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 Web-портала

«Электронная анкета абитуриента»

(Личный кабинет абитуриента)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

(направление: 09.03.01 «Информатика и вычислительная техника»)

Выполнил: студент IV курса

группы ИВТ-12, ИМИ СВФУ

Ребров Андрей Андреевич

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

(подпись)

Руководитель: старший преподаватель кафедры ИТ

Эверстов Владимир Васильевич

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

(подпись)

Якутск

2016

Оглавление

[Введение 3](#_Toc452467838)

[Глава 1. Приемная кампания СВФУ им. М.К. Аммосова 5](#_Toc452467839)

[1.1. Процесс проведения приемной кампании СВФУ 5](#_Toc452467840)

[1.2. Концепция сервиса 11](#_Toc452467841)

[1.3. Обзор аналогов 13](#_Toc452467842)

[1.4. Методы разработки ПО 15](#_Toc452467843)

[1.5. Обзор инструментов разработки web-приложений 18](#_Toc452467844)

[Выводы к главе 1 27](#_Toc452467845)

[Глава 2. Разработка модуля «Личный кабинет абитуриента» 28](#_Toc452467846)

[2.1. Требования к модулю 28](#_Toc452467847)

[2.2. Проектирование 30](#_Toc452467848)

[2.3. Выбор инструментов разработки 37](#_Toc452467849)

[2.4. Описание ПО 37](#_Toc452467850)

[Выводы к главе 2 57](#_Toc452467851)

[Заключение 59](#_Toc452467852)

[Список использованной литературы и источников 60](#_Toc452467853)

[Приложения 6](#_Toc452467854)2

Введение

Для поступления в СВФУ абитуриенту необходим обширный список документов и справок, а также нужно заполнить заявление на поступление. Все это выполняется вручную, что занимает много времени, желающих поступить в СВФУ с каждым годом все больше и больше, а число работников приемной комиссии остается неизменным, потому образуются огромные очереди в местах приема документов, что негативно сказывается как на и абитуриентах, так и на работниках приемной комиссии.

Информационные технологии позволяют проводить сбор и обработку информации в удаленном режиме, что значительно упрощает жизнь человека. Поэтому в качестве решения проблемы предлагается создание программного обеспечения, которое автоматизирует работу приемной комиссии.

Разрабатываемый web-портал имеет следующие преимущества: уменьшает влияния человеческого фактора на сбор и обработку данных абитуриента, экономия времени, отсутствие очередей в приемную комиссию.

**Цель работы -** разработка web-портала «Электронная анкета абитуриента» для автоматизации сбора данных абитуриентов, поступающих в ФГАОУ ВПО СВФУ им. М.К. Аммосова.

**Задачи**:

* исследовать процесс проведения приемной кампании СВФУ им. М.К. Аммосова;
* проанализировать набор документов, требуемых при поступлении в СВФУ;
* разработать модуль личный кабинет абитуриента для web-портал «Электронная анкета абитуриента».

**Объект исследования**: Приемная компания ФГАОУ ВПО СВФУ им. М.К. Аммосова

**Предмет исследования:** Процесс сбора данных поступающих абитуриентов.

**Структура курсовой 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.

**Глава 1. Приемная кампания СВФУ им. М.К. Аммосова**

* 1. **Процесс проведения приемной кампании СВФУ**

Сегодня Северо-Восточный федеральный университет имени М. К. Аммосова – многоотраслевой вуз, располагающий широкими возможностями для проведения учебной, воспитательной и научно-исследовательской работы, в котором обучается около 21 тысячи студентов. В состав университета входят 6 научно-исследовательских институтов, 11 институтов, 8 факультетов, 3 филиала - Политехнический институт в г. Мирный, Технический институт в г. Нерюнгри и Чукотский филиал, который был открыт в конце 2010 года в г. Анадырь.

СВФУ ведет прием для обучения по всем уровням профессионального образования, так, в этом году, заявлен набор на 4 профессии начального профессионального образования, 7 специальностей среднего профессионального образования, 53 направления подготовки бакалавров, 12 специальностей, 22 направления подготовки магистров [1].

В СВФУ реализуется широкий спектр программ бакалавриата и программ подготовки специалиста, ориентированный на различные потребности населения и региона в целом.

Прием на обучение осуществляется на места в рамках контрольных цифр приема граждан на обучение за счет бюджетных ассигнований федерального бюджета, бюджетов субъектов Российской Федерации, местных бюджетов и на места по договорам об образовании, заключаемым при приеме на обучение за счет средств физических или юридических лиц [1].

Прием на обучение по программам бакалавриата и программам специалитета проводится на основании результатов единого государственного экзамена (ЕГЭ), признаваемых в качестве результатов вступительных испытаний, и (или) по результатам проводимых СВФУ самостоятельно вступительных испытаний [2].

Прием на обучение по программам магистратуры проводится по результатам вступительных испытаний, установление перечня и проведение которых осуществляется СВФУ самостоятельно [2].

Прием в СВФУ по программам бакалавриата и программам специалитета допускается по результатам ЕГЭ, сданным в 2012, 2013, 2014, 2015 годах. Лицам, проходившим военную службу по призыву и уволенным с военной службы, в течение года после увольнения с военной службы при поступлении предоставляется право использовать результаты ЕГЭ, сданного ими в течение года до призыва на военную службу [3].

Отдельные категории поступающих на базе среднего общего образования могут поступать на обучение по результатам общеобразовательных испытаний, проводимых СВФУ самостоятельно:

* лица с ограниченными возможностями здоровья, дети-инвалиды, инвалиды;
* иностранные граждане;
* лица прошедшие государственную итоговую аттестацию по образовательным программам среднего общего образование не в форме ЕГЭ (в том числе в иностранных образовательных организациях) в течение 1 года до дня завершения приема документов и вступительных испытаний включительно [3].

Поступающий на обучение про программам бакалавриата или программам специалитета вправе подать заявление о приеме одновременно не более чем в 5 организаций высшего образования и участвовать в конкурсе не более чем по 3 специальностям и (или) направлениям подготовки в каждой из указанных организаций [4].

По каждой из указанных специальностей и направлений подготовки поступающий может одновременно подать заявление о приеме установленной формы для обучения в головном вузе СВФУ и для обучения в филиалах СВФУ, по различным формам обучения, по различным программам бакалавриата в пределах направления подготовки (программам специалитета в пределах специальности) и (или) по различным совокупностям программ бакалавриата в пределах направления подготовки (совокупностям программ специалитета в пределах специальности), по которым проводятся конкурсы [4].

В правилах приема СВФУ указано, что для участия в конкурсах абитуриенту нужно:

1. Заполнить заявление установленной формы в с указанием следующих сведений:
2. Фамилия, имя, отчество (при наличии);
3. Дата рождения;
4. Сведения о гражданстве (отсутствии) гражданства
5. Реквизиты документа, удостоверяющего личность (в том числе указания, когда и кем выдан документ);
6. Сведения об образовании и документе установленного образца;
7. Условия поступления на обучение с указанием приоритетности поступления на обучение по различным условиям поступления;
8. При поступлении на обучение по программам бакалавриата и программам специалитета – сведения о наличии или отсутствии у поступающего особых прав (при наличии особых прав – с указанием сведений о документах, подтверждающих наличие таких прав);
9. При поступлении на обучение по программам бакалавриата и программам специалитета – сведения о сдаче ЕГЭ и его результатах (при наличии нескольких результатов ЕГЭ, срок действия которых не истек, указывается, какие результаты ЕГЭ и по каким общеобразовательным предметам должны быть использованы);
10. При поступлении на обучение по программам бакалавриата и программам специалитета - сведения о намерении участвовать в конкурсе по результатам общеобразовательных вступительных испытаний, проводимых СВФУ самостоятельно (с указанием оснований для участия в конкурсе по результатам таких вступительных испытаний и перечня вступительных испытаний);
11. Сведения о намерении сдавать вступительные испытания, проводимые СВФУ самостоятельно, (с указанием перечня вступительных испытаний);
12. Сведения о необходимости создания для поступающего специальных условий при проведении вступительных испытаний в связи с его ограниченными возможностями здоровья или инвалидностью (с указанием перечня вступительных испытаний и специальных условий);
13. Сведения о наличии или отсутствии у поступающего индивидуальных достижений (при наличии - с указанием сведений о них);
14. Сведения о наличии или отсутствии у поступающего потребности в предоставлении места для проживания в общежитии в период обучения;
15. Почтовый адрес и (или) электронный адрес (по желанию поступающего);
16. Способ возврата поданных документов в случае не поступления на обучение (в случае представления оригиналов документов)[5].
17. Предоставить оригиналы (копии) документов ответственному лицу для проверки правильности указанных данных в заявление
18. Заверить личной подписью поступающего следующие факты [5]:
19. Ознакомление поступающего (в том числе через информационные системы общего пользования):

с копией лицензии на осуществление образовательной деятельности (с приложением);

с копией свидетельства о государственной аккредитации (с приложением);

с информацией о предоставляемых поступающим особых правах и преимуществах при приеме на обучение по программам бакалавриата и программам специалитета;

с датами завершения представления поступающими оригинала документа установленного образца и (или) заявления о согласии на зачисление;

с правилами приема в СВФУ, в том числе с правилами подачи апелляции по результатам вступительных испытаний, проводимых СВФУ самостоятельно;

1. Согласие поступающего на обработку его персональных данных;
2. Ознакомление поступающего с информацией о необходимости указания в заявлении о приеме достоверных сведений и представления подлинных документов;
3. При поступлении на обучение на места в рамках контрольных цифр;

при поступлении на обучение по программам бакалавриата, программам специалитета - отсутствие у поступающего диплома бакалавра, диплома специалиста, диплома магистра;

при поступлении на обучение по программам магистратуры - отсутствие у поступающего диплома специалиста, диплома магистра, за исключением поступающих, имеющих высшее профессиональное образование, подтверждаемое присвоением им квалификации "дипломированный специалист";

при поступлении на обучение по программам бакалавриата и программам специалитета:

подтверждение одновременной подачи заявлений о приеме не более чем в 5 организаций высшего образования, включая СВФУ;

при подаче нескольких заявлений о приеме на обучение в СВФУ - подтверждение одновременной подачи заявлений о приеме не более чем по 3 специальностям и (или) направлениям подготовки СВФУ;

Документы, необходимые для поступления, представляются (направляются) в СВФУ одним из следующих способов [5]:

1. Представляются поступающим или доверенным лицом в СВФУ:

* СВФУ – Российская Федерация, Республика Саха (Якутия), 677000, г. Якутск, ул. Кулаковского, д.50, корпус технических факультетов;
* Технический институт (филиал) СВФУ в г. Нерюнгри – Российская Федерация, Республика Саха (Якутия), 678960, г. Нерюнгри, ул. Кравченко, д.16;
* Политехнический институт (филиал) СВФУ в г. Мирном - Российская Федерация, Республика Саха (Якутия), 678170, г. Мирный, ул. Тихонова, д.5/1;
* Чукотский филиал СВФУ, Российская Федерация, Чукотский автономный округ, 689000, г. Анадырь, ул. Студенческая, д.3;

При этом поступающему или доверенному лицу выдается расписка в приеме документов.

1. Направляются в СВФУ через операторов почтовой связи общего пользования по адресу: Российская Федерация, Республика Саха (Якутия), 677000, г.Якутск, ул.Белинского, д.58, Приемная комиссия СВФУ. Документы принимаются, если они поступили в СВФУ не позднее срока завершения приема документов, установленного правилами приема в СВФУ.
2. Направляются в электронно-цифровой форме способом направления скан-копий документов, необходимых для поступления, по электронной почте Приемной комиссии по адресу pk\_svfu@mail.ru либо через личный кабинет абитуриента на официальном сайте СВФУ. Документы принимаются, если они поступили на указанный электронный адрес не позднее срока завершения приема документов, установленного правилами приема в СВФУ.

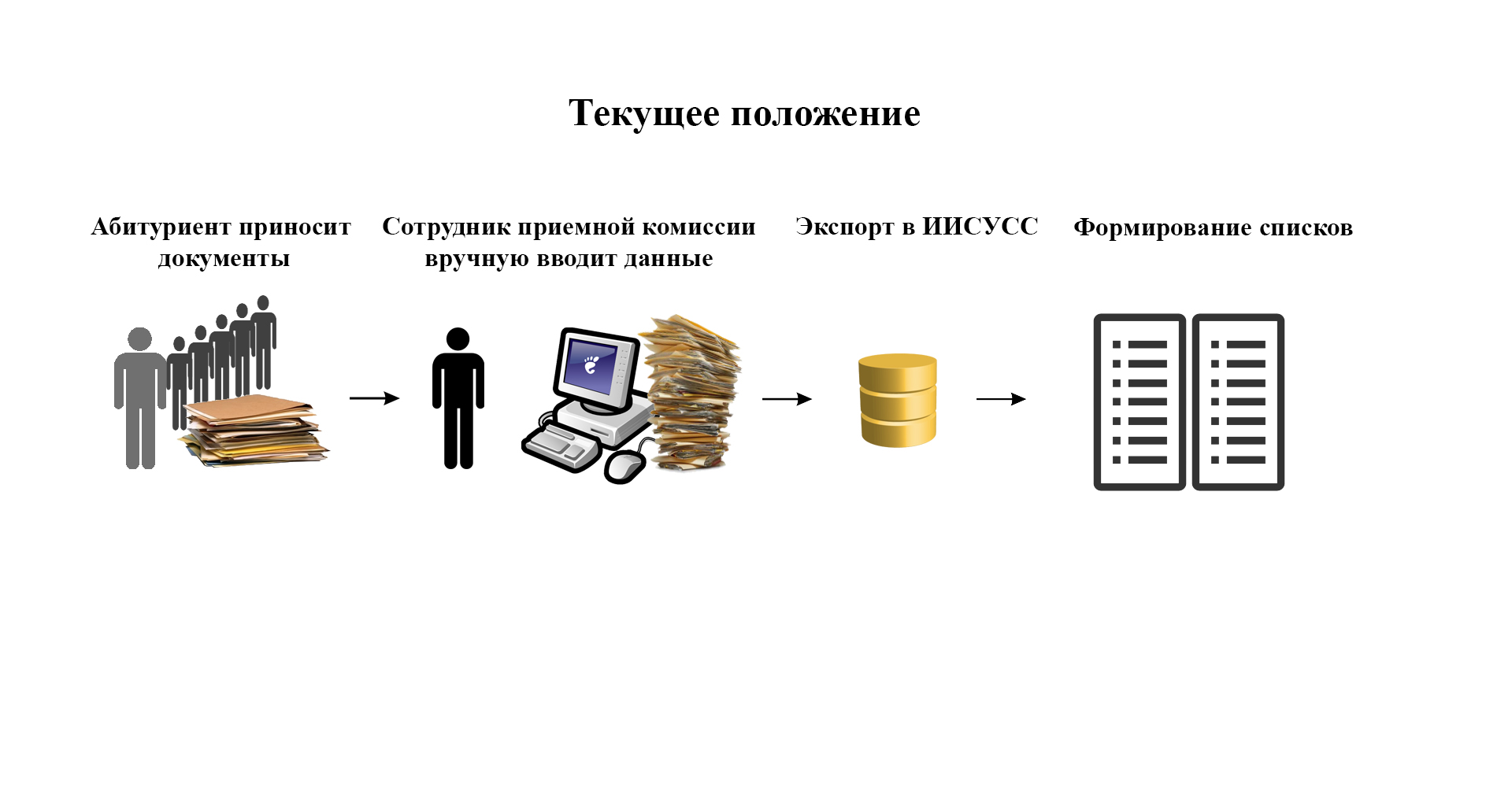


Рис.1. Текущий процесс подачи заявлений

Подводя итоги, заметим что, процесс сбора документов является самым ответственным процессом проведения приемной кампании университета, поэтому важно, чтобы сбор документов проходил с минимальным количеством ошибок и несоответствий. Очереди плохо отражаются на внимательности человека, вследствие чего повышается вероятность допущения ошибки при заполнении анкеты или при проверке документов по причине человеческого фактора. Автоматизация процесса сбора документов не только решит проблему с очередями, но и упростит работу приёмной комиссии, в частности, сократит объем ручной работы и повысит производительность работников (Рис. 1).

* 1. **Концепция сервиса**

Для решения проблемы, поставленной в пункте 1.1 данной работы, мы предлагаем создать web-портал, который будет объединять в себе несколько инструментов для автоматизации деятельности приемной комиссии университета (Рис. 2).

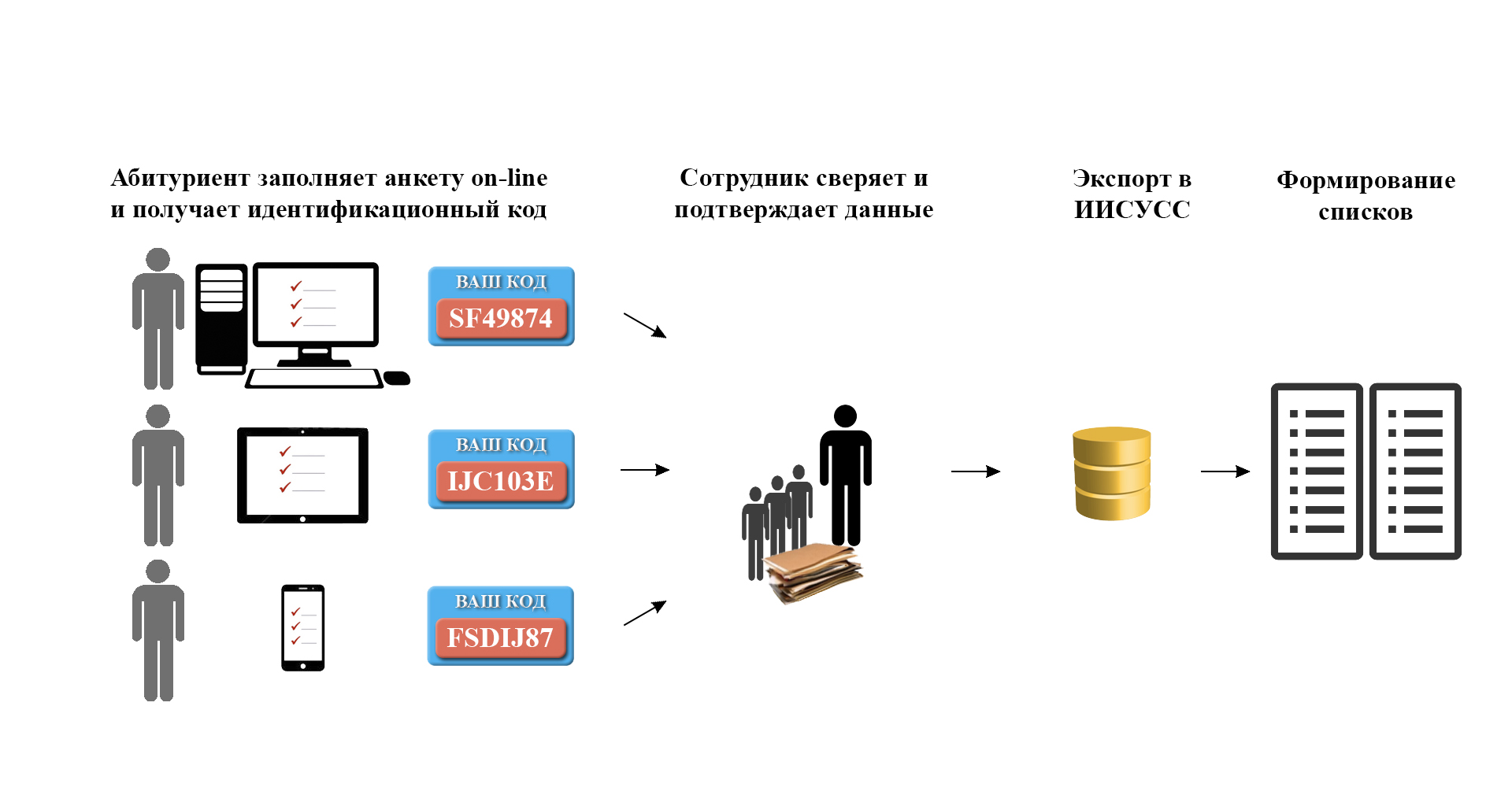


Рис.2. Концепция сервиса

Если на текущий момент все документы абитуриентов вводятся в систему учета студенческого состава ИИСУСС вручную одним, максимум двумя людьми в отборочной комиссии каждого подразделения университета, то данный проект предлагает ручной труд этих сотрудников делегировать самим абитуриентам. Тем самым увеличить эффективность работы приемной комиссии.

Разрабатываемый портал позволит свести алгоритм действий абитуриента к следующим шагам (Рис. 3):

1. Заходит на портал, затем регистрируется.
2. После чего ему необходимо заполнить необходимую информацию для поступления, после чего он сможет подать заявление.



Рис.3. Алгоритм действий абитуриента

1. После подачи заявления, у заявления появится регистрационный номер.
2. Отправляется с документами в приемную комиссию.
3. Предоставляет полученный ранее регистрационный номер, затем работник комиссии находит по этому номеру его заявление, после этого идет сверка данных указанных в анкете с оригиналами.
4. При успешной проверке данных, распечатывается анкета, в которой абитуриент ставит свою подпись в подтверждение того, что анкета составлена верно и соглашается на обработку данных.
5. Абитуриент получает расписку о сдаче документов.
6. Ожидание результатов.
   1. **Обзор аналогов**

Автоматизация процесса сбора документов далеко не новшество, в России уже существует несколько вариантов решения. Рассмотрим некоторые из них.

1. Предварительная регистрация абитуриента МГТУ им. Н.Э. Баумана, <http://priem.bmstu.ru/ru/registration/>

Данный сайт вполне рабочий, но имеются недостатки:

* Отсутствует выбор формы обучения.
* Отсутствует выбор приоритета при выборе специальности. Поступая в ВУЗ, абитуриент может выбрать 3 направления подготовки, при этом должен указываться приоритет профилей в порядке убывания, здесь же отсутствует выбор приоритета.
* Жесткий список результатов ЕГЭ. Имеется 6 полей для заполнения результатов ЕГЭ по каждой дисциплине, название дисциплины записано в сокращенном варианте.
* Отсутствие поля «Совпадает с адресом по прописке». При совпадении адресов по прописке и регистрации, придется указывать адрес дважды .

1. Анкета абитуриента Нижегородский Институт Управления, <http://niu.ranepa.ru/?p=25676>.

Недостатки:

* Некачественная верстка сайта. Поля ввода текста не вмещаются в отведенный для анкеты отдел, из-за чего появляется scroll-бар внизу страницы. Имеется пустое пространство между scroll-баром и кнопкой «Далее» (Рис. 4).
* Перезагрузка страницы при выборе уровня образования и указании приоритета направлений подготовки.
* Отсутствует автозаполнение полей, кроме «Имеющееся образования», «Уровень желаемого образования».
* После нажатия кнопки «Далее» ничего не происходит.

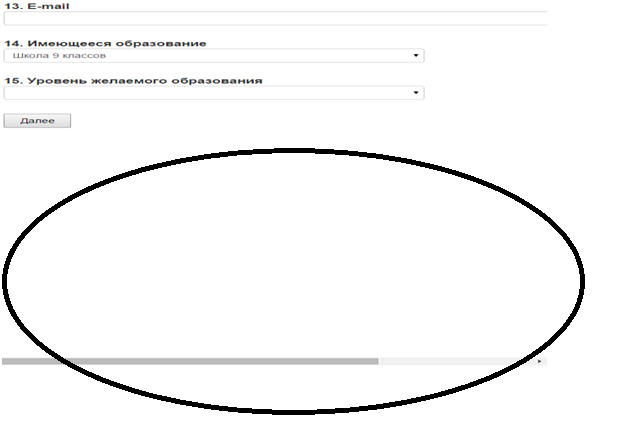


Рис.4. Страница подачи заявления

1. Личный кабинет абитуриента СВФУ. <http://priem.s-vfu.ru/epz/>

Согласно правилам приема в ФГАОУ ВПО СВФУ им. М.К. Аммосова, абитуриент может отправить необходимые документы для поступления в электронно-цифровом виде через личный кабинет абитуриента на официальном сайте СВФУ.

Зарегистрироваться в системе не представляется возможным по состоянию на 15.06.2015, поскольку необходимо активировать учетную запись, для этого на почту должно высылаться письмо со ссылкой на активацию, но письмо так и не пришло в течение 8 дней (Рис. 5).

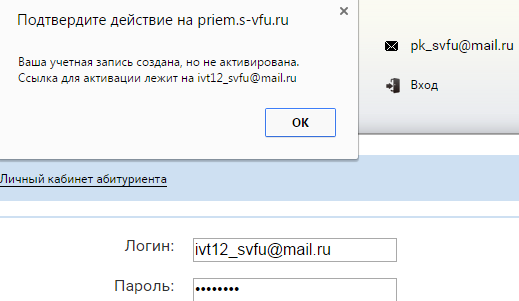


Рис.5. Страница авторизации на сайте priem.s-vfu.ru

На данный момент (31.05.2016) возможность входа отсутствует на странице сайта (Рис. 6).

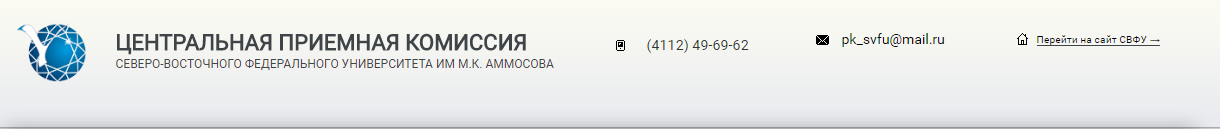


Рисунок 6 - Главная страница priem.s-vfu.ru

* 1. **Методы разработки ПО**

**Каскадная модель**

Каскадная модель — модель процесса разработки программного обеспечения, в которой процесс разработки выглядит как поток, последовательно проходящий фазы анализа требований, проектирования, реализации, тестирования, интеграции и поддержки. В качестве источника названия часто указывают статью, опубликованную У. У. Ройсом в 1970 году [6].

В каскадной модели Ройса, следующие фазы шли в таком порядке:

1. Определение требований
2. Проектирование
3. Конструирование (также «реализация» либо «кодирование»)
4. Воплощение
5. Тестирование и отладка (также «верификация»)
6. Инсталляция
7. Поддержка

Переход от одной фазы к другой происходит только после полного и успешного завершения предыдущей

Следуя каскадной модели, разработчик переходит от одной стадии к другой строго последовательно. Сначала полностью завершается этап «определение требований», в результате чего получается список требований к ПО. После того как требования полностью определены, происходит переход к проектированию, в ходе которого создаются документы, подробно описывающие для программистов способ и план реализации указанных требований. После того как проектирование полностью выполнено, программистами выполняется реализация полученного проекта. На следующей стадии процесса происходит интеграция отдельных компонентов, разрабатываемых различными командами программистов. После того как реализация и интеграция завершены, производится тестирование и отладка продукта; на этой стадии устраняются все недочёты, появившиеся на предыдущих стадиях разработки. После этого программный продукт внедряется и обеспечивается его поддержка — внесение новой функциональности и устранение ошибок [6].

Тем самым, каскадная модель подразумевает, что переход от одной фазы разработки к другой происходит только после полного и успешного завершения предыдущей фазы, и что переходов назад либо вперёд или перекрытия фаз не происходит [6].

Исходя из выше сказанного, напрашиваются следующие выводы:

* Данная методология подразумевает четкое формирование требований в начале разработки, а также отрицает возможность динамического изменения в процессе разработки.
* Невозможность предварительной оценки качества программного обеспечения пользователем.
* Для решения проблем, возникших во время разработки, необходимо возвращаться к предыдущим шагам модели, что приводит к увеличению затрат и нарушению графика работы.

Поскольку данный проект подразумевает работу с клиентом, а также трудность в составление требования ПО и ограниченность во времени, разумнее будет отказаться от данного метода разработки в пользу более гибких методов.

**Методология Agile**

Существуют также итеративные модели разработки, которые основываются на выполнение работ параллельно с непрерывным анализом полученных результатов и корректировкой предыдущих этапов работы. При этом в каждой фазе развития повторяется цикл Планирование – Реализация – Проверка – Оценка. В последние годы особую популярность получили так называемые гибкие методы разработки [7].

Гибкая методология разработки (agile-методы) — серия подходов к разработке программного обеспечения, ориентированных на использование итеративной разработки, динамическое формирование требований и обеспечение их реализации в результате постоянного взаимодействия внутри самоорганизующихся рабочих групп, состоящих из специалистов различного профиля. Методы agile не включают практик, а определяет ценности и принципы, которыми руководствуются успешные команды [7].

Основными идеями agile являются:

* Люди и взаимодействие важнее процессов и инструментов;
* Работающий продукт важнее исчерпывающей документации;
* Сотрудничество с заказчиком важнее согласования условий контракта;
* Готовность к изменениям важнее следования первоначальному плану;

Рассмотрим один из методов agile-методологии.

**FDD**

Feature driven development (FDD, разработка, управляемая функциональностью) — итеративная методология разработки программного обеспечения, одна из гибких методологий разработки (agile). FDD представляет собой попытку объединить наиболее признанные в индустрии разработки программного обеспечения методики, принимающие за основу важную для заказчика функциональность (свойства) разрабатываемого программного обеспечения. Основной целью данной методологии является разработка реального, работающего программного обеспечения систематически, в поставленные сроки [8].

FDD включает в себя пять базовых видов деятельности:

* разработка общей модели;
* составление списка необходимых функций системы;
* планирование работы над каждой функцией;
* проектирование функции;
* реализация функции.

Первые два процесса относятся к началу проекта. Последние три осуществляются для каждой функции. Разработчики в FDD делятся на «хозяев классов» и «главных программистов». Главные программисты привлекают хозяев задействованных классов к работе над очередным свойством. Работа над проектом предполагает частые сборки и делится на итерации, каждая из которых предполагает реализацию определенного набора функций [8].

* 1. **Обзор инструментов разработки web-приложений**

Разработка web-приложений является процессом сложным и требует от программиста знаний различных технологий, языков программирования и многого другого. Современный сайт представляет собой сочетание программ на различных языках программирования (как серверных, так и клиентских), фреймворков для этих языков, html – кода, а так же различных наборов css шаблонов для создания красочного дизайна. Рассмотрим языки программирования.

**PHP**

PHP — скриптовый язык программирования, применяемый для разработки веб-приложений. Поддерживается подавляющим большинством хостинг-провайдеров, является одним из лидеров среди языков программирования, применяемых для создания динамических веб-сайтов. По синтаксису язык очень близок к C*.*В сайтах на PHP часть кода, написанная на HTML, отвечает за внешний вид и вывод информации, а PHP в свою очередь за динамику и интерактивность функционала. Благодаря своей простоте, гибкости и скорости выполнения операций, язык PHP является одним из самых популярных стандартов, в веб-разработке [9].

В области веб-программирования, в частности серверная часть, PHP — один из популярных сценарных языков (наряду с JSP, Perl и языками, используемыми в ASP.NET) благодаря своей простоте, скорости выполнения, богатой функциональности, кроссплатформенности и распространению исходных кодов на основе лицензии PHP [10].

Преимущества [11]:

* легок в освоении на всех этапах;
* поддерживается большим сообществом пользователей и разработчиков;
* имеет развитую поддержку баз данных;
* имеется огромное количество библиотек и расширений языка;
* может использоваться в изолированной среде;
* предлагает нативные средства организации веб-сессий, программный интерфейс расширений;
* является довольно полной заменой проприетарной среды ASP (Active Server Pages) от Microsoft;
* может быть развёрнут почти на любом сервере;
* портирован под большое количество аппаратных платформ и операционных систем.

Недостатки [11]:

* веб-приложения, написанные на PHP, зачастую имеют проблемы с безопасностью;
* не подходит для создания десктопных приложений или системных компонентов;
* имеет слабые средства для работы с исключениями;
* глобальные параметры конфигурации влияют на базовый синтаксис языка, что затрудняет настройку сервера и разворачивание приложений;
* объекты передаются по значению, что смущает многих программистов, привыкших к передаче объектов по ссылке, как это делается в большинстве других языков.

**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 [12].

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) [12].

Преимущества [13]:

* 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.

Недостатки [13]:

* 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;
* 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.

**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. По особенностям синтаксиса он близок к языкам Perl и Eiffel, по объектно-ориентированному подходу — к Smalltalk. Также некоторые черты языка взяты из Python, Lisp, Dylan и Клу [14].

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 [14].

Преимущества [11]:

* 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 [12]:

* 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.

Для программирования динамического поведения страницы воспользуемся jQuery

**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 [15].

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 [15].

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 [15].

Далее рассмотрим web-фреймворки на рассмотренных языках программирования.

**Joomla!**

Joomla! — система управления содержимым (CMS), написанная на языках PHP и JavaScript, использующая в качестве хранилища базы данных СУБД MySQL или другие индустриально-стандартные реляционные СУБД [16].

CMS Joomla! включает в себя минимальный набор инструментов при начальной установке, который дополняется по мере необходимости. Это снижает загромождение административной панели ненужными элементами, а также снижает нагрузку на сервер и экономит место на хостинге [17].

Joomla! позволяет отображать интерфейс фронтальной и административной части на любом языке. Каталог расширений содержит множество языковых пакетов, которые устанавливаются штатными средствами администрирования [16].

**Ruby on Rails**

Ruby on Rail**s** — фреймворк, написанный на языке программирования Ruby. Ruby on Rails предоставляет архитектурный образец Model-View-Controller (модель-представление-контроллер) для веб-приложений, а также обеспечивает их интеграцию с веб-сервером и сервером базы данных [17].

Главной целью считается упрощение разработки веб-приложений и их создание в малом количестве кода, нежели в иных фреймворках, с минимальной конфигурацией. Метапрограммирование Ruby как раз таки позволяет достичь всего этого [18].

Основными принципами Ruby on Rails являются исключение конфигурирования, вводя, где это возможно, общепринятых соглашений, и исключение повторного кода, который несет одинаковый смысл. Компоненты Ruby on Rails интегрированы таким образом, что теперь программисту не надо писать дескрипторы, которые необходимы для связи между собой или повторять определения, находящиеся в программном коде в реляционной базе данных [18].

Сам же фреймворк Ruby on Rails и его дополнительные расширения распространяются посредством такой системы, как Ruby Gems, которая стандартизирует каналы распространения и формат пакетов [18].

**Django**

Django — свободный фреймворк для веб-приложений на языке Python, использующий шаблон проектирования MVC. Проект поддерживается организацией Django Software Foundation [19].

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) [19].

Также, в отличие от других фреймворков, обработчики URL в Django конфигурируются явно при помощи регулярных выражений, а не выводятся автоматически из структуры моделей контроллеров [19].

Для работы с базой данных Django использует собственный ORM, в котором модель данных описывается классами Python, и по ней генерируется схема базы данных.

Далее рассмотрим наборы инструментов для оформления дизайна сайта.

**Bootstrap**

Twitter Bootstrap — свободный набор инструментов для создания интерфейса сайтов и веб-приложений. Включает в себя CSS шаблоны оформления для типографики, веб-форм, кнопок, меток, блоков навигации и прочих компонентов веб-интерфейсов, включая JavaScript расширения[18].

Bootstrap использует самые современные наработки в области CSS и HTML, поэтому необходимо быть внимательным при поддержке старых браузеров [20].

Основные преимущества Twitter Bootstrap 3:

* 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;

**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 [21].

Возможности и преимущества:

* 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

**Выводы к главе 1**

В данной главе нами была сформулирована проблема организации приемной кампании СВФУ. Анализ аналогичных программных продуктов показал, что они не подходят под требования СВФУ. В итоге было решено разработать веб-портал для сбора и обработки документов абитуриентов. Для написания web-портала был проведен анализ методов разработки ПО, на сегодняшний день очень распространены методы на основе методологии Agile, которая в свою очередь очень подходит для корпоративных разработок. Затем был исследован текущий ассортимент инструментов разработки, среди прочих были рассмотрены такие как Python, PHP, Bootstrap.

**Глава 2. Разработка модуля «Личный кабинет абитуриента»**

* 1. **Требования к модулю**

Процесс разработки ПО начинается с выявления требований к ней. Для нашего модуля «Личный кабинет абитуриента» были получены следующие функциональные требования.

Модуль должен:

1. Предоставлять пользователю возможность зарегистрироваться на портале, где он должен ввести следующие данные:
   1. Логин;
   2. Пароль;
   3. Электронная почта;
   4. Captcha.
2. Предоставлять возможность для заполнения заявления. Для того чтобы заполнить заявление, пользователь должен заполнить профиль:
   1. Личные данные:
      1. ФИО;
      2. Дата рождения;
      3. Место рождения;
      4. Пол;
      5. Национальность;
      6. Гражданство;
      7. Контактный телефон;
      8. Военная обязанность;
      9. Изучаемый иностранный язык;
      10. СНИЛС;
      11. Нуждаемость в общежитие;
      12. Адресная информация:
          1. Тип адреса (по прописке/фактическое);
          2. Область;
          3. Район/улус;
          4. Город;
          5. Улица;
          6. Дома;
          7. Корпус;
          8. Квартира.
   2. Документы:
      1. Информация о документе, удостоверяющий личность:
         1. Вид;
         2. Серия;
         3. Номер;
         4. Дата выдачи;
         5. Кем выдан.
      2. Информация о документе, об образовании:
         1. Вид;
         2. Серия;
         3. Номер.
      3. Информация о предыдущем образовании:
         1. Дата поступления;
         2. Дата окончания;
         3. Название учебного заведения.
   3. Заявление:
      1. Подразделение университета;
      2. Направление/специальность;
      3. Профиль;
      4. Форма обучения;
      5. Уровень обучения.
   4. Экзамены:
      1. Результаты ЕГЭ;
         1. Название дисциплины;
         2. Балл;
         3. Год.
      2. Вступительные экзамены:
         1. Название дисциплины.
   5. Льготы и достижения:
      1. Льготы:
         1. Тип льготы;
         2. Категория;
         3. Документы.
      2. Достижения:
         1. Тип документа;
         2. Номер документа;
         3. Достижение.
3. Предоставлять возможность редактирования заявления;
4. Мониторинг «статуса» заявления.

Для удобства и уменьшения вероятности возникновения ошибок необходимо этот массив информации разбить на группы по смысловому значению и заполнение формы должно быть разбито на несколько независимых друг от друга этапов.

Для упрощения процесса заполнения данных в модуле должен иметься плагин для автозаполнения Select2. Также должна присутствовать валидация данных. При успешном заполнении данные должны заноситься в базу данных.

После подачи заявления, абитуриент должен получить регистрационный номер.

* 1. **Проектирование**

Ознакомившись с процессом подачи заявления, были выявлены классы, связанные с абитуриентом, университетом, подразделением университета и данными абитуриента.

Разработана модель классов UML, отражающая особенности предметной области. Она состоит из 25 классов, связанных между собой. Эти же классы используются при реализации моделей Django проекта. Рассмотрим классы модели (диаграмму модели вы можете просмотреть в приложении 1):

1. **AttrType**

Определяет тип Attribute. Содержит в себе name, что подразумевает название типа.

1. **Attribute**

Определяет атрибут. Содержит в себе атрибут name. Атрибут name определяет название атрибута.

1. **AttrValue**

Определяет значение атрибута. Содержит в себе атрибут value. Value определяет значение атрибута.

1. **Person**

Данный класс содержит в себе информацию о человеке. Состоит из атрибутов: lname, name, mname, fullname, sex, birthdate,. Атрибуты lname, name, mname определяет фамилию, имя и отчество человека, sex определяет пол, birthdate дату рождения.

1. **Abiturient**

Класс является наследником класса Person. Содержит информацию о абитуриенте, помимо атрибутов класса Person, состоит из: birthplace, hostel, nationality, citizenship, foreign\_lang, user, info\_progress. Birthplace определяет место рождения абитуриента, hostel необходимость в общежитии, nationality определяет национальность абитуриента, citizenship определяет гражданство, foreign\_lang определяет изучаемый иностранный язык, user содержит информацию о логине, пароле и электронной почте абитуриента, info\_progress определяет степень заполнения данных.

1. **Application**

Класс Application определяет информацию о заявлении. Состоит из атрибутов: date, number, budget, withfee. Date определяет дату подачи заявления, number определяет номер заявки в базе данных, budget определяет поступает ли абитуриент в рамках контрольных цифра приема, withfee определяет поступает ли абитуриент по договору об оказании платных услуг.

1. **Address**

Класс Address определяет адрес фактического проживания и адрес по прописке. Состоит из: zipcode, street, house, building, flat, adrs\_type\_is\_same. Zipcode определяет индекс, street определяет улицу, house определяет дом, building определяет корпус, flat определяет квартиру, adrs\_type\_is\_same определяет совпадает ли адрес по прописке с адресом фактического проживания.

1. **Contacts**

Определяет контактную информацию человека. Состоит из value. Value определяет значение контакта.

1. **DocAttr**

Определяет атрибут документа. Состоит из одного поля value, которое определяет значение атрибута.

1. **Docs**

Определяет документ человека. Состоит из: serialno, number, issueDate, isCopy. Serialno определяет серию документа, number определяет номер документа, issueDate опреляет дату выдачи документа, isCopy определяет предоставлен оригинал документа или его копия.

1. **Education**

Класс содержит информацию о предыдущем образовании абитуриента. Состоит из level и enterDate. Level определяет уровень предыдущего образования, enterDate определяет дату поступления.

1. **Exams**

Определяет экзамены. Состоит из: points, year. Points определяет количество баллов полученные по итогам экзамена, year определяет в каком год сдавался экзамен.

1. **Department**

Класс Department определяет институт/факультет. Состоит из name. Name определяет название института/факультета.

1. **University**

Класс University определяет университет. Состоит из university, который определяет название университета.

1. **Education\_Prog**

Класс education\_prog определяет направление подготовки. Состоит из name. Name определяет название направления подготовки.

1. **Education\_Prog\_Form.**

Класс определяет форму обучения для определенного направления подготовки. Состоит из eduform, который определяет форму обучения.

1. **Profile**

Класс Profile определяет профиль обучения. Состоит из name. Name определяет название профиля.

1. **ApplicationProfiles**

Класс определяет профиль обучения для заявления.

1. **NeedDocuments**

Класс определяет набор документов, дающих преимущество при поступлении, для каждого профиля обучения.

1. **Exams\_needed**

Класс Exams\_needed определяет необходимые экзамены для поступления и их минимальный порог. Состоит из min\_points. Min\_points определяет минимальное количество баллов для поступления.

1. **Privilegies**

Класс Privilegies предназначен организации связи между классами Docs и Person.

1. **Milit**

Класс Milit содержит информацию о воинской обязанности человека. Состоит из: liableForMilit,isServer,yearDismissial, rank. LiableForMilit определяет военнообязанн ли человек, isServer определяет отслужил ли человек, yearDismissial определяет год увольнения из рядов Российской Армии, rank определяет воинское звание.

1. **DepAchieves**

Класс DepAchieves определяет список достижений учитываемых в институте/факультете. Состои из: name, points. Атрибут name определяет наименование достижения, points определяет количество баллов.

1. **Achievements**

Класс содержит в себе информацию о достижениях абитуриента.

1. **Relation**

Данный класс определяет родственников абитуриента.

Для того чтобы попасть в личный кабинет абитуриенту необходимо либо зарегистрироваться, либо авторизоваться. В личном кабинете абитуриент может прогресс заполнения информации о себе, просматривать статус своих заявлений, а также воспользоваться инструкцией в случае необходимости. На странице «Личные данные» абитуриент должен заполнить всю необходимую информацию о себе, также может редактировать ее в случае необходимости. После заполнения всей необходимой информации он сможет подать заявление, для этого необходимо перейти на страницу «Заявления». Также при необходимости абитуриент может изменить электронную почту и пароль, которые указывались при регистрации, для этого необходимо перейти на страницу «Учетная запись».

Примерный макет сайта выглядит следующим образом (Рис. 7):

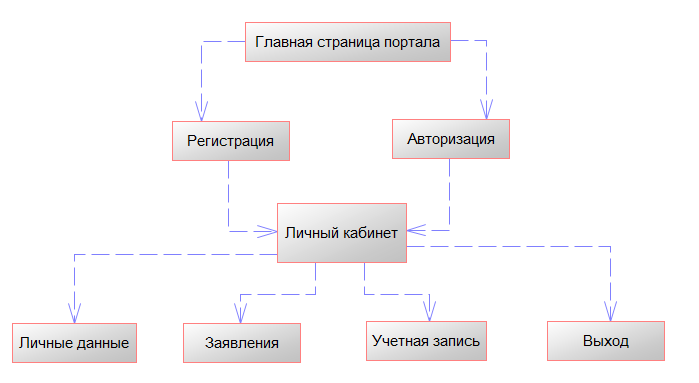


Рис.7. Макет сайта "Электронная анкета абитуриента"

В связи со значительным объемом информации, которую абитуриент должен предоставить приемной комиссии, для поступления, размер страницы личных данных получится большим. Физически человек не может оперировать параллельно более чем 7 объектами [22]. Следовательно, при организации диалога с пользователем необходимо разбить вводимую информацию на группы. Для достижения этой цели реализуем ввод информации в качестве, так называемого, мастера (wizard). Мастер – это инструмент повышения автоматизации выполнения пошаговых операций.

Примерный макет страницы «Личные данные» выглядит следующим образом. Процесс подачи заявления разобьется на 6 вкладок (шагов). Каждая из этих вкладок будет содержать информацию, касающуюся отдельных групп данных абитуриента (Рис. 8).

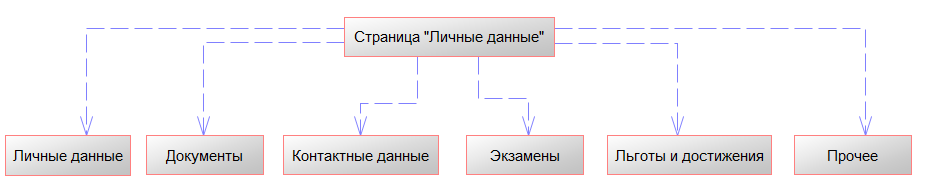


Рис.8. Макет страницы "Личные данные"

Подробная информация:

* **Личные данные.**

Содержит в себе информацию о человеке. Здесь указывается фамилия, имя, отчество, дата рождения, место рождения, национальность (указывается по желанию), гражданство.

* **Документы.**

Содержит в себе информацию о документах, необходимых для поступления. Здесь указывается вид документа, удостоверяющего личность, серия, номер, дата выдачи, кем выдан, вид документа об образовании, серия, номер, предыдущие образование, дата поступления, дата окончания, название учебного заведения, снилс.

* **Контактные данные.**

Содержит в себе контактные данные абитуриента. Указывается адресная информация, т.е. место прописки и место фактического проживания. Здесь указывается тип адреса, индекс, область, район/улус, город/село, улица, дом, корпус, квартира, совпадает ли адрес прописки совпадает с адресом факт. места жительства. Также во вкладке указывается контакты абитуриента, т.е. тип контакта и его номер. Дополнительно указываются родственники, т.е. кем приходится, ФИО и номер телефона.

* **Экзамены**

Содержит информацию о необходимых для поступления экзаменах. Состоит из двух таблиц. В первую таблицу добавляется информация о результатах ЕГЭ, во вторую таблицу добавляется информация о вступительных экзаменах.

* **Льготы и достижения**

Содержит информацию о льготах и достижения абитуриента. Здесь указывается категория, тип льготы и наименование достижения, подтверждающий документ.

* **Прочее**

Здесь указывается: изучаемый иностранный язык, нуждается ли абитуриент в общежитие, является ли он военнообязанным, служил ли в армии, год увольнения из рядов Российской Армии, воинское звание.

* 1. **Выбор инструментов разработки**

В результате анализа положительных и отрицательных сторон языков программирования, используемых для создания web приложений, проделанного в пункте 1.5, в качестве языка программирования проекта будет использоваться Python, поскольку он довольно прост в изучении.

Поскольку в качестве языка программирования будет использоваться Python, то в качестве web-фреймворка воспользуемся Django.

Для создания дизайна будет использоваться набор шаблонов Bootstrap, поскольку шаблоны выглядит гармонично и просты в использовании.

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

Дополнительным инструментом разработки является – Select2. Select2– плагин написанный на javaScript для автозаполнения текстового поля. Данный плагин очень пригодится, поскольку в модуле имеется огромное количество текстовых полей, которые необходимо заполнить типовыми данными.

* 1. **Описание ПО**

При заходе на портал абитуриенту необходимо зарегистрироваться, либо при наличии действующей учетной записи, авторизоваться на портале.

Страница регистрации оформлена в виде wizard’a и состоит из 3-х шагов (Рис. 9).



Рис.9. Страница регистрации

Для навигационного меню вкладок и прогресс бара использовался следующий код:

<div class="navbar">

<div class="navbar-inner">

<div class="container col-md-offset-2">

<ul>

<li><a href="#tab1" data-toggle="tab">Создание пользователя</a></li>

<li><a href="#tab2" data-toggle="tab">Личные данные</a></li>

<li><a href="#tab3" data-toggle="tab">Завершить</a></li>

</ul>

</div>

</div>

</div>

<div class="progress">

<div id="progress-bar" class="progress-bar progress-bar-striped active" role="progressbar" style=""></div>

</div>

Классы navbar, navbar-inner, container были описаны в Bootstrap’е. Они отвечают за дизайн навигационного меню. Каждое название вкладок является ссылкой на определенной tab. К примеру, «Создание пользователя» является ссылкой к классу #tab1, в свою очередь этот класс содержит в себе весь контент страницы.

Отрывок кода определяющего класс tab1:

<div class="tab-pane" id="tab1">

<div class="container-fluid">

<div class="row" >

<div class="col-xs-12 col-sm-12 col-md-4 col-md-offset-4" id ="usernameField">

<p>

<label class="control-label" for="username">Имя пользователя</label>

<input type="text" class="form-control" name="username" id="username" placeholder="Имя пользователя">

<span id="usernameIcon"></span>

<span id="usernameFieldDescr" class="help-block"></span>

</p>

</div>

</div>

Классы row, container-fluid, col-xs-12 col-sm-12 col-md-4 col-md-offset-4 описаны в Bootstrap. При вызове класса row содержимое этого класса отображаться на отдельной строке. Класс col-xs-12 col-sm-12 col-md-4 col-md-offset-4 определяет макет разметки строки и отступ от левого края. Тэг label предназначен для определения метки для элемента ввода, коим является “sname” .

Для заполнения поля дата рождения используется плагин datetimepicker (Рис. 10).

Скрипт инициализации плагина для поля birthday.

$("#birthday").datetimepicker({

format:'DD/MM/YYYY',

locale:"ru"

});

Отрывок html-кода:

<label class="control-label" for="birthday">Дата рождения</label>

<div class='input-group date' id='birthday'>

<input type="text" class="form-control" name="birthday" />

<span class="input-group-addon">

<span class="glyphicon glyphicon-calendar"></span>

</span>

</div>

<span id="birthdayIcon"></span>

<span id="birthdayFieldDescr" class="help-block"></span>

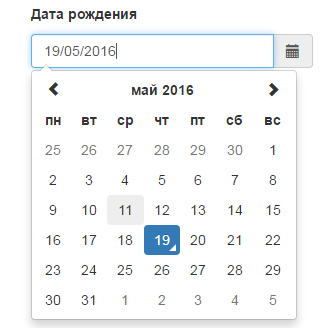


Рис.10. Пример использования DateTimePicker

Также для защиты от ботов, на портале используется captcha. Для создание капчи используется плагин RealPersonCaptcha.

Код инициализации captch’и:

$(function()

{

$('#captcha').realperson(

{

regenerate: 'Чтобы обновить captcha нажмите на неё',

});

});

Html-код:

<div class="row">

<div class="col-xs-12 col-sm-12 col-md-4 col-md-offset-4" id="captchaField">

<p>

<input type="text" id="captcha" name="captcha">

<input type="hidden" id="captchaHash" name="captchaHash">

</p>

</div>

</div>

Скрытое поле “captchaHash” используется для хранения hash-кода капчи. Капча проверяется следующим образом (Рис. 11):

def rpHash(person):

hash = 5381

value = person.upper()

for caracter in value:

hash = (( np.left\_shift(hash, 5) + hash) + ord(caracter))

hash = np.int32(hash)

return hash

If (rpHash(request.POST.get('captcha','')) == int(request.POST.get('captchaHash',''))):

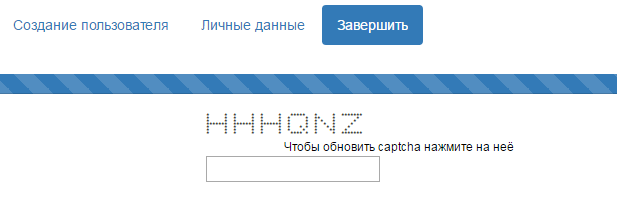


Рис.11. Captcha RealPerson

Введенное значение капчи абитуриентом конвертируется в hash-код и сравнивается с hash-кодом на странице.

Если абитуриент уже имеет учетную запись, то он может авторизоваться (Рис. 12).

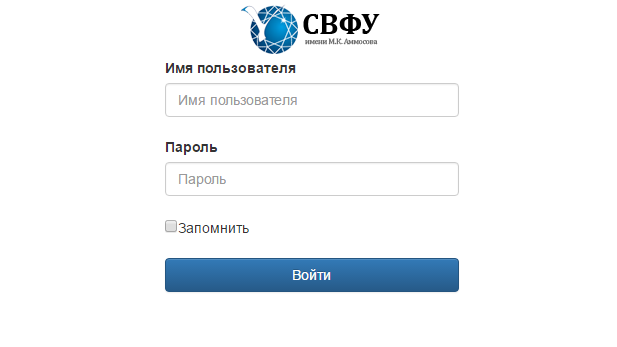


Рис.12. Страница авторизации

После успешной регистрации, либо после авторизации, абитуриент попадает в личный кабинет, который выглядит следующим образом (Рис. 13):

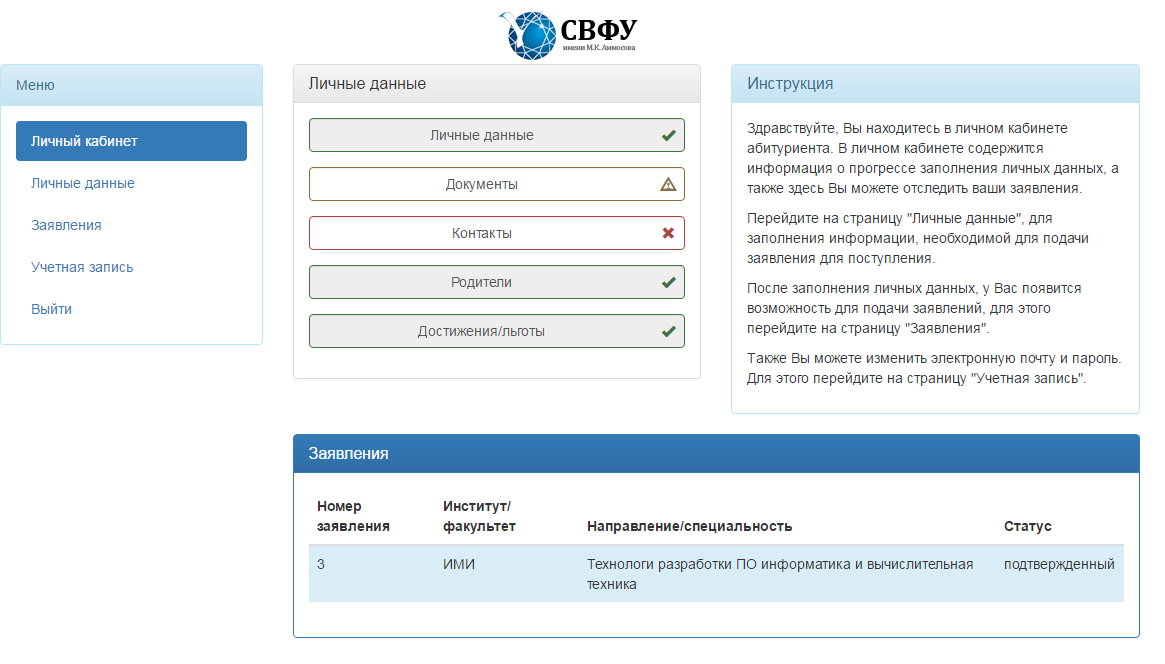


Рис.13. Страница "Личный кабинет"

Страница разбита на три блока, не считая меню. Код блока «Заявления».

<div class="panel panel-primary">

<div class="panel-heading">

<h3 class="panel-title">Заявления</h3>

</div>

<div class="panel-body">

{% if applications %}

<table class="table">

<thead>

<tr>

<th>Номер заявления</th>

<th>Институт/факультет</th>

<th>Направление/специальность</th>

<th>Статус</th>

</tr>

</thead>

<tbody>

{% for item in applications %}

<tr {% if item.appState.value == "Поданно" %} class="info" {% else %} {% if item.appState.value == "Экспортированно" %} class="success" {% else %} class="info" {% endif %} {% endif %}>

<td>{{item.id}}</td>

<td>{{item.department}}</td>

<td>{{item.edu\_prog.edu\_prog}}</td>

<td>{{item.appState.value}}</td>

</tr>

{% endfor %}

</tbody>

</table>

{% else %}

У вас пока нет заявлений :(

{% endif %}

</div>

</div>

Если у абитуриента уже имеются поданные заявления, то с сервера отправляется список заявлений Applications. Список содержит в себе информацию о статусе заявления, институт/факультет, направление подготовки, а также регистрационный номер заявления. Вся эта информация формируется в виде таблицы.

В случае отсутствия заявлений, таблица не создается, вместо нее появляется сообщение об отсутствие поданных заявлений.

Серверный код:

@login\_required(login\_url='authapp:index')

def PersonProfile(request):

person=Abiturient.objects.filter(user=request.user).first()

if person is None:

return redirect('/staff')

args={'currentpage':1}

applications=Application.objects.filter(abiturient\_\_user=request.user)

if applications is not None:

args['applications']=applications

args.update(csrf(request))

return render(request, 'anketa/profile.html', args)

Функция login\_required означает, что на страницу можно попасть только авторизованным пользователям, в случае если пользователь не авторизован, то происходит переадресация на страницу авторизации.

Abiturient.objects.filter(user=request.user) в данной строке кода из базы данных вытягивается первый объект из класс Abiturient, у которого поле user совпадает с авторизованным пользователем. После получения объекта класса Abiturient, из класса Applications вытягивается объекты в котором указан данный abiturient.

applications=Application.objects.filter(abiturient\_\_user=request.user)

if applications is not None:

args['applications']=applications

При наличии заявлений, на сервер отправляется список объектов applications.

HTML-код левого меню выглядит следующим образом:

<div class="col-xs-6 col-sm-3 sidebar-offcanvas" id="sidebar">

<div class="panel panel-info">

<div class="panel-heading">Меню</div>

<div class="panel-body">

<ul class="nav nav-pills nav-stacked">

<li {% if currentpage == 1 %} class="active"> <a href="#"> {% else %}> <a href="{% url 'profile' %}"> {% endif %}Личный кабинет</a></li>

<li {% if currentpage == 2 %} class="active"> <a href="#"> {% else %}> <a href="{% url 'persondata' %}"> {% endif %}Личные данные</a></li>

<li {% if currentpage == 3 %} class="active"> <a href="#"> {% else %}> <a href="{% url 'applicationList' %}"> {% endif %}Заявления</a></li>

<li {% if currentpage == 4 %} class="active"> <a href="#"> {% else %}> <a href="{% url 'account' %}"> {% endif %}Учетная запись</a></li>

<li><a href="{% url 'authapp:logout' %}">Выйти</a></li>

</ul>

</div>

</div>

</div>

Переменная currentpage передается со стороны сервера, если переменная совпадает с указанным числом, то элементу списку приписывается class=”active”, которое обводит указанную страницу в навигационном меню.

Для создания страницы «Личные данные» помимо datetimepicker’а использовался плагин Select2. Плагин Select2 использовался для автозаполнения полей. Некоторые поля должны заполняться справочными данными, которые должны быть одинаковыми. Для обеспечения этого требования мы используем автозаполнение этого поля. Как только пользователь начнет вводить информацию, на сервер будет отсылаться строка запроса. После этого на сервере происходит выборка данных, т.е. ищутся данные, которые имеют совпадения со строкой запроса, если совпадения найдены, формируется список, который будет отправляться клиенту. Эти данные будут выводиться на экран в виде выпадающего списка, из которого пользователь и будет выбирать необходимый ему вариант.

В качестве примера рассмотрим поле «Гражданство»:

$("#citizenship").select2({

placeholder: "РФ",

ajax:{

url:{% url 'citizenship' %},

dataType:'json',

delay:250,

data:function(params){

return{

query:params

};

},

processResults:function(data,page)

{

return{

results:data

};

},

cache:true

}

});

Далее, настраиваются свойства ajax-запроса, в скобках описывается его опции: здесь url является ссылкой на страничку citizenship, которая отвечает за выборку данных из базы, dataType определяет тип возвращаемого файла, delay время ожидание ответа, функция data определяет исходный данные отправляемые на сервер, minimumInputLength определяет минимальную длину текста, после которого срабатывает ajax, функция processResults преобразует ответные данные сервера.

Как говорилось ранее, ссылкой на базу данных является страница, которая в свою очередь описывается в urls.

Отрывок из urls.py:

url(r'^citizenship/$', Citizenship, name = 'citizenship')

Обработку данных отправленных на страницу citizenship будет выполняться функцией Citizenship, которая описана в файле views.py. Она и будет осуществлять выборку данных.

Код функции Citizenship:

def Citizenship(request):

trry = AttrValue.objects.filter(attribute\_\_name\_\_icontains = u'гражданство')

part = request.GET.get('query','')

if len(part)>0:

trry = trry.filter(value\_\_icontains = part)

trry = trry.values('id', 'value')

result = []

for item in trry:

result.append({'id':item['id'], 'text':item['value']})

return HttpResponse(json.dumps(result), content\_type="application/json")

Массив trry содержит в себе значения AttrValue, имя атрибута которых содержит в себе слово «Гражданство».

Элементы массива записываются в список объектов в формате {id, text}. Результаты возвращаются в json-формате.

При успешном выполнении html страница примет следующий вид (Рис. 14):

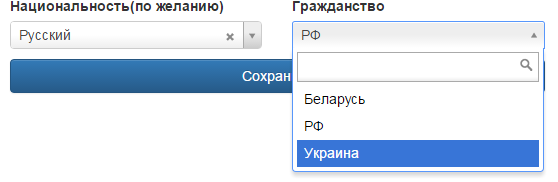


Рис.14. Использование Select2 для указания гражданства

Плагин Select2 используется для заполнения многих полей, к примеру: область/край/респ, район, город, улица, национальность, гражданство, наименование органа выдавшего документ, наименование учебного заведения, институт/факультет, направление/специальность, профиль, тип льготы, категория льготы, воинское звание.

Также для заполнения некоторых данных пригодится динамическое добавление полей. К примеру, при указании результатов ЕГЭ будут динамически добавляться поля для заполнения таблицы баллов ЕГЭ.

Отрывок html-кода:

<table class="table" id="table">

<thead id="tablehead">

<tr>

<th>Дисциплина</th>

<th>Баллы</th>

<th>Год</th>

<th>Действие</th>

</tr>

</thead>

<tbody id="egeTableBody">

{% if exams %}

{% for item in exams %}

<tr class="info">

<td><input type="hidden" class="form-control" name="egeDisc" value ={{item.subject}}></td>

<td><input type="text" class="form-control" name="egePoints" value ={{item.points}}></td>

<td><input type="text" class="form-control" name="egeYear" value ={{item.year}}></td>

<td><button class="btn btn-default btn-sm" id="delExamRow" type="button">Удалить</button></td>

</tr>

{% endfor %}

{% else %}

<tr class="info">

<td><input type="hidden" class="form-control" name="egeDisc"></td>

<td><input type="text" class="form-control" name="egePoints"></td>

<td><input type="text" class="form-control" name="egeYear"></td>

<td><button class="btn btn-default btn-sm" id="delExamRow" type="button">Удалить</button></td>

</tr>

{% endif %}

</tbody>

</table>

</div>

<div class="row">

<button class="btn btn-primary btn-block" type="button" id="addExam">Добавить</button>

</div>

Для того чтобы добавить строку, необходимо кликнуть на кнопку «Добавить», после этого добавится еще одно поле. Для удаление какой-либо на строки необходимо кликнуть на кнопку «Удалить», после этого удалиться строка где была нажата кнопка (Рис. 15).

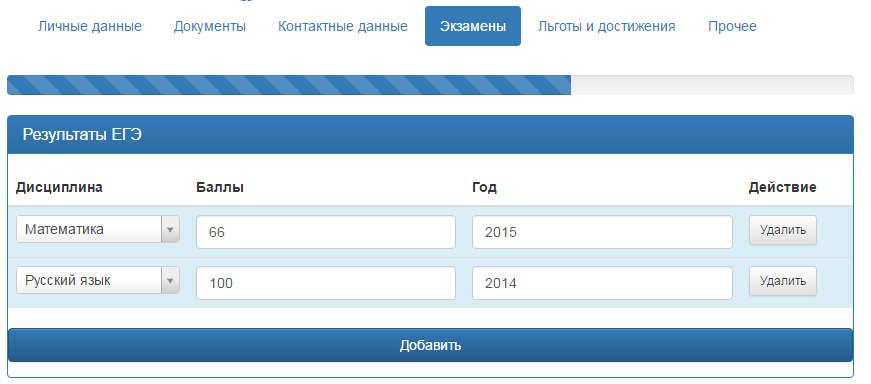


Рис.15. Отображение указанных ранее экзаменов на страницу

Для добавления поля используется следующий скрипт:

<script>

$("#addExam").on("click", function()

{

if(egerows<maxExams)

{

egerows++;

var newRow = $('<tr><td><input type="hidden" class="form-control" name="egeDisc"></td><td><input type="text" class="form-control" name="egePoints"></td><td><input type="text" class="form-control" name="egeYear"></td><td><button class="btn btn-default btn-sm" id="delExamRow" type="button">Удалить</button></td></tr>');

$("#egeTableBody").append(newRow);

initSelect2(newRow.find('input[name^="egeDisc"]'));

}

});

</script>

Если число строк меньше максимально допустимого, то создается новый объект, содержащий в себе HTML-код, после к таблице egeTableBody в конец добавляется данный объект (Рис. 16).

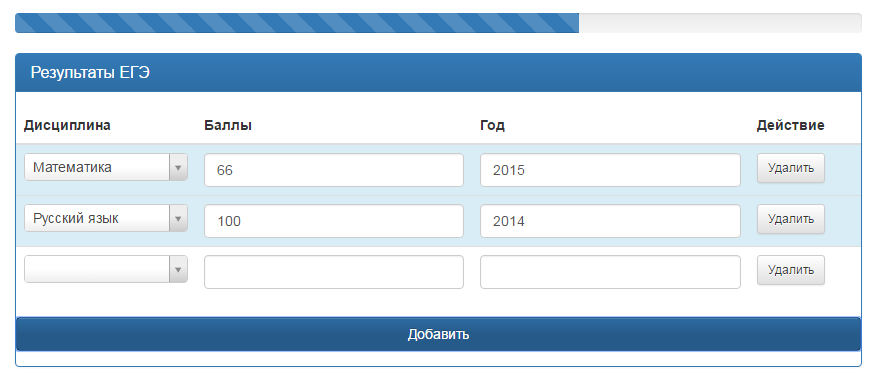


Рис.16. Добавление поля для экзаменов

Для удаления поля используется следующий скрипт:

$(document).on("click","button[id\*='delExamRow']", function()

{

if(egerows>1)

{

$(this).parent().parent().remove();

egerows--;

}

});

Данная функция срабатывает для всех кнопок документа, где id кнопки содержит в себе строку «delExamRow». Если количество строк больше 1, то строка удаляется путем определения родителя и последующим удалением его (Рис. 17).

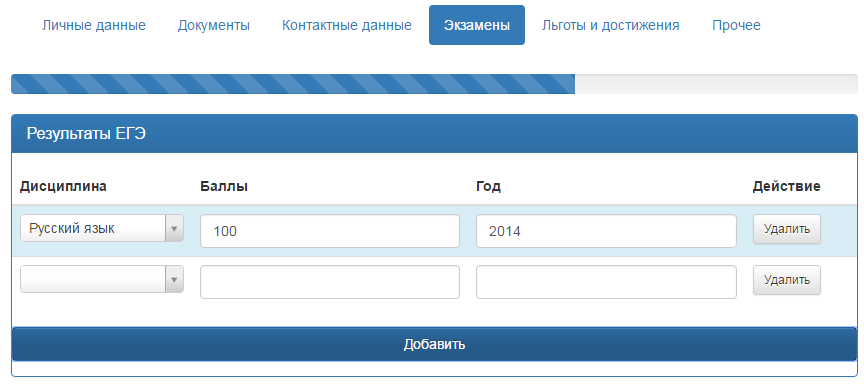


Рис.17. Удаление поля

Функция динамического добавления полей также будет использоваться для указания контактов и родственников абитуриента (Рис. 18).

Также плагин Select2 позволяет указывать в одном поле несколько объектов. К примеру, данная опция используется для указания дополнительных экзаменов для поступления.

Код на стороне сервера полностью идентичен, различие только в коде скрипта, в коде скрипта добавляется поле multiple.

$('#additionalExams').select2({

language:"ru",

multiple:true,

ajax:{

url:{% url 'examsubject' %},

dataType:'json',

delay:250,

data:function(params){return{query:params,

id:$("#eduname").select2('data').id};},

processResults:function(data,page)

{return{results:data};},

cache:true

}

});

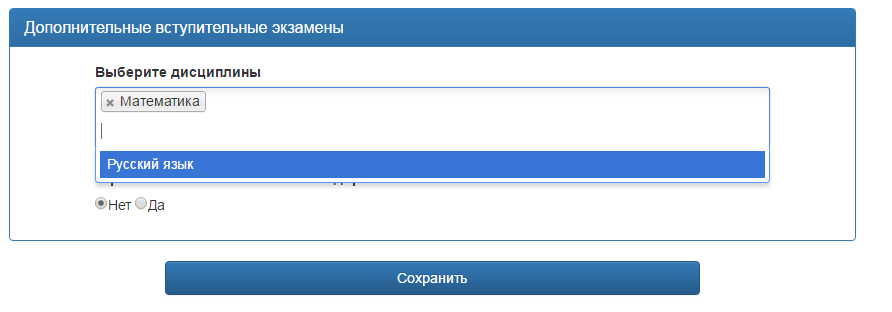


Рис.18. Использование мульти-select для указания дополнительных вступительных экзаменов

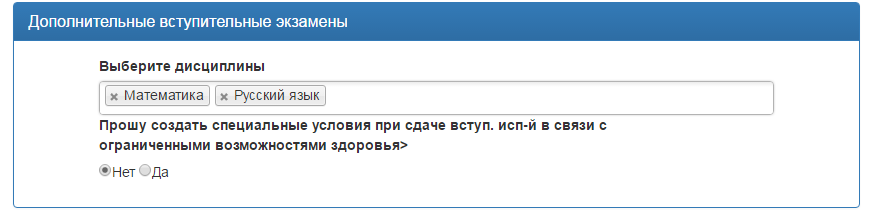


Рис.19. Мульти-select

Для сохранения данных в каждой вкладке имеется кнопка «Сохранить». При нажатии кнопки происходит ajax-запрос на сервер, с указанием текущей страницы для сохранения.

$(document).on("click","button[id\*='save']", function()

{$("#currentPage").val(($(this).attr('id').slice('4')));

$.ajax({

url:{% url 'add\_data\_to\_person' %},

type:$("#persondata").attr('method'),

data:$("#persondata").serialize(),

dataProcess:true,

timeout:500,

success:function(data)

{if (data==="success") {$.notify("Сохранено.", "success");}

else{$.notify("Что-то пошло не так :(", "error");}},

error:function(){$.notify("Отсутствует соединение :(", "error");}});});

Для оповещения о результатах сохранения используется плагин notify. К примеру, при успешном выполнении ajax-запроса, проверяется информация, которая пришла с сервера, при успешном сохранение возвращается строка “success”, после чего происходит оповещение об успешном сохранение информации.

{$.notify("Сохранено.", "success");}

Оповещение появляется в верхнем правом углу страницы (Рис. 20).

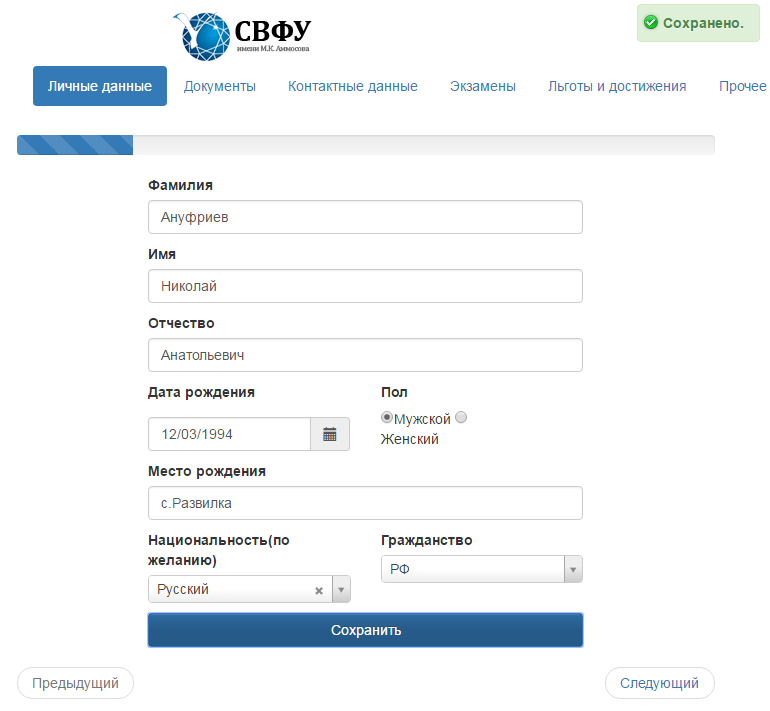


Рис.20. Пример оповещения после нажатия кнопки "Сохранить"

Код сохранения информации на сервере для первой вкладки:

def AddDataToPerson(request):

result="success"

if request.method == 'POST':

try:

page=int(request.POST.get('currentPage',''))

abit=Abiturient.objects.get(user=request.user)

#progress = abit.info\_progress.split()

if page==1: #Личные данные

#pageIsComplete=True;

abit.sname=request.POST.get('sname','')

abit.fname=request.POST.get('name','')

abit.mname=request.POST.get('mname','')

abit.birthplace=request.POST.get('birthplace','')

if(len(request.POST.get('birthday','')))>0:

abit.birthdate=datetime.datetime.strptime(request.POST.get('birthday',''),'%d/%m/%Y').strftime('%Y-%m-%d')

if(len(request.POST.get('nation','')))>0:

abit.nationality=AttrValue.objects.get(pk=request.POST.get('nation',''))

if(len(request.POST.get('citizenship','')))>0:

abit.citizenship=AttrValue.objects.get(pk=request.POST.get('citizenship',''))

if(len(request.POST.get('sex','')))>0:

abit.sex=request.POST.get('sex','')

Здесь проверяется длина значений поля, если длина больше 0, то значение поля сохраняется в базе данных. Для полей, которые инициализированы как Select2 объект, на сервер приходит идентификатор объекта в базе, после чего сохраняется объект класса AttrValue с указанным идентификатором.

В конечном итоге, после заполнения всех вкладок, страница выглядит следующим образом (Приложение 2).

На странице «Заявления», помимо возможности мониторинга статуса, необходимо реализовать возможность добавления, удаления и редактирования заявлений. Для этого необходимо вытягивать из базы данных информацию о заявление по идентификатору. Для этого необходимо написать ajax-функцию:

def GetSelectedApplication(request):

result={}

application =Application.objects.get(pk = request.GET.get('id',''))

app\_profiles = ApplicationProfiles.objects.filter(application = application)

result['department\_id']=application.department.id

result['department\_name']=application.department.name

result['edu\_prog\_id']=application.edu\_prog.edu\_prog.id

result['edu\_prog\_name']=application.edu\_prog.edu\_prog.name+' ' + application.edu\_prog.edu\_prog.qualification.value

result['edu\_prog\_eduform\_id']=application.edu\_prog.id

result['edu\_prog\_eduform\_name']=[x[1] for x in EduForm if x[0] == application.edu\_prog.eduform][0]

profiles=[]

for item in app\_profiles:

profiles.append({'id':item.profile.id,'profile':item.profile.name})

result['profiles']=profiles

result['profiles\_len']=len(profiles)

return HttpResponse(json.dumps(result), content\_type="application/json")

Далее, необходимо загрузить полученные данные в select2, для этого преобразуем success функцию ajax’a:

success:function(data){

var list = new Array();

for(var i=0;i<parseInt(data['profiles\_len']);i++){

list[i]={"id":data['profiles'][i]['id'],"text":data['profiles'][i]['profile']};

}

$("#eduname").select2('enable');

$("#eduprof").select2('enable');

$("#eduprior").select2('enable');

$("#eduform").select2('enable');

$("#department").select2('data',{id: data['department\_id'], text:data['department\_name']});

$("#eduname").select2('data',{id: data['edu\_prog\_id'], text:data['edu\_prog\_name']});

$("#eduform").select2('data',{id: data['edu\_prog\_eduform\_id'], text:data['edu\_prog\_eduform\_name']});

$("#eduprof").select2('data',list);

},

Массив используется для того, чтобы записать значения, полученные с сервера, в мульти-select поле «Профили».

Для удаления заявлений создадим функцию DeleteApplication:

def DeleteApplication(request):

result = {'result':0, 'error\_msg':''}

if request.method == 'GET':

Application.objects.get(pk=request.GET.get('id')).delete()

return HttpResponse(json.dumps(result), content\_type="application/json")

На сервер передается идентфикатор заявления, который необходимо удалить, по идентификатору находится заявление в базе и удаляется функцией delete(), после чего происходит перезагрузка страницы.

Для редактирования и добавления заявления воспользуемся функцией SaveApplication:

def SaveApplication(request):

result = {'result':0, 'error\_msg':''}

#print(request.POST)

if request.method == 'POST':

if(int(request.POST.get('facepalm',''))>0):

application=Application.objects.get(pk=request.POST.get('facepalm'))

ApplicationProfiles.objects.filter(application=application).delete()

else:

application = Application()

application.abiturient=Abiturient.objects.get(user=request.user)

application.department = Department.objects.get(pk = request.POST.get('department',''))

application.date = datetime.datetime.strftime(datetime.datetime.now(), '%Y-%m-%d')

application.edu\_prog=Education\_Prog\_Form.objects.get(pk=request.POST.get('eduform',''))

application.eduform = application.edu\_prog.eduform

application.budget = True

application.withfee = False

kaketomenyabesit = AttrValue.objects.filter(attribute\_\_name\_\_icontains=u'Статус заявления').filter(value\_\_icontains=u'Подан').first()

application.appState = kaketomenyabesit

application.points =100

application.save()

if (len(request.POST.get('eduprof'))>0):

profs=request.POST.get('eduprof','').split(',')

for item in profs:

appProf = ApplicationProfiles()

appProf.application=application

appProf.profile=Profile.objects.get(pk=item)

appProf.save()

return HttpResponse(json.dumps(result), content\_type="application/json")

Поле «facepalm» необходимо для того, чтобы определять создается, либо редактируется заявление. Если редактируется, то происходит поиск в базе данных необходимого заявления, при этом удаляются все объекты ApplicationProfiles, которые привязаны к этому заявлению. В ином случае, создается объект класса Application. Далее, внесенные данные сохраняются в базу данных, заявление получает статус «Поданный» и происходит перезагрузка страницы (Рис. 21).

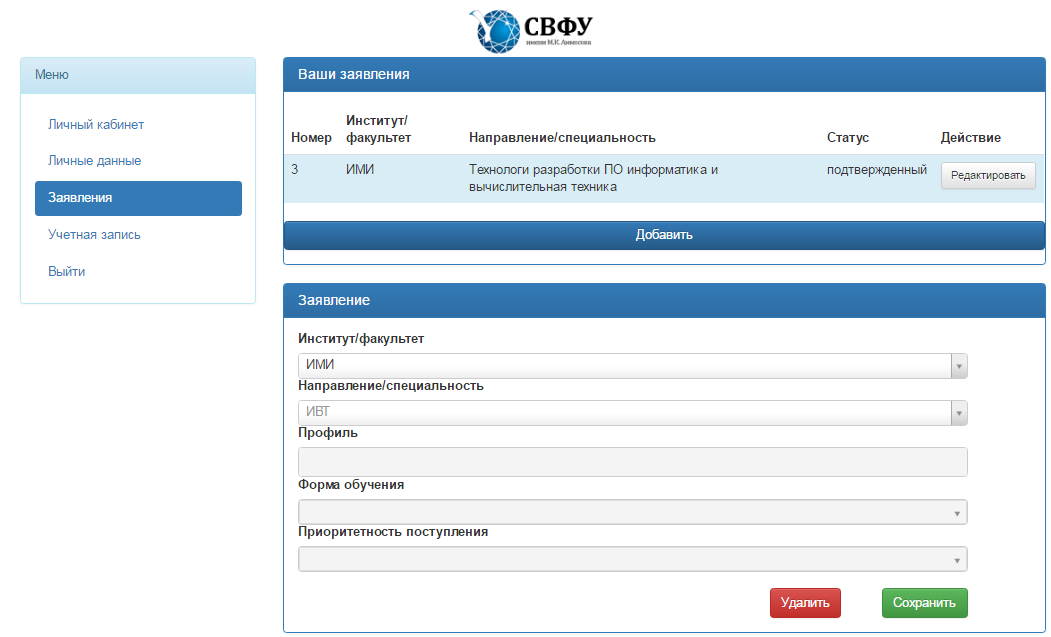


Рис.21. Страница "Заявления"

Также необходимо реализовать возможность изменения электронной почты и пароля. Для начала на странице «Учетная запись» необходимо выводить текущую электронную почту, для этого с базы необходимо вытянуть значение почты:

@login\_required(login\_url='authapp:index')

def Account(request):

args={'currentpage':4}

person=Abiturient.objects.filter(user=request.user).first()

if person is None:

return redirect('/staff')

args['email'] = request.user.email

args.update(csrf(request))

return render(request,'anketa/account.html',args)

Далее, для задания новых значений почты и пароля необходимо написать ajax-функцию:

def AccountInfoChanging(request):

print(request.POST)

result = {'result':0, 'error\_msg':''}

if request.method =='POST':

try:

if len(request.POST.get('email',''))>0:

user = request.user

user.email = request.POST.get('email','')

user.save()

if len(request.POST.get('passwordCurrent',''))>0:

username = request.user.username

password = request.POST.get('passwordCurrent', '')

user = authenticate(username=username, password=password)

if user is None:

result['result']=1

result['error\_msg']="Указан неверный пароль."

else:

if request.POST.get('passwordNew','') ==

request.POST.get('passwordNewVerify',''):

user.set\_password(request.POST.get('passwordNew',''))

user.save()

else:

result['result']=1

result['error\_msg']="Значения нового пароля не совпадают."

except Exception as e:

result['result']=1

result['error\_msg']=str(e)#"Что-то пошло не так."

return HttpResponse(json.dumps(result), content\_type="application/json")

Если введенное значение поля «email» не пустое, то текущему пользователю задается новое значение почты.

Для смены пароля необходимо сначала пройти еще одну аутентификацию, после этого сравниваются значения полей, в которых указан новый пароль пользователя, если значения одинаковы, то новый пароль заносится в базу данных. Если аутентификация не проходит, то появляется уведомление о том, что текущий пароль указан неверно. Если новые пароли не совпадают, то появляется уведомление о том, что пароли значения паролей не совпадают (Рис. 22).



Рис.22. Страница "Учетная запись"

**Выводы к главе 2**

Во второй главе рассмотрены процессы проектирования модуля «Личный кабинет абитуриента» для web-портала «Электронная анкета абитуриента»:

* Выявлены требования к модулю;
* Разработана диаграмма модели классов UML;
* Спланирована диаграмма переходов между страницами модуля;

Описан процесс реализации модуля, при котором использовались язык программирования python, web-фреймворк Django, скриптовый язык jQuery, Twitter Bootstrap для оформления интерфейса, Select2 для автозаполнения полей. Приведено описание готового продукта и алгоритмов работы программы.

**Заключение**

В процессе выполнения выпускной квалификационной работы был изучен процесс проведения приемной кампании в ФГАОУ ВПО Северо-Восточный федеральный университет им. М.К. Аммосова. Проведенное изучение позволяет утверждать, что автоматизация процесса сбора документов абитуриентов позволит повысить эффективность работы приемной комиссии.

Применяемая методология разработки ПО FDD, выделяет 5 этапов разработки. Процесс проектирования общей модели заключается в выявлении требований к ПО и разработке модели классов. Анализ документов, необходимых для поступления, прописанных в правилах приема университета, позволил сконструировать модель классов UML, состоящая из 25 классов.

При разработке модуля использовались язык программирования python, web-фреймворк Django, скриптовый язык jQuery, Twitter Bootstrap для оформления интерфейса, Select2 для автозаполнения полей. В результате были разработаны два приложения проекта Django, которые включают в себя: 9 шаблонов и 41 функцию.

Разработанный модуль автоматизирует процесс сбора документов, сокращает объем ручного труда, что позволяет повысить эффективность сотрудника приемной комиссии.

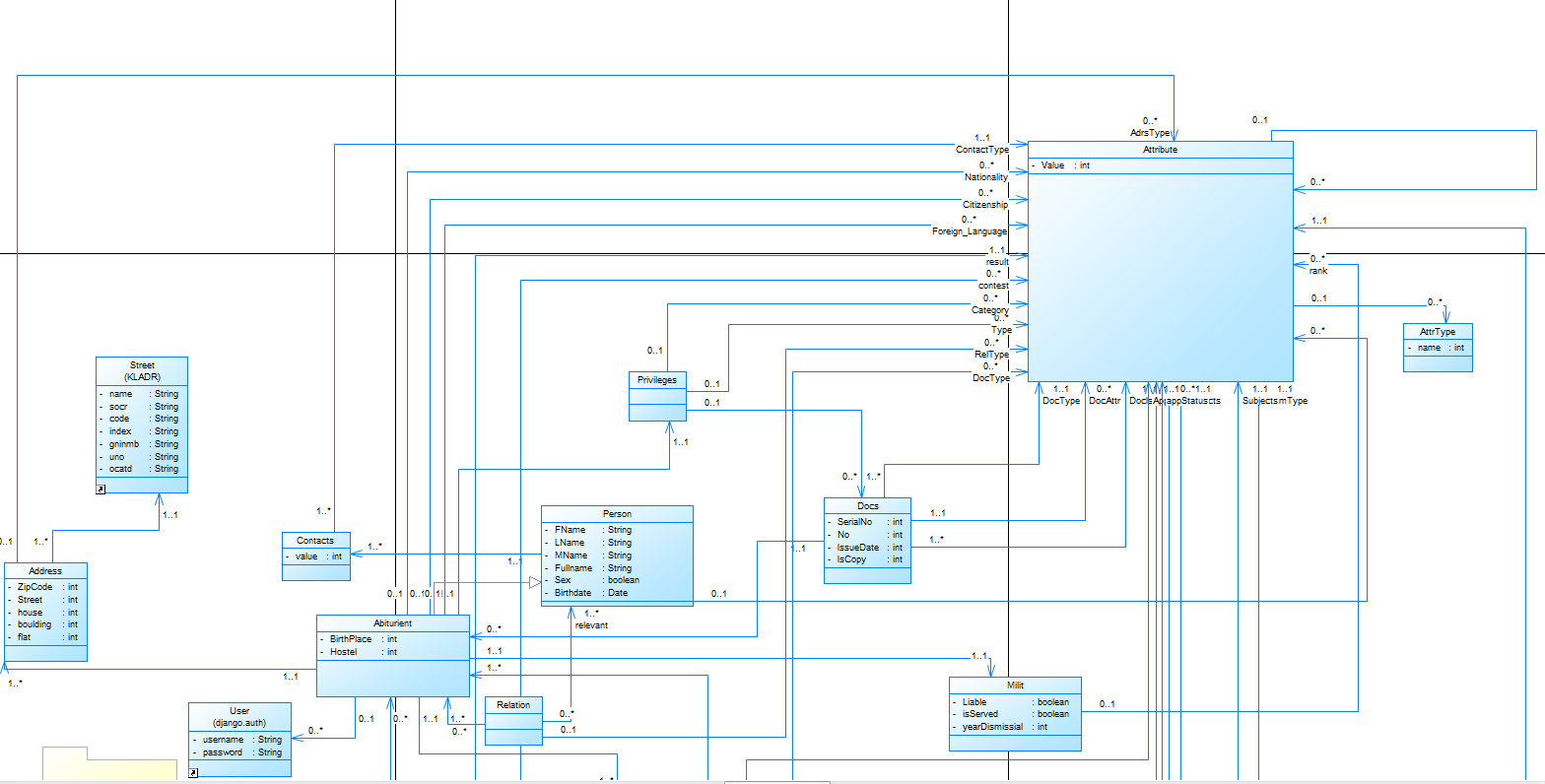
Цель выпускной квалификационной работы достигнута, все поставленные задачи решены.

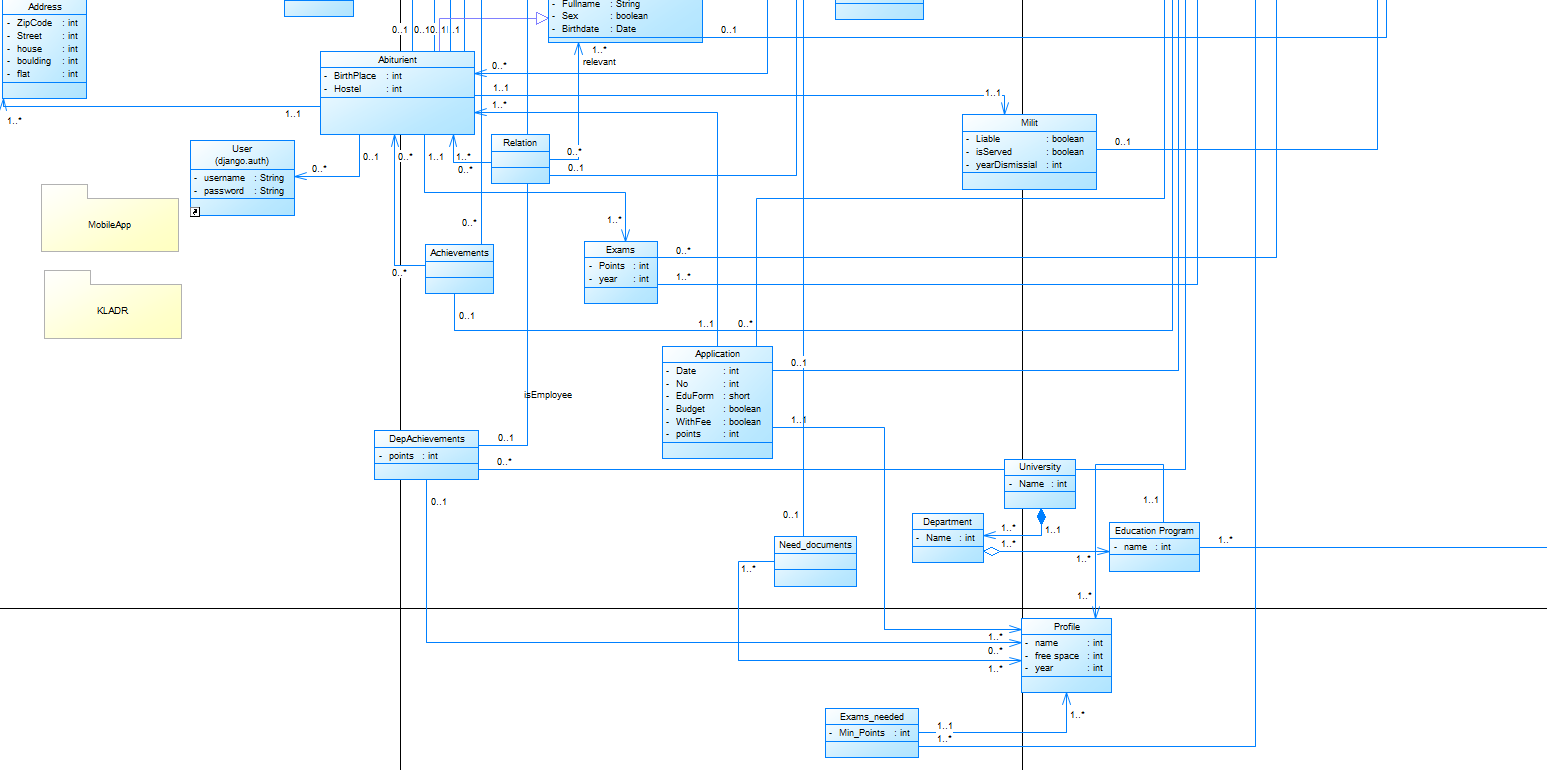
**Список использованной литературы и источников**

1. Общее положение правил // Приемная комиссия СВФУ. [2015-2015]. <URL:http://priem.s-vfu.ru/pravila-priema-2015-bakalavriat-spetsialitet-magistratura/>.
2. Перечень вступительных испытаний и минимальное количество баллов, подтверждающее успешное прохождение вступительных испытаний // Приемная комиссия СВФУ. [2015-2015]. URL:http://priem.s-vfu.ru/pravila-priema-2015-bakalavriat-spetsialitet-magistratura/.
3. Правила приема в федеральное государственное автономное образовательное учреждение высшего профессионального образования «Северо-Восточный федеральный университет имени М.К.Аммосова» // Приемная комиссия СВФУ. [2015-2015]. URL:http://priem.s-vfu.ru/pravila-priema-2015-bakalavriat-spetsialitet-magistratura/.
4. Перечень документов // Приемная комиссия СВФУ. [2015-2015]. URL:http://priem.s-vfu.ru/pravila-priema-2015-bakalavriat-spetsialitet-magistratura/.
5. Способы подачи и отзыва документов // Приемная комиссия СВФУ. [2015-2015]. <URL:http://priem.s-vfu.ru/pravila-priema-2015-bakalavriat-spetsialitet-magistratura/>.
6. Каскадная модель // Википедия. [2016—2016]. Дата обновления: 30.03.2016. URL: http://ru.wikipedia.org/?oldid=77453470 (дата обращения: 30.03.2016).
7. Гибкая методология разработки // Википедия. [2016—2016]. Дата обновления: 24.05.2016. URL: http://ru.wikipedia.org/?oldid=78563368 (дата обращения: 24.05.2016).
8. Feature driven development // Википедия. [2016—2016]. Дата обновления: 29.04.2016. URL: http://ru.wikipedia.org/?oldid=78066897 (дата обращения: 29.04.2016).
9. Заметка про PHP // Хабрахабр. [2014-2014]. <URL:http://habrahabr.ru/company/pvs-studio/blog/235189/>.
10. PHP // Википедия. [2015—2015]. Дата обновления: 08.06.2015. URL: <http://ru.wikipedia.org/?oldid=71283125>
11. PHP, Ruby, Python – краткая характеристика трёх языков программирования // Сайтостроение от А до Я.[2014-2014]. URL:http://www.internet-technologies.ru/articles/article\_1991.html
12. Python // Википедия. [2015—2015]. Дата обновления: 27.04.2015. URL: <http://ru.wikipedia.org/?oldid=70456050> (дата обращения:18.05.2015)
13. Python – обзор // Pythonicway. [2014-2014]. <URL:http://pythonicway.com/python-overview>
14. Ruby // Википедия. [2015—2015]. Дата обновления: 02.05.2015. URL: <http://ru.wikipedia.org/?oldid=70560047> (дата обращения:02.05.2015)
15. JQuery // Википедия. [2015—2015]. Дата обновления: 07.04.2015. URL: <http://ru.wikipedia.org/?oldid=69850906> (дата обращения:07.04.2015)
16. Joomla! // Википедия. [2015—2015]. Дата обновления: 22.05.2015. URL: <http://ru.wikipedia.org/?oldid=70951341> (дата обращения:22.05.2015)
17. Ruby on Rails // Википедия. [2015—2015]. Дата обновления: 31.05.2015. URL: <http://ru.wikipedia.org/?oldid=71140844> (дата обращения:31.05.2015)
18. Обзор языка программирования Ruby on Rails // Rubiks. [2013-2013] . URL: <http://www.rubiks.ru/osnovi/obzor.aspx>
19. Django // Википедия. [2015—2015]. Дата обновления: 09.06.2015. URL: <http://ru.wikipedia.org/?oldid=71306610> (дата обращения:09.06.2015)
20. Twitter Bootstrap // Википедия. [2015—2015]. Дата обновления: 11.06.2015. URL: <http://ru.wikipedia.org/?oldid=71350963> (дата обращения: 11.06.2015).
21. Обзор Foundation // g-rain design. [2014-2014]. <URL:http://g-rain-design.ru/blog/posts/foundation-responsive-framework-review/>
22. Магическое число семь плюс-минус два // Википедия. [2014—2014]. Дата обновления: 29.12.2014. URL: http://ru.wikipedia.org/?oldid=67599209

**Приложения**

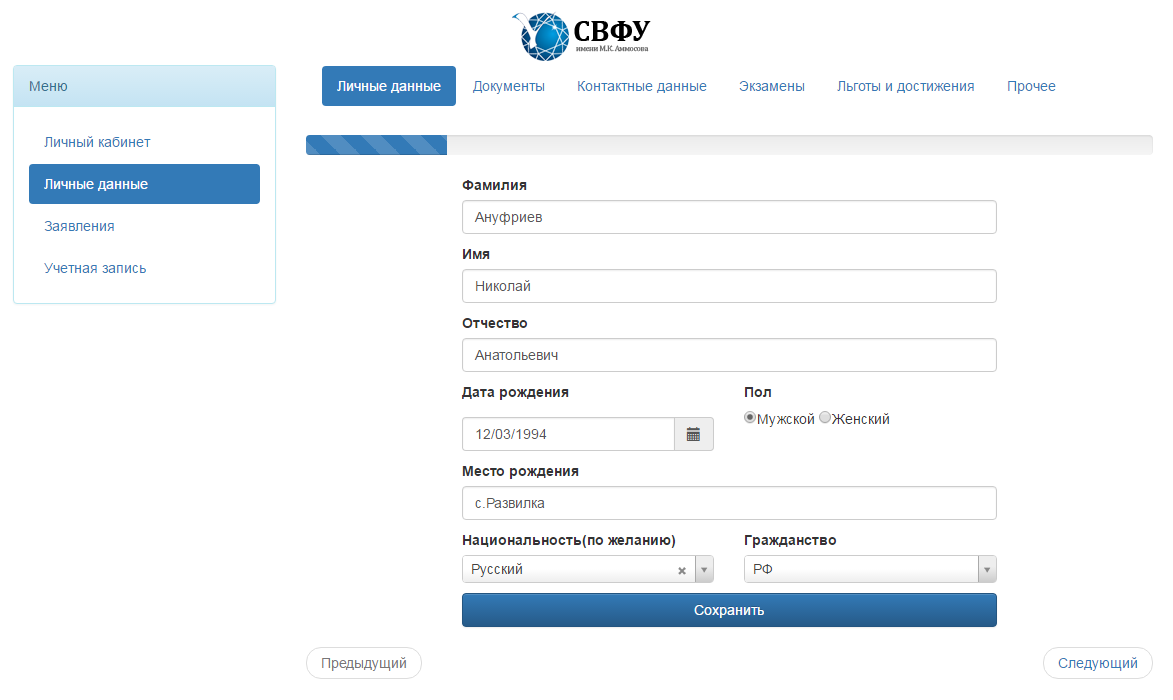
**Приложение 1. Модель классов**

****

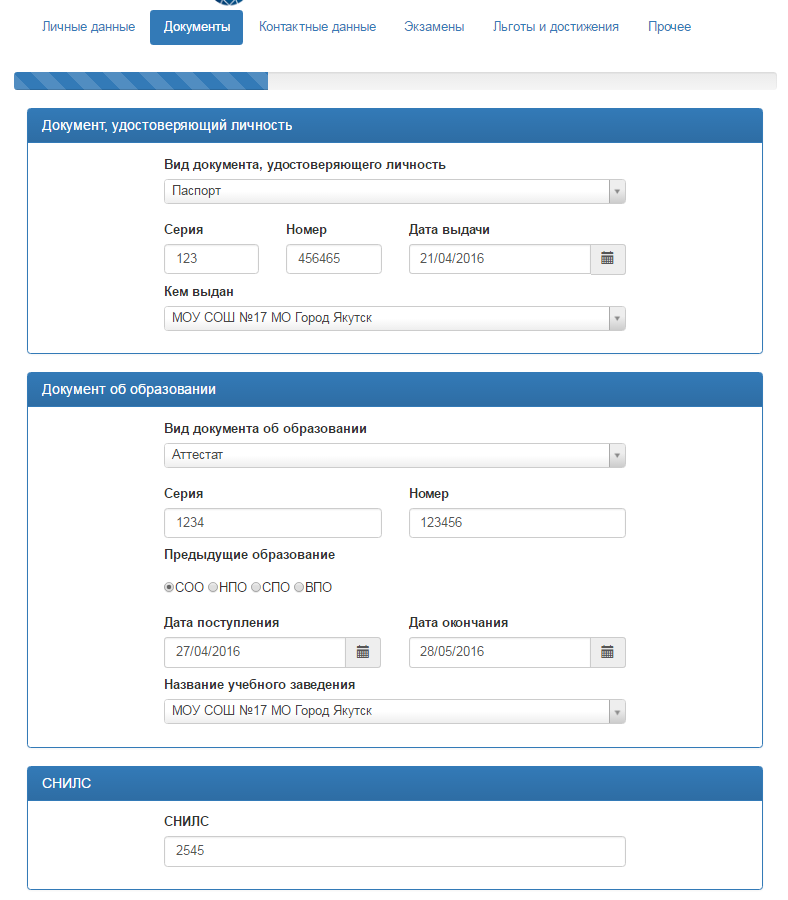
****

**Приложение 2. Страница «Личные данные»**

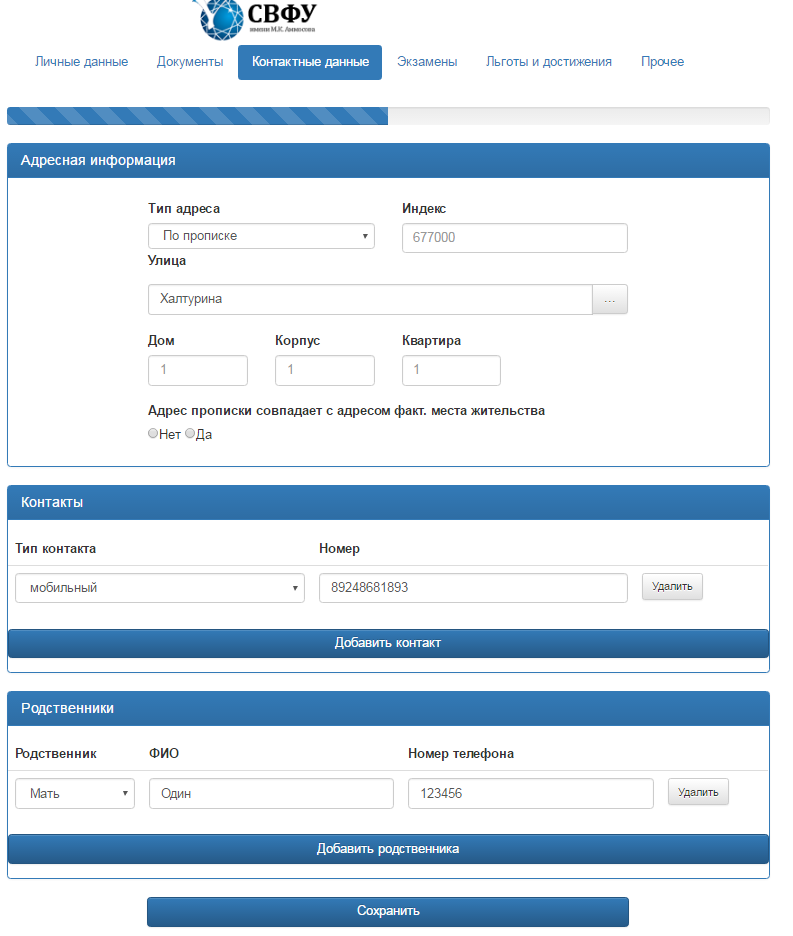
Вкладка «Личные данные»

****

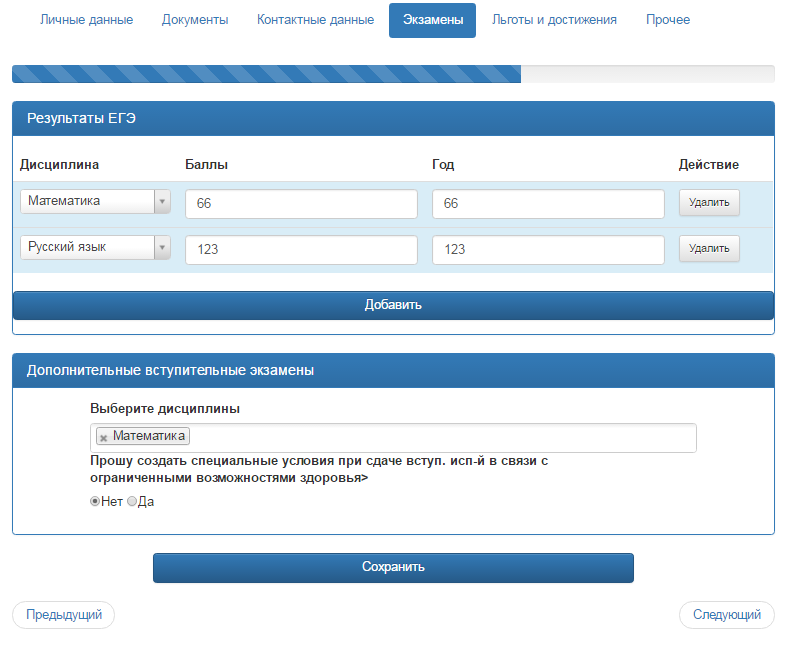
Вкладка «Документы»



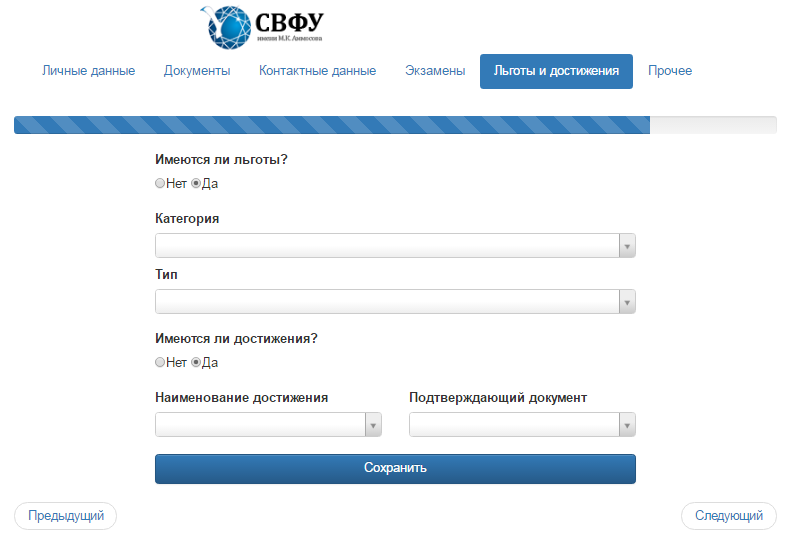
Вкладка «Контактные данные»



Вкладка «Экзамены»



Вкладка «Льготы и достижения»



Вкладка «Прочее»

