){

return{

query: params,

};

},

processResults: function(data, page){

return {

results: data,

};

},

cache: true

}

});

Для добавления нового и изменения уже имеющегося атрибута было использовано модальное окно (рис.14).

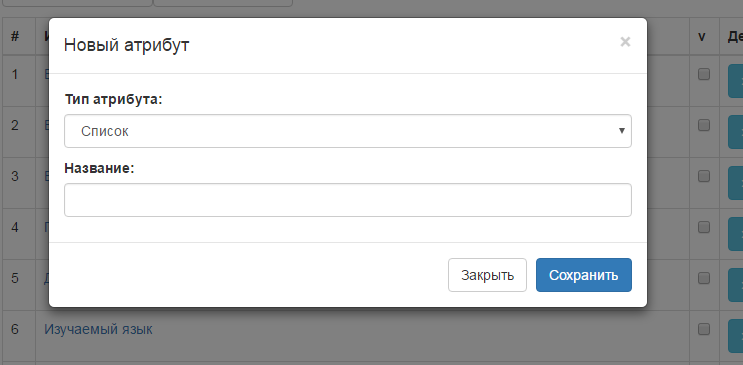


Рис.14. Добавление нового атрибута

$('.change').on('click',function(){

var $value = $(this).attr('value');

$.ajax({

url: "{% url 'staff:get\_attr' %}",

method: "GET",

data: {

'attribute\_id':$value,

},

dataType: "json",

dataProcess:true,

success:function(data)

{

$('option:selected').removeAttr('selected');

$('#attr\_id\_detail').val(data[0]['id']);

$("#attrtype\_detail>option[value='"+

data[0]['type']['id']+"']").attr('selected','selected');

$('#attr\_name\_detail').val(data[0]['name']);

$('#attr\_detail').modal();

$('#myModalLabel').html('Изменить атрибут '+'"'+data[0]['name']+'"');

},

error:function(){

alert('4toto poshlo ne tak');

}

});

});

При нажатии кнопки детали в списке атрибутов (рис.14) откроется список значений этого атрибута (рис.15)



Рис.15. Список значений атрибута

Как для атрибутов (рис.13), так и для значений атрибутов, добавление нового и изменения уже имеющегося значения используется модальное окно (рис 16).

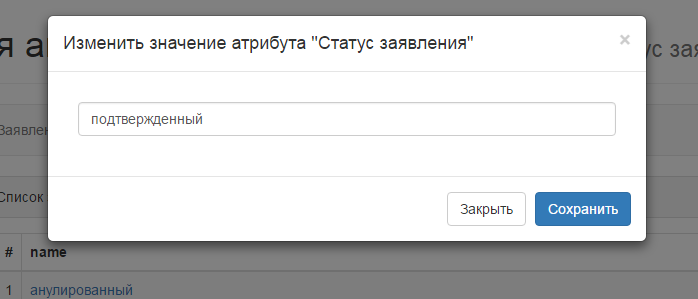


Рис.16. Добавление/изменение значения атрибута

**Выводы ко второй главе**

Входе проектирования сервиса нами была выбрана адаптивная методология разработки ПО (FDD). В качестве языка программирования выбор пал на высокоуровневый язык программирования Python. Каркасом для веб приложения на языке Python будет свободный Фреймворк Django. Для создания дизайна будет использоваться набор шаблонов Bootstrap, библиотека Java скриптов jQuery. В качестве СУБД в портале будет использован MySQL.

Исходя из требований к данному модулю была спроектирована модель классов, архитектура модуля и опираясь на расставленные приоритеты в порядке убывания были разработаны функции.

# Заключение

Подводя итоги выполнения выпускной квалификацонной работы можно сказать о том, что основные поставленные задачи были решены, цель работы достигнута: спроектирован и разработан модуль АРМ сотрудника.

В ходе проектирования модуля АРМ сотрудника был проведен обзор аналогов. Выявлены плюсы и минусы аналогичных продуктов. Рассмотрены возможные методы разработки ПО и основные инструменты WEB разработки.

Перед началом разработки модуля «АРМ сотрудника. Административная часть» нами была выбрана адаптивная методология разработки ПО и такие инструменты как: высокоуровневый язык программирования Python, фреймворк Django, Bootstrap3. И сформулированы должные требования исходя из которых была построена модель классов, архитектура модуля и разработана требуемая функциональность.

Таким образом мы автоматизировали процесс проведения приемной кампании. И разделили труд (каждый абитуриент заполняет свои данные сам, при этом сразу в электронный вид, после чего они проверяются ответственным сотрудником приемной комиссии). Что позволит сэкономить время и минимизировать риски совершения ошибок при подачи заявлений. И в целом повысит эффективность работы.

# Список использованных источников и литературы

1. Agile - гибкая методология разработки [Электронный ресурс]: Сайт авторских расширений для CMS Joomla!® allforjoomla — Электрон. дан. — Режим доступа: <http://allforjoomla.ru/info/2205-agile-gibkaya-metodologya>
2. Bootstrap (фреймворк) [Электронный ресурс]: Материал из Википедии — свободной энциклопедии: Версия 78380717, сохранённая в 02:30 UTC 16 мая 2016 / Авторы Википедии // Википедия, свободная энциклопедия. — Электрон. дан. — Сан-Франциско: Фонд Викимедиа, 2016. — Режим доступа: <http://ru.wikipedia.org/?oldid=78380717>
3. Django [Электронный ресурс]: Материал из Википедии — свободной энциклопедии: Версия 78442098, сохранённая в 02:03 UTC 19 мая 2016 / Авторы Википедии // Википедия, свободная энциклопедия. — Электрон. дан. — Сан-Франциско: Фонд Викимедиа, 2016. — Режим доступа: <http://ru.wikipedia.org/?oldid=78442098>
4. Feature Driven Development [Электронный ресурс]: WordPress с открытым исходным кодом — это популярнейшая платформа для интернет - публикаций. /Автор: Киселев Алексей / — Электрон. дан. — Режим доступа: <https://akiselev87.wordpress.com/2011/02/23/feature-driven-development/>
5. JQuery [Электронный ресурс]: Материал из Википедии — свободной энциклопедии: Версия 77973626, сохранённая в 15:46 UTC 24 апреля 2016 / Авторы Википедии // Википедия, свободная энциклопедия. — Электрон. дан. — Сан-Франциско: Фонд Викимедиа, 2016. — Режим доступа: <http://ru.wikipedia.org/?oldid=77973626>
6. Python [Электронный ресурс]: Материал из Википедии — свободной энциклопедии: Версия 77643549, сохранённая в 01:22 UTC 15 мая 2016 / Авторы Википедии // Википедия, свободная энциклопедия. — Электрон. дан. — Сан-Франциско: Фонд Викимедиа, 2016. — Режим доступа: <http://ru.wikipedia.org/?oldid=77643549>
7. Ruby [Электронный ресурс]: Материал из Википедии — свободной энциклопедии: Версия 78216706, сохранённая в 11:47 UTC 7 мая 2016 / Авторы Википедии // Википедия, свободная энциклопедия. — Электрон. дан. — Сан-Франциско: Фонд Викимедиа, 2016. — Режим доступа: <http://ru.wikipedia.org/?oldid=78216706>
8. Ruby on Rails [Электронный ресурс]: Материал из Википедии — свободной энциклопедии: Версия 78449816, сохранённая в 11:05 UTC 19 мая 2016 / Авторы Википедии // Википедия, свободная энциклопедия. — Электрон. дан. — Сан-Франциско: Фонд Викимедиа, 2016. — Режим доступа: <http://ru.wikipedia.org/?oldid=78449816>
9. Каскадная модель [Электронный ресурс]: Материал из Википедии — свободной энциклопедии: Версия 77453470, сохранённая в 03:41 UTC 30 марта 2016 / Авторы Википедии // Википедия, свободная энциклопедия. — Электрон. дан. — Сан-Франциско: Фонд Викимедиа, 2016. — Режим доступа: <http://ru.wikipedia.org/?oldid=77453470>, свободный
10. Методологии разработки программного обеспечения [Электронный ресурс]: «Хабрахабр»: ресурс для IT-специалистов. — Электрон. дан. — Режим доступа: <https://habrahabr.ru/sandbox/43802/>
11. Обзор библиотеки Foundation для разработки сайтов с адаптивным дизайном [Электронный ресурс]: g-rain design / сайт фрилансера Сергея Эстрина/ версия документа от 14.05.2014 01:32:18 / — Электрон. дан. — / Режим доступа: <http://g-rain-design.ru/blog/posts/foundation-responsive-framework-review/>
12. Правила приема 2016. Бакалавриат, специалитет, магистратура [Электронный ресурс]: Материал из официального сайта университета «Северо-Восточный федеральный университет имени М.К.Аммосова» / УТВЕРЖДЕНО приказом СВФУ от 14.11.15 г. № 1222-ОД — Электрон. дан. — / Режим доступа: <http://priem.s-vfu.ru/pravila-priema-2016-bakalavriat-spetsialitet-magistratura#1>
13. Система управления содержимым [Электронный ресурс]: Материал из Википедии — свободной энциклопедии: Версия 77644391, сохранённая в 10:16 UTC 8 апреля 2016 / Авторы Википедии // Википедия, свободная энциклопедия. — Электрон. дан. — Сан-Франциско: Фонд Викимедиа, 2016. — Режим доступа:<http://ru.wikipedia.org/?oldid=77644391>

# Приложения

Приложение 1. Файл views.py (список функций)

from django.shortcuts import render, render\_to\_response, redirect

from django.http import HttpResponseRedirect, HttpResponse

from django.contrib import auth

from django.core.context\_processors import csrf

from django.core.urlresolvers import reverse

from django.contrib.auth.decorators import login\_required, user\_passes\_test

from django.views.generic import ListView

from django.core.paginator import Paginator, EmptyPage, PageNotAnInteger

from django.db import transaction

from django import template

import json

from datetime import date

import datetime

from staff.models import Employee, Position, Contacts as ContactsStaff

from anketa.models import Department, Attribute, AttrType, Relation, Person, Application, Abiturient, Docs, AttrValue, Profile, Contacts, Address, Education\_Prog, Education\_Prog\_Form, Privilegies, Exams, DepAchieves, Milit, DocAttr, Achievements

from django.contrib.auth.models import User

# Create your views here.

register = template.Library()

def CheckUserIsStaff(user):

@login\_required(login\_url = '/login')

@user\_passes\_test(CheckUserIsStaff, login\_url = '/auth')

def index(request):

@login\_required(login\_url = '/login')

@user\_passes\_test(CheckUserIsStaff, login\_url = '/auth')

def login(request):

@login\_required(login\_url = '/login')

@user\_passes\_test(CheckUserIsStaff, login\_url = '/auth')

def logout(request):

@login\_required(login\_url = '/login')

@user\_passes\_test(CheckUserIsStaff, login\_url = '/auth')

def Employee\_list(request):

@transaction.atomic

def Add\_Employee(values):

@login\_required(login\_url = '/login')

@user\_passes\_test(CheckUserIsStaff, login\_url = '/auth')

def AddEmployee(request):

@login\_required(login\_url = '/login')

@user\_passes\_test(CheckUserIsStaff, login\_url = '/auth')

def EditEmployee(request, employee\_id):

@login\_required(login\_url = '/login')

@user\_passes\_test(CheckUserIsStaff, login\_url = '/auth')

def AddContact(employee, contacts):

@transaction.atomic

def save\_user\_profile(user, values):

@login\_required(login\_url = '/login')

@user\_passes\_test(CheckUserIsStaff, login\_url = '/auth')

def Employee\_Useraccount(request):

@login\_required(login\_url = '/login')

@user\_passes\_test(CheckUserIsStaff, login\_url = '/auth')

def Application\_list (request):

def attribute\_dels(values):

def attrvalue\_dels(values):

def attribute\_add(values):

def attrvalue\_add(attribute, values):

@login\_required(login\_url = '/login')

@user\_passes\_test(CheckUserIsStaff, login\_url = '/auth')

def Catalogs(request):

@login\_required(login\_url = '/login')

@user\_passes\_test(CheckUserIsStaff, login\_url = '/auth')

def Catalogs\_attrvalue(request, attribute\_id):

#================ajax functions====================================

@login\_required(login\_url = '/login')

@user\_passes\_test(CheckUserIsStaff, login\_url = '/auth')

def Get\_Attrs(request):

@login\_required(login\_url = '/login')

@user\_passes\_test(CheckUserIsStaff, login\_url = '/auth')

def Get\_Attr(request):

@login\_required(login\_url = '/login')

@user\_passes\_test(CheckUserIsStaff, login\_url = '/auth')

def Get\_Attr\_val(request):

@login\_required(login\_url = '/login')

@user\_passes\_test(CheckUserIsStaff, login\_url = '/auth')

def Contact\_dels(request):

Приложение 2. Файл urls.py

from django.conf.urls import patterns, include, url

from django.conf.urls.static import static

from django.conf import settings

from staff import views

from django.contrib import admin

admin.autodiscover()

urlpatterns = patterns('',

url(r'^login/','staff.views.login', name = 'login'),

url(r'^logout/','staff.views.logout',name = 'logout'),

#url(r'^$',views.index,name='index'),

url(r'^employee\_list/',views.Employee\_list, name = 'employee\_list'),

url(r'^employee\_add/',views.AddEmployee, name = 'employee\_add'),

url(r'^employee\_edit/(?P<employee\_id>\d+)',views.EditEmployee, name = 'employee\_edit'),

url(r'^employee\_useraccount/',views.Employee\_Useraccount, name = 'employee\_acc'),

url(r'^$',views.Application\_list, name = 'application\_list'),

url(r'^catalogs/',views.Catalogs, name = 'catalogs'),

url(r'^catalogs\_attrvalue/(?P<attribute\_id>\d+)',views.Catalogs\_attrvalue, name = 'catalogs\_attrvalue'),

url(r'^application\_review/(?P<application\_id>\d+)',views.Application\_review, name = 'application\_review'),

url(r'get\_attrs/',views.Get\_Attrs,name='get\_attrs'),

url(r'get\_attr/',views.Get\_Attr,name='get\_attr'),

url(r'get\_attr\_val/',views.Get\_Attr\_val,name='get\_attr\_val'),

url(r'contact\_dels/',views.Contact\_dels,name='contact\_dels'),

url(r'wiz\_cont\_dels/',views.Wiz\_cont\_dels, name='wiz\_cont\_dels'),

url(r'wiz\_cont\_apply',views.Wiz\_cont\_apply, name='wiz\_cont\_apply'),

url(r'^add\_data\_to\_person/$',views.AddDataToPerson,name="add\_data\_to\_person"), )

Приложение 3. Файл models.py

from django.db import models

from django.contrib.auth.models import User

from anketa.models import Attribute, Department, AttrValue, Application

class Employee(models.Model):

user = models.ForeignKey(User, verbose\_name=u'Пользователь', db\_index=True)

department = models.ForeignKey(Department,verbose\_name=u'Подразделение', db\_index=True)

position = models.ForeignKey('Position', verbose\_name=u'Должность',db\_index=True)

uniemployee = models.IntegerField(u'УнивСотрудник', db\_index=True)

fullname = models.CharField(u'ФИО', max\_length = 200, blank=True, null=True, db\_index=True)

first\_name = models.CharField(u'Фамилия', max\_length=100)

mid\_name = models.CharField(u'Имя', max\_length = 100)

last\_name = models.CharField(u'Отчество', max\_length = 100)

def \_\_str\_\_(self):

return self.fullname

def save(self, \*args, \*\*kwargs):

s=''

if len(self.first\_name)>0:

s+=self.first\_name

if len(self.mid\_name)>0:

s+=' '+self.mid\_name

if len(self.last\_name)>0:

s+=' '+self.last\_name

self.fullname = s

super(Employee, self).save(\*args, \*\*kwargs)

class Position(models.Model):

name = models.CharField(u'Должность',max\_length=200, db\_index=True)

def \_\_str\_\_(self):

return self.name

class Contacts(models.Model):

employee = models.ForeignKey(Employee, verbose\_name=u'Сотрудник', db\_index=True)

contact\_type = models.ForeignKey(AttrValue, verbose\_name=u'Тип контакта')

value = models.CharField(u'Контакт', max\_length = 150, db\_index=True)

def \_\_str\_\_(self):

return self.employee.fullname+' '+self.value