

РЕФЕРАТ

Пояснительная записка 99 с., 46 рис., 5 табл., 21 источник ПРОГРАММНОЕ СРЕДСТВО АВТОМАТИЗАЦИИ ОКАЗАНИЯ МЕДИЦИНСКОЙ КОНСУЛЬТАЦИИ

Объектом исследования является программное средство автоматизации оказания медицинской консультации.

Ключевые слова:

NodeJS, JavaScript, PostgreSQL, медицина, консультации.

Цель работы - предоставление возможности медицинского консультирования на расстоянии, что позволит получать более квалифицированные медицинские услуги в отдаленных точках российской федерации.

Предлагаемое программное средство позволит предоставлять услуги медицинского консультирования без необходимости непосредственного физического присутствия рядом с врачом. Это уменьшит количество очередей в больницах, а так же позволит получать квалифицированную медицинскую консультацию в отдаленных точках страны.

Проведен анализ достоинств и недостатков существующих программных продуктов. С их помощью разработаны и спроектированы функциональные требования к приложению.

На основе функциональных требований разработана архитектура программного средства и модель базы данных.

Разработаны тесты для проверки соответствия функциональным требованиям и корректности работы приложения.

Приведено технико-экономическое обоснование эффективности разработки и использования программного средства.

СОДЕРЖАНИЕ

Введение	6
1 Анализ прототипов, литературных источников формирование требований к проектируемому программному средству	7
1.1 Аналитический обзор литературы	7
1.2 Обзор аналогов программных средств	8
1.3 Требования к проектируемому программному средству	9
2 Моделирование предметной области и разработка функциональных требований	13
2.1 Описание функциональности ПС	13
2.2 Спецификация функциональных требований	14
3 Проектирование программного средства	16
3.1 Разработка программной архитектуры	16
3.2 Разработка модели базы данных	17
3.3 Разработка схемы алгоритма работы с программой	20
3.4 Разработка алгоритма входа пользователя в систему	20
4 Создание программного средства	22
4.1 Обоснование выбора средств разработки ПС	22
4.2 Используемые модули и фреймворки	25
4.3 Описание классов и методов	25
5 Тестирование программного средства	26
6 Руководство по установке и использованию	30
6.1 Серверная часть	30
6.2 Клиентская часть	30
7 Техничко-экономическое обоснование эффективности разработки и внедрения ПС	54
7.1 Введение	54
7.2 Расчет сметы затрат и цены ПС	54
8 Организация безопасных условий труда инженеров-программистов в БГУИР	55
Заключение	60
Список использованных источников	61
Приложение А Текст программного модуля сервера	63

ОПРЕДЕЛЕНИЯ И СОКРАЩЕНИЯ

В настоящей пояснительной записке применяются следующие определения и сокращения.

Пациент - лицо, изъявившее желание воспользоваться услугой медицинской консультации

Врач - лицо, оказывающее консультации по средствам использования системы

Телемедицина - использование компьютерных и телекоммуникационных технологий для обмена медицинской информацией.

Программа – данные, предназначенные для управления конкретными компонентами системы обработки информации в целях реализации определенного алгоритма.

AJAX - Asynchronous Javascript and XML (асинхронный JavaScript и XML)

JSON – JavaScript Object Notation

SQL – Structured Query Language (язык структурированных запросов)

URL – Uniform Resource Locator (единообразный локатор ресурса)

ВВЕДЕНИЕ

Одной из наиболее часто используемых услуг в области здравоохранения является первичная медицинская консультация. Зачастую пациенты чувствуя какой либо дискомфорт не знают к какому врачу обратиться и идут на первичную медицинскую консультацию к педиатру(если это ребенок) либо к терапевту. Терапевт в свою очередь делает общий осмотр и направляет пациента к более узконаправленному специалисту. А в случае если человек живет в удаленной точке РФ, то ввиду трудности добраться до врача люди прибегают к самолечению, либо неквалифицированным рекомендациям на форумах и сайтах.

Целью данного проекта является разработка программного средства, позволяющего пациентам получать, а врачам оказывать, медицинскую помощь на расстоянии. Разработанное ПО должно быть доступно из любого места, подключенного к сети Интернет, а также иметь возможность расширения за счет модульной архитектуры.

Подобная разработка должна обладать повышенной надежностью, защищенностью и гибкостью. История общения пациента с врачом должна быть доступна только им обоим. Работа в сети Интернет сопряжена с риском перехвата данных, особенно если работа производится в публичной локальной сети. Чтобы этого избежать, можно использовать шифрование передаваемых данных, как клиентское, так и серверное. Поддерживаемый всеми современными браузерами протокол передачи данных SSL также обеспечивает дополнительную защиту от утечки информации.

Для удобства использования интерфейс разрабатываемого программного средства должен быть понятным и легко осваиваемым. Поэтому для решения задачи можно использовать современные возможности HTML5, CSS3 и популярного JavaScript фреймворка ReactJS.

Таким образом, необходимо разработать программное средство, которое должно сочетать в себе удобство использования, надежность хранения данных и отказоустойчивость, позволяющее оказывать и получать услуге медицинской консультации на расстоянии.

1 АНАЛИЗ ПРОТОТИПОВ, ЛИТЕРАТУРНЫХ ИСТОЧНИКОВ ФОРМИРОВАНИЕ ТРЕБОВАНИЙ К ПРОЕКТИРУЕМОМУ ПРО- ГРАММНОМУ СРЕДСТВУ

1.1 Аналитический обзор литературы

1.1.1 Федеральный закон “Об основах охраны здоровья граждан в Российской Федерации”

29.07.2017 в федеральный закон “Об основах охраны здоровья граждан в Российской Федерации”, который регулирует отношения, возникающие в сфере охраны здоровья граждан в Российской Федерации. Были внесены изменения, а именно закон был дополнен статьей 36.2 “Особенности медицинской помощи, оказываемой с применением телемедицинских технологий”, которая описывает основные правила применения телемедицинских технологий. А именно следующие пункты:

1) Медицинская помощь с применением телемедицинских технологий организуется и оказывается в порядке, установленном уполномоченным федеральным органом исполнительной власти, а также в соответствии с порядками оказания медицинской помощи и на основе стандартов медицинской помощи.

2) Консультации пациента или его законного представителя медицинским работником с применением телемедицинских технологий осуществляются в целях:

– профилактики, сбора, анализа жалоб пациента и данных анамнеза, оценки эффективности лечебно-диагностических мероприятий, медицинского наблюдения за состоянием здоровья пациента;

– принятия решения о необходимости проведения очного приема (осмотра, консультации).

3) При проведении консультаций с применением телемедицинских технологий лечащим врачом может осуществляться коррекция ранее назначенного лечения при условии установления им диагноза и назначения лечения на очном приеме (осмотре, консультации).

4) Дистанционное наблюдение за состоянием здоровья пациента назначается лечащим врачом после очного приема (осмотра, консультации). Дистанционное наблюдение осуществляется на основании данных о пациенте, зарегистрированных с применением медицинских изделий, предназначенных для мониторинга состояния организма человека, и (или) на основании данных, внесенных в единую государственную информационную систему в сфере здравоохранения, или государственную информационную систему в сфере здравоохранения субъекта Российской Федерации, или медицинскую информационную систему, или информационные системы, указанные в части 5 статьи 91 настоящего Федерального закона.

5) Применение телемедицинских технологий при оказании медицинской помощи осуществляется с соблюдением требований, установленных законодательством Российской Федерации в области персональных данных, и соблюдением врачебной тайны.

6) В целях идентификации и аутентификации участников дистанционного взаимодействия при оказании медицинской помощи с применением телемедицинских технологий используется единая система идентификации и аутентификации.

7) Документирование информации об оказании медицинской помощи пациенту с применением телемедицинских технологий, включая внесение сведений в его медицинскую документацию, осуществляется с использованием

усиленной квалифицированной электронной подписи медицинского работника.”;

1.2 Обзор аналогов программных средств

Для создания принципиально нового решения в виде программного продукта необходимо ознакомиться с существующими аналогами в данной сфере. Анализ достоинств и недостатков этих аналогов позволит сформировать требования к проектируемому программному средству, учитывающие опыт существующих разработок и внести в них улучшения или изменения.

В качестве исследуемых аналогов были выбраны программные продукты, связанные с оказанием услуг телемедицины

В результате поиска были обнаружены ресурсы, представленные в таблице 1.1. В ней сведены данные по найденным аналогам и их существенным признакам.

В результате поиска были выявлены схожие технические решения, используемые при разработке исследуемых программных продуктов.

Проанализируем отличия разрабатываемого программного средства от выявленных устройств.

1) Программный продукт “Яндекс.Здоровье” (пункт 1 таблица 1.1). Это кроплатформенные продукт. Имеет свои реализации на таких платформах как Android, iOS, Web. Является платным. Схожесть продуктов обусловлена сферой применения (телемедицина).

Программный продукт обладает следующими достоинствами:

- Широкий выбор специалистов
- Возможность пользоваться системой не только с персонального компьютера, но и с мобильных устройств
- Ведение истории обращения пациента

Выявленные недостатки данного продукта:

- Оказание услуг платное
- Продукт принадлежит конкретной компании и не возможно его использовать на базе какого либо медицинского учреждения

2) Программный продукт “DOC+” (пункт 2 таблица 1.1). Это кроплатформенные продукт. Имеет свои реализации на таких платформах как Android, iOS, Web. Является платным. Схожесть продуктов обусловлена сферой применения (телемедицина).

Программный продукт обладает следующими достоинствами:

- Широкий выбор специалистов
- Возможность пользоваться системой не только с персонального компьютера, но и с мобильных устройств
- Ведение истории обращения пациента
- Широкий выбор услуг

Выявленные недостатки данного продукта:

- Оказание услуг платное
- Продукт принадлежит конкретной компании и не возможно его использовать на базе какого либо медицинского учреждения

3) Программный продукт “GetDoctor” (пункт 3 таблица 1.1). Это кроплатформенные продукт. Является платным. Схожесть продуктов обусловлена сферой применения (телемедицина).

Программный продукт обладает следующими достоинствами:

- Широкий выбор специалистов
- Выявленные недостатки данного продукта:

- Продукт принадлежит конкретной компании и не возможно его использовать на базе какого либо медицинского учреждения
- Нестабильная работа системы

Таблица 1.1 — Результаты поиска аналогов программных средств

Наименование ресурса и источник	Признаки выявленных аналогов
1. Программный продукт "Яндекс.Здоровье" ссылка[6]	Функции системы: <ul style="list-style-type: none"> - консультирование по видео, аудио, в чате - выбор типа врача - консультация как у терапевта так и у узкопрофильных специалистов - ведение медицинской карты пациента
2. DOC+ (https://docplus.ru/)[7]	Функции системы: <ul style="list-style-type: none"> - Онлайн консультация - Вызов врача на дом - ЭКГ на дому - Сдача анализов на дому
3. GetDoctor (http://getdoctor.com/) [8]	Функции системы: <ul style="list-style-type: none"> - Онлайн консультация - Получение рекомендаций

В результате поиска аналогов выявлены программные средства, схожие по назначению, однако обладающие функциональными отличиями и представляемыми возможностями. Каждый из сервисов предлагает услугу телемедицины, но возможности получить коробочный продукт и развернуть его на базе любого медицинского учреждения нет. Как следствие, патентная чистота разрабатываемого программного продукта является очевидной. Следовательно, разработка программного средства автоматизации приемной кампании обоснована и целесообразна.

1.3 Требования к проектируемому программному средству

1.3.1 Назначение разработки

Функциональное назначение программного средства

Функциональным назначением программного средства является предоставление возможности оказания и получения медицинской консультации на расстоянии.

Эксплуатационное назначение программного средства

Программное средство может использоваться как для получения и оказания консультационных услуг, так и для упрощения мед. работниками поиска, хранения информации о пациентах.

Конечными пользователями программного средства могут являться врачи, пользователи получающие консультационные услуги, руководители врачей для оценки работы их подопечных.

1.3.2 Состав выполняемых функций

Программное средство должно обеспечивать возможность выполнения перечисленных ниже функций:

- возможность обмена сообщениями в режиме реального времени

- функциональность разграничения доступа к различным функциям систем
- функциональность просмотра истории общения пациента с врачом

1.3.3 Требования к организации входных данных

Входные данные для программного средства должны быть представлены в виде вводимого пользователем с клавиатуры текста: идентификатор; пароль; электронная почта. После процесса аутентификации клиенту предоставляются возможности работы с системой.

Данные, вводимые пользователем, должны проверяться на корректность, как в процессе аутентификации, так и перед осуществлением каких-либо действий в системе.

1.3.4 Требования к организации выходных данных

В качестве выходных данных будут выступать веб-страницы, отображающие пользовательские данные и результаты их обработки.

1.3.5 Требования к временным характеристикам

Требования к временным характеристикам должны зависеть от количества работающих в данный момент времени человек, т.к. программное средство предоставляет совместный доступ к системе.

1.3.6 Требования к надежности

Требования к обеспечению надежного (устойчивого) функционирования программы

Надежное (устойчивое) функционирование программы должно быть обеспечено выполнением совокупности организационно-технических мероприятий, перечень которых приведен ниже:

- организацией бесперебойного питания технических средств;
- выполнением требований «ГОСТ 31078-2002. Защита информации.

Испытания программных средств на наличие компьютерных вирусов»;

- необходимым уровнем квалификации сотрудников.

Время восстановления после отказа

Время восстановления после отказа, вызванного сбоем электропитания технических средств (иными внешними факторами), не фатальным сбоем (не крахом) операционной системы, не должно превышать времени, необходимого на перезагрузку операционной системы и запуск программы, при условии соблюдения условий эксплуатации технических и программных средств.

Время восстановления после отказа, вызванного неисправностью технических средств, фатальным сбоем (крахом) операционной системы, не должно превышать времени, требуемого на устранение неисправностей технических средств и переустановки программных средств.

Отказы из-за некорректных действий оператора

Отказы программы возможны вследствие некорректных действий пользователя при взаимодействии с операционной системой. Во избежание возникновения отказов программы по указанной выше причине следует обеспечить работу конечного пользователя без предоставления ему ограниченного доступа.

Обеспечить максимально безотказную работу программы при любых входных данных, а также при любых действиях пользователя. Программный продукт должен соответствовать требованиям безопасности, установленным ГОСТ 27451-87, ГОСТ 26104-89.

1.3.7 Требования к составу и параметрам технических и программных средств

Требования к техническим средствам

Серверная часть программного средства должна функционировать на ЭВМ со следующими минимальными характеристиками:

- процессор Xenon с тактовой частотой 2 ГГц и более;
- 1 жесткий диска объемом в 100 гб;
- оперативная память 4 Гб и более;
- сетевая карта Ethernet 1 Гбит.

Клиентская часть программного средства должна функционировать на ЭВМ со следующими минимальными характеристиками:

- процессор Intel Pentium 4 с тактовой частотой 2 ГГц и более;
- оперативная память 2 Гб и более;
- сетевая карта Ethernet 10/100 Мбит.
- операционная система: OS Windows, Linux, Mac OS X.

1.3.8 Требования к информационной и программной совместимости

Требования к информационным структурам и методам решения

Требования к информационным структурам на входе и выходе, а также к методам решения не предъявляются.

Требования к исходным кодам и языкам программирования

Исходные коды программы должны быть реализованы на языке JavaScript. В качестве интегрированной среды разработки программы должна быть использована среда Visual Studio Code.

Требования к программным средствам, используемым программой

Системные программные средства, используемые программой, должны быть представлены локализованной версией операционной системы Windows, Linux или Mac OS X.

Требования к защите информации и программ

В системе должен быть обеспечен надлежащий уровень защиты информации в соответствии с законом о защите персональной информации и программного комплекса в целом от несанкционированного доступа – “Информационные технологии. Средства защиты информации от несанкционированного доступа в автоматизированных системах.

Требование к пользовательскому интерфейсу

Разработать удобный и интуитивно понятный пользовательский интерфейс. Программа будет состоять из серверной части и частей врача, руководителя врача и пациента. Цвет интерфейса: светло-синий, спокойный цвет.

Требования к патентной чистоте

Проектируемое решение программного продукта должно обладать патентной чистотой.

1.3.9 Обоснование выбора языка

Язык программирования JavaScript

JavaScript – прототипно-ориентированный сценарный язык программирования. Является диалектом языка ECMAScript [10].

JavaScript обычно используется как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам.

Основные архитектурные черты: динамическая типизация, слабая типизация, автоматическое управление памятью, прототипное программирование,

функции как объекты первого класса.

На JavaScript оказали влияние многие языки, при разработке была цель сделать язык похожим на Java, но при этом лёгким для использования непрограммистами. Языком JavaScript не владеет никакая-либо компания или организация, что отличает его от ряда языков программирования, используемых в веб-разработке.

Название «JavaScript» является зарегистрированным товарным знаком компании Oracle Corporation.

JavaScript является объектно-ориентированным языком, но используемое в языке прототипирование обуславливает отличия в работе с объектами по сравнению с традиционными класс-ориентированными языками. Кроме того, JavaScript имеет ряд свойств, присущих функциональным языкам – функции как объекты первого класса, объекты как списки, карринг, анонимные функции, замыкания – что придаёт языку дополнительную гибкость.

Несмотря на схожий с Си синтаксис, JavaScript по сравнению с языком Си имеет коренные отличия:

- объекты, с возможностью интроспекции;
- функции как объекты первого класса;
- автоматическое приведение типов;
- автоматическая сборка мусора;
- анонимные функции.

В языке отсутствуют такие полезные вещи, как:

- модульная система: JavaScript не предоставляет возможности управлять зависимостями и изоляцией областей видимости;
- стандартная библиотека: в частности, отсутствует интерфейс программирования приложений по работе с файловой системой, управлению потоками ввода-вывода, базовых типов для бинарных данных;
- стандартные интерфейсы к веб-серверам и базам данных;
- система управления пакетами, которая бы отслеживала зависимости и автоматически устанавливала их.

Данный язык является основополагающим в реализации данного программного продукта, реализованного в рамках этого дипломного проекта.

В качестве формата передачи данных могут использоваться фрагменты простого текста, HTML-кода, JSON или XML.

2 МОДЕЛИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ И РАЗРАБОТКА ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ

2.1 Описание функциональности ПС

Для представления функциональной модели была выбрана диаграмма вариантов использования UML [13], которая отражает отношения между актерами и прецедентами и позволяет описать систему на концептуальном уровне. Прецедент соответствует отдельному сервису системы, определяет один из вариантов её использования и описывает типичный способ взаимодействия пользователя с системой. UML предназначен для определения, визуализации, проектирования и документирования программных систем.



Рисунок 2.1 — Диаграмма вариантов использования

Диаграмма вариантов использования разрабатываемого программного средства представлена на рисунке 2.1. На диаграмме можно выделить два основных составляющих элемента – актер и прецедент. Актер – стилизованный человек, обозначающий набор ролей пользователя, взаимодействующего с некоторой сущностью. Прецедент – эллипс с надписью, обозначающий выполняемые системой действия, приводящие к наблюдаемым актером результатам.

На основании представленной диаграммы вариантов использования можно сделать вывод, что в системе будет существовать пять основных актеров: абитуриент, оператор ЭВМ, технический секретарь приемной комиссии, сотрудник приемной комиссии, ответственный секретарь приемной комиссии. Рассмотрим каждый из прецедентов более подробно для каждого актера.

2.1.1 Пациент

Пациенту предоставляются следующие возможности:

1) регистрация: новый пользователь, желающий пользоваться программным средством, вводит свои данные в соответствующую регистрационную форму (уникальный идентификатор; пароль; электронную почту), после чего заполненную форму отправляет на сервер;

2) авторизация: пользователь, являющийся полноценным пользователем программного средства, вводит собственный уникальный идентификатор и пароль, указанный при регистрации, в форму, после чего отправляет эти данные на удаленный сервер, который проверяет их и предоставляет доступ к системе;

3) отправка сообщения: авторизовавшийся пользователь имеет возможность отправлять мгновенные сообщения врачу.

4) просмотр рекомендаций, оставленных врачом: по завершении консультации врач оставляет рекомендацию пациенту. Пациент в свою очередь имеет возможность просматривать историю оставленных ему рекомендаций.

5) просмотр профиля врача: авторизовавшийся пациент имеет возможность посмотреть информацию о враче, который проводит консультацию

6) просмотр истории сообщений: пациент имеет возможность просматривать историю общения с врачом за прошедший период времени

2.1.2 Врач

1) авторизация: врач вводит собственный уникальный идентификатор и пароль, которые получает от администратора системы, после чего отправляет эти данные на удаленный сервер, который проверяет их и предоставляет доступ к системе;

2) отправка сообщения: врач имеет возможность отправлять сообщение пациенту. Все сообщения врача и пациента будут храниться в удаленной базе данных.

3) просмотр истории сообщений: врач может посмотреть историю общения с пациента за прошлый промежуток времени.

4) просмотр профиля пациента: врач имеет возможность посмотреть профиль пациента

5) отправлять рекомендацию пациенту: для завершения консультации врачу необходимо отправить рекомендацию, в которой врач на основании диалога с пациентом предоставляет список рекомендованных действий, которые необходимо сделать пациенту чтоб справиться с проблемой

2.1.3 Администратор

1) авторизация: администратор вводит адрес своей почты и пароль, которые указывает во время настройки системы, после чего отправляет эти данные на удаленный сервер, который проверяет их и предоставляет доступ к системе;

2) поиск врачей: администратор может искать врача в системе, для просмотра его профиля

3) поиск пациентов: врач может искать пользователей в системе, для просмотра их профиля

4) просмотр профиля пациента: администратор может просматривать профиль пациентов в системе

5) просмотр профиля врача: может просматривать профили врачей в системе

2.2 Спецификация функциональных требований

На основании анализа исходных данных для проектируемого программного средства можем выделить, что основной целью является создание качественного программного продукта, позволяющего решить существующие проблемы таких как:

- общение между врачом и пациентом на расстоянии;
- хранение информации о пациентах;
- хранение информации о врачах;

В ходе разработки будут реализованы следующие возможности:

- функция общения между пациентом и врачом

- функции хранения и поиска информации о врачах
- функции хранения и поиска информации о пациентах
- просмотр статистики работы врачей
- оценка работы врачей пациентами

Проект представляет собой онлайн систему с возможностью мгновенного обмена сообщениями.

Доступ к системе будет предоставляться трем категориям пользователей.

1) Пациент. Предоставляется возможность отправки сообщений врачам, просмотр информации о врачах, просмотр рекомендаций оставленных врачом.

2) Врач. Возможность обмена мгновенными сообщениями с пациентами, отправка рекомендаций пациентам, просмотр профиля пациентов.

3) Администратор. Регистрация врачей в системе. Просмотр профилей врачей и пациентов. Поиск профилей врачей и пациентов. Просмотр статистики работы врачей.

3 ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

3.1 Разработка программной архитектуры

Прежде чем приступать к непосредственной реализации программного средства, необходимо определиться с архитектурой, а также компонентов, на основе которых будет построено конечное приложение.

В первую очередь, необходимо провести анализ необходимой аппаратной конфигурации, на которой будут работать части конечного программного средства, и описать их взаимодействие между собой. Для описания узлов и их связей будем использовать диаграмму развертывания (рисунок 3.1).

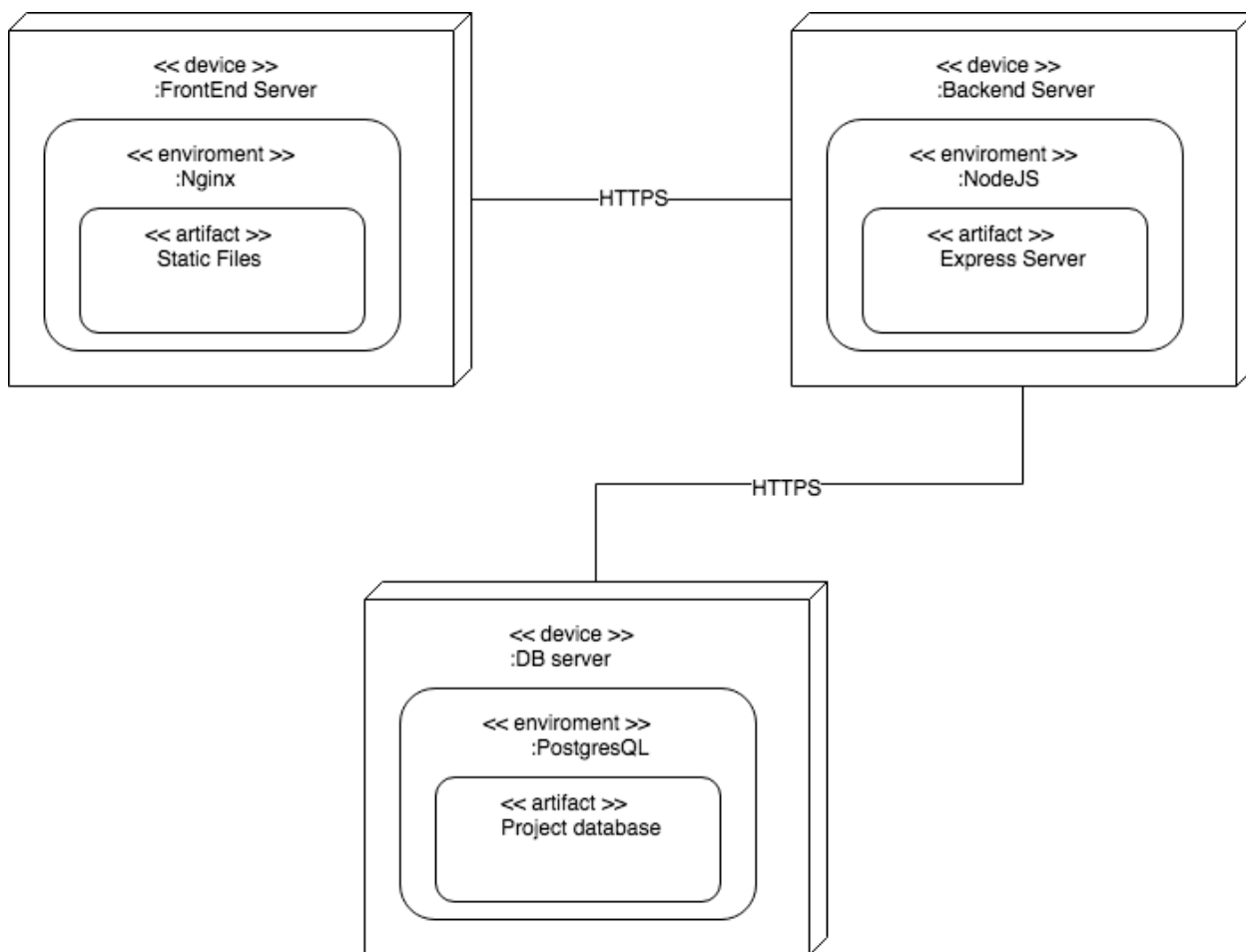


Рисунок 3.1 — Диаграмма развертывания

На основе вышеизображенной диаграммы можно сделать следующие выводы:

- 1) узлы могут располагаться в различных частях мира и взаимодействовать между собой через сеть Интернет;
- 2) сервер базы данных поддерживаются в рабочем состоянии отдельно от основного сервера;
- 3) клиент, осуществляющий работу с системой с помощью HTTPS;
- 4) HTTPS протокол применяется как клиентами, так и всевозможными серверами, с целью осуществления обмена запросами и ответами на них.

3.2 Разработка модели базы данных

Неотъемлемой частью конечного программного средства является база данных, используемая системой в процессе работы. Информационную модель предметной области можно представить на языке IDEF1X. Модель базы данных представлена на рисунке 3.2, а также на листе формата A1 (см. Графическое приложение).

TODO: переделать схему БД

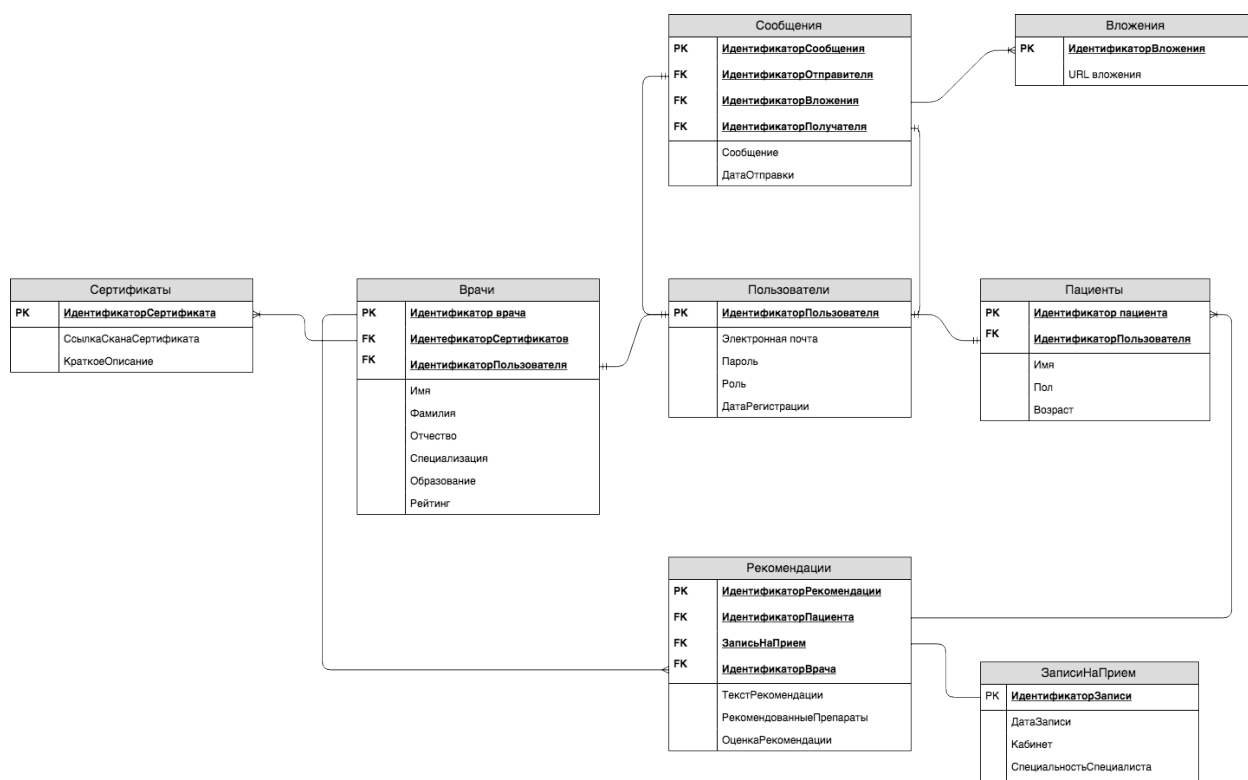


Рисунок 3.2 — Информационная модель предметной области

В модели использованы три типа связей: не идентифицирующая ноль-или-один-ко-многим, не идентифицирующая многие-ко-многим и идентифицирующая один-ко-многим. Первая обозначена штриховой линией с ромбом со стороны родительской сущности и кружком со стороны дочерней сущности, вторая – штриховой линией с ромбом со стороны родительской сущности и кружком со стороны дочерней сущности, третья – линией с кружком со стороны дочерней сущности. При наличии связи «один ко многим» одна запись в одной таблице связана с множеством записей в другой таблице. Связь между таблицами организуется на основе общего поля. На стороне «один» должно выступать ключевое поле, содержащее уникальные значения – такое поле называют внешним ключом. Значения на стороне «многие» могут повторяться.

Организация связей между таблицами обеспечивает целостность данных. Система не допустит, чтобы одноименные поля в разных таблицах имели разные значения. Ввод данных автоматически контролируется. Рассмотрим сущности по-отдельности.

3.2.1 Сущность «Пользователи»

Представляет всех пользователей в приложении со следующими полями – ИдентификаторПользователя - идентификатор пользователя (первич-

ный ключ);

- Пароль - пароль пользователя для авторизации в системе;
- Роль - роль пользователя в системе(администратор/пациент/врач);
- ДатаРегистрации - дата регистрации пользователя в системе;

Сущность имеет идентифицирующие связи «один к многим» с сущностью «Сообщения», один к одному с сущностями «Пациенты», «Врачи»

3.2.2 Сущность «Сообщения»

Представляет сообщения в системе со следующими полями:

– ИдентификаторСообщения – идентификатор сообщения (первичный ключ);

– ИдентификаторОтправителя – внешний ключ на сущность «Пользователи», представляет пользователя, отправившего сообщение;

– ИдентификаторПолучателя – внешний ключ на сущность «Пользователи», представляет пользователя, которому адресовано сообщение;

Сущность имеет связь «многие к одному» с сущностью «Пользователи», «один ко многим» с сущностью «Вложения»

3.2.3 Сущность «Вложения»

Представляет описания вложения в сообщение следующими полями:

– ИдентификаторВложения – идентификатор вложения (первичный ключ);

– ИдентификаторСообщения – внешний ключ на сущность «Сообщения»;

– URLвложения – путь до самого вложения на сервере;

Сущность имеет связь «многие к одному» с сущностью «Сообщения».

3.2.4 Сущность «Врачи»

Представляет врачей со следующими полями:

– ИдентификаторВрача – идентификатор врача (первичный ключ);

– ИдентификаторПользователя – внешний ключ на сущность «Пользователи»;

– Имя – имя врача;

– Фамилия – фамилия врача;

– Отчество – отчество врача;

– Специализация – область на которой специализируется врач; //тут тоже можно специализации вынести в таблицу отдельную

– Образование - образовательные учреждения, законченные врачом // можно вынести в отдельную таблицу

– Рейтинг - средняя оценка врача на основе всех его рекомендаций // возможно стоит удалить поле

Сущность имеет связь «один к одному» с сущностью «Пользователи», «один ко многим» с сущностями «Сертификаты», «Рекомендации».

3.2.5 Сущность «Сертификаты»

Представляет сертификаты, полученные врачом за время работы со следующими полями:

– ИдентификаторСертификата – идентификатор сертификата (первичный ключ);

– ИдентификаторВрача – внешний ключ на сущность «Врачи»;

– СсылкаСканаСертификата – ссылка на картинку со сканом сертификата;

- КраткоеОписание – описание сертификата.
- Сущность имеет связь «многие к одному» с сущностью «Врачи».

3.2.6 Сущность «Пациенты»

Представляет пациентов со следующими полями:

- ИдентификаторПациента – идентификатор пациента (первичный ключ);
- ИдентификаторПользователя – внешний ключ на сущность «Пользователи»;
- Имя – имя пациента, по которому можно к нему обращаться;
- Пол – пол пациента;
- Возраст – возраст пациента.

Сущность имеет связь «один к одному» с сущностью «Пользователи» и «один к многим» с сущностью «Рекомендации».

3.2.7 Сущность «Рекомендации»

Представляет описания рекомендаций:

- ИдентификаторРекомендации – идентификатор рекомендации (первичный ключ);
- ИдентификаторПациента – внешний ключ на сущность «Пациенты»;
- ИдентификаторВрача – внешний ключ на сущность «Врачи»;
- Запись на прием – внешний ключ на сущность «ЗаписиНаПрием»;
- Рецепт - внешний ключ на сущность «Рецепты» // TODO: добавить таблицу рецепты
- Текст рекомендации - сама рекомендация по лечению заболевания от врача;
- Оценка рекомендации - оценка рекомендации пользователем;

Сущность имеет связь «один ко многим» с сущностью «Абитуриенты». Сущность имеет связь «многие к одному» с сущностями «Врачи» и «Пациенты», «один к одному» с сущностями «ЗаписиНаПрием» и «Рецепты».

3.2.8 Сущность «ЗаписиНаПрием»

- ИдентификаторЗаписи - идентификатор записи на прием (первичный ключ);
- ДатаЗаписи - дата приема пациента
- СпециальностьСпециалиста - специальность специалиста к которому пациент записан на прием

3.2.9 Сущность «Рецепты»

- ИдентификаторРецепта - идентификатор рецепта (первичный ключ)
- ИдентификаторЛекарста - внешний ключ на сущность «Лекарства».
- ИдентификаторРекомендации - внешний ключ на сущность «Рекомендации»

Сущность имеет связь «один к одному» с сущностью «Рекомендации»; Сущность имеет связь «один к многим» с сущностью «Лекарства».

3.2.10 Сущность «Лекарства»

- ИдентификаторЛекарста - идентификатор лекарства (первичный ключ);
 - НазваниеЛекарства - название лекарства;
 - ОписаниеЛекарства - описание лекарства.
- Сущность имеет связь «многие к одному» с сущностью «Рецепты»;

3.3 Разработка схемы алгоритма работы с программой

Схема алгоритма выполнена на листе формата А1 (см. Графическое приложение), а также на рисунке 3.5. Данная схема отображает алгоритм работы с сайтом. Браузер пытается подключиться и загрузить страницу. Далее пользователь имеет ряд путей работы: переходы по ссылкам внутри сайта и использование его разделов. Если пользователь закрывает вкладку или переходит по внешней ссылке, то работа с сайтом закончена.

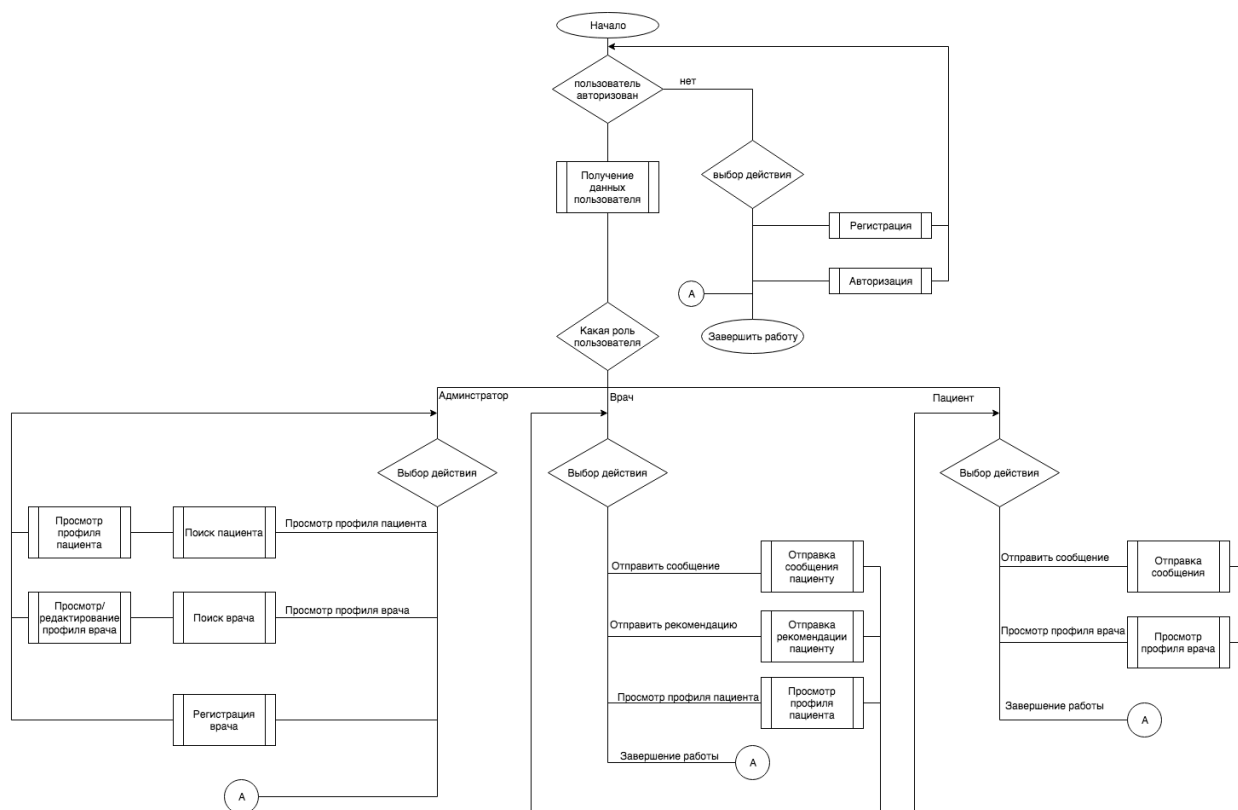


Рисунок 3.3 — Схема алгоритма работы с программой

3.4 Разработка алгоритма входа пользователя в систему

Вход пользователя в систему состоит из аутентификации и авторизации. Аутентификация представляет собой поиск соответствия пользователя названному им идентификатору. Авторизация – предоставление этому пользователю возможностей в соответствии с положенными ему правами. Алгоритм аутентификации представлен на рисунке 3.6.

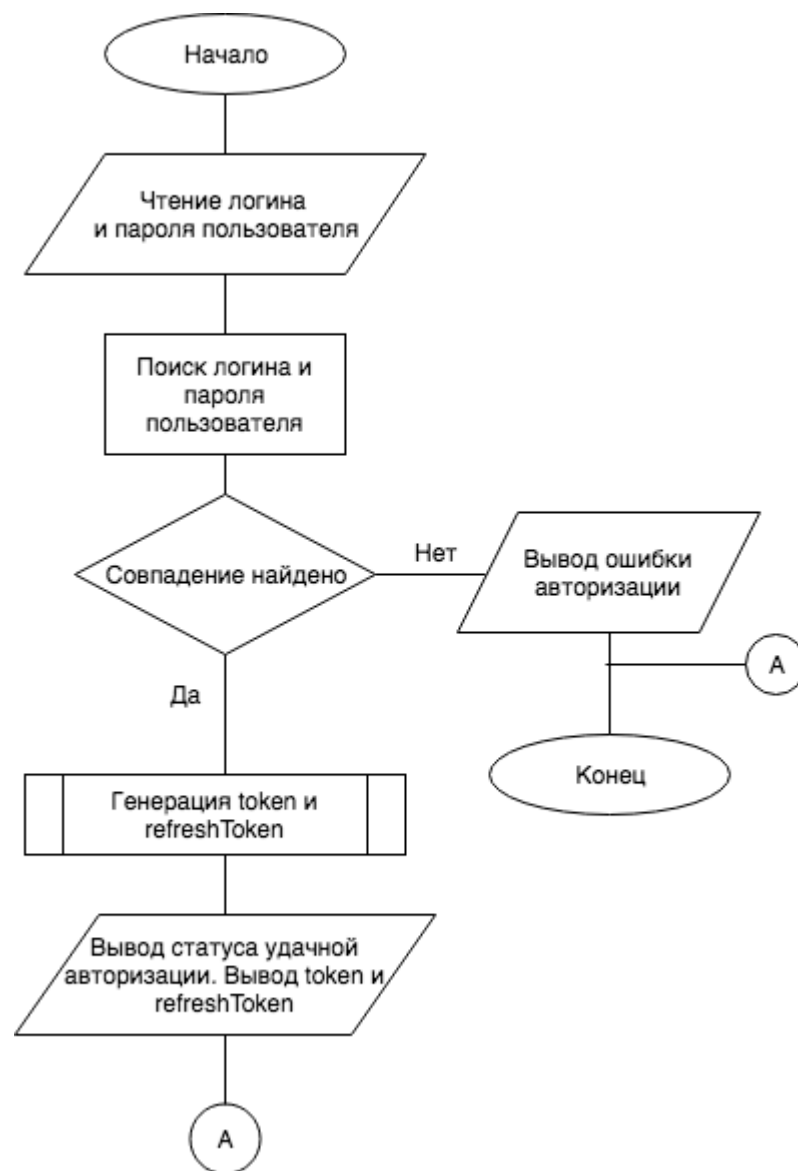


Рисунок 3.4 — Схема алгоритма аутентификации

Данный алгоритм реализует функцию аутентификации в системе. Сразу после входа на страницу аутентификации система предлагает ввести пользователю логин и пароль. В базе данных происходит поиск пользователя по его логину. В случае если пользователь с введенными логином не найден система возвращает ошибку. В случае если пользователь найден, то извлекается хешированный пароль этого пользователя из базы данных. Хешированный пароль сравнивается с хешированным паролем, который ввел пользователь. В случае их не совпадения система возвращает ошибку авторизации, иначе происходит генерация двух токенов. Основного токена, по которому будет проверяться авторизован пользователь или нет и “refresh” токена, на основании которого будет генерироваться основной токен в случае истечения срока его работы.

4 СОЗДАНИЕ ПРОГРАММНОГО СРЕДСТВА

Спецификация функциональных требований и спроектированная архитектура программного средства служат фундаментом, на котором основывается выбор наиболее подходящих технологий для разработки программного средства. Успешное и обоснованное завершение данных этапов позволит создать расширяемое, надежное и функциональное приложение, призванное решать поставленные задачи.

4.1 Обоснование выбора средств разработки ПС

4.1.1 GraphQL

GraphQL- язык запросов данных, разработанный внутри Facebook в 2012 году, после чего публично выпущенный в 2015 году. Он предоставляет альтернативу архитектуре REST и ad-hoc для веб-сервисов. Он позволяет клиентам определять структуру требуемых данных, и точно такая же структура данных возвращается с сервера. Это строго типизированная среда выполнения, которая позволяет клиентам определять, какие данные необходимы, поэтому предотвращает чрезмерное количество возвращаемых данных.

4.1.2 Протокол шифрования SSL

SSL (англ. Secure Sockets Layer – уровень защищённых сокетов) – криптографический протокол, который обеспечивает безопасность связи. Он использует асимметричную криптографию для аутентификации ключей обмена, симметричное шифрование для сохранения конфиденциальности, коды аутентификации сообщений для целостности сообщений. Протокол широко используется для обмена мгновенными сообщениями и передачи голоса через IP (англ. Voice over IP – VoIP) в таких приложениях, как электронная почта, Интернет-факс и др.

SSL изначально разработан компанией Netscape Communications для добавления протокола HTTPS в свой веб-браузер Netscape Navigator. Впоследствии, на основании протокола SSL 3.0 был разработан и принят стандарт RFC, получивший имя TLS.

Протокол SSL позволяет общаться клиенту с сервером в сети, предотвращая перехват или фальсификацию. Так как протоколы могут работать либо без SSL, либо поверх SSL, то для клиента необходимо указать серверу, хочет ли он установить соединение SSL или нет. Есть две возможности сделать это. Одним из вариантов является использование различных номеров портов для соединения SSL (например, порт 443 для HTTPS). Другой заключается в использовании регулярного номера порта, сервер установит соединение с клиентом, используя протокол конкретного механизма (например, STARTTLS для почты и новостных протоколов). После того как клиент и сервер решили использовать SSL, они ведут переговоры, отслеживая состояние соединения с помощью процедуры рукопожатия. Вовремя этого рукопожатия клиент и сервер соглашаются на различные параметры, используемые для установки безопасного соединения. После завершения процедуры рукопожатия начинается защищенное соединение. Клиент и сервер используют сеансовые ключи для шифрования и дешифрования данных, которые они посылают друг другу. Это нормальный алгоритм работы по защищенному каналу. В любое время, в связи с внутренним или внешним раздражителем (автоматическое вмешательство или вмешательство пользователя), любая из сторон может пересмотреть сеанс связи. В этом случае весь процесс повторяется. SSL работает модульным спо-

собою [15].

В протоколе SSL все данные передаются в виде записей-объектов, состоящих из заголовка и передаваемых данных.

Данная технология отвечает за безопасную передачу различных данных между клиентской частью и сервером данных.

4.1.3 Язык программирования JavaScript

JavaScript – прототипно-ориентированный сценарный язык программирования. Является диалектом языка ECMAScript.

JavaScript обычно используется как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам.

Основные архитектурные черты: динамическая типизация, слабая типизация, автоматическое управление памятью, прототипное программирование, функции как объекты первого класса.

На JavaScript оказали влияние многие языки, при разработке была цель сделать язык похожим на Java, но при этом лёгким для использования непрограммистами. Языком JavaScript не владеет какая-либо компания или организация, что отличает его от ряда языков программирования, используемых в веб-разработке.

Название «JavaScript» является зарегистрированным товарным знаком компании Oracle Corporation.

JavaScript является объектно-ориентированным языком, но используемое в языке прототипирование обуславливает отличия в работе с объектами по сравнению с традиционными класс-ориентированными языками. Кроме того, JavaScript имеет ряд свойств, присущих функциональным языкам – функции как объекты первого класса, объекты как списки, карринг, анонимные функции, замыкания – что придаёт языку дополнительную гибкость.

Несмотря на схожий с Си синтаксис, JavaScript по сравнению с языком Си имеет коренные отличия:

- объекты, с возможностью интроспекции;
- функции как объекты первого класса;
- автоматическое приведение типов;
- автоматическая сборка мусора;
- анонимные функции.

В языке отсутствуют такие полезные вещи, как:

- модульная система: JavaScript не предоставляет возможности управлять зависимостями и изоляцией областей видимости;
- стандартная библиотека: в частности, отсутствует интерфейс программирования приложений по работе с файловой системой, управлению потоками ввода-вывода, базовых типов для бинарных данных;
- стандартные интерфейсы к веб-серверам и базам данных;
- система управления пакетами, которая бы отслеживала зависимости и автоматически устанавливала их.

Данный язык является основополагающим в реализации клиентской части программного средства, реализованного в рамках этого дипломного проекта.

4.1.4 Технология AJAX

Asynchronous Javascript and XML – подход к построению интерактивных пользовательских интерфейсов веб-приложений, заключающийся в «фоно-

вом» обмене данными браузера с веб-сервером. В результате, при обновлении данных веб-страница не перезагружается полностью, и веб-приложения становятся быстрее и удобнее.

AJAX – не самостоятельная технология, а концепция использования нескольких смежных технологий. AJAX базируется на двух основных принципах:

1)

Использование технологии динамического обращения к серверу «на лету», без перезагрузки всей страницы полностью, например:

- с использованием XMLHttpRequest (основной объект);

- через динамическое создание дочерних фреймов;

- через динамическое создание тега

- через динамическое создание тега , как это реализовано в google analytics.

2)

Использование DHTML для динамического изменения содержания страницы.

Действия с интерфейсом преобразуются в операции с элементами DOM (англ. Document Object Model), с помощью которых обрабатываются данные, доступные пользователю, в результате чего представление их изменяется. Здесь же производится обработка перемещений и щелчков мышью, а также нажатий клавиш. Каскадные таблицы стилей, или CSS (англ. Cascading Style Sheets), обеспечивают согласованный внешний вид элементов приложения и упрощают обращение к DOM-объектам. Объект XMLHttpRequest (или подобные механизмы) используется для асинхронного взаимодействия с сервером, обработки запросов пользователя и загрузки в процессе работы необходимых данных.

Три из этих четырех технологий – CSS, DOM и JavaScript – составляют DHTML (англ. Dynamic HTML) [16]. По мнению некоторых специалистов средства DHTML, появившиеся в 1997 году, подавали большие надежды, но так и не оправдали их.

В качестве формата передачи данных могут использоваться фрагменты простого текста, HTML-кода, JSON или XML.

Данная технология применяется для взаимодействия клиентской части с серверной без необходимости полной перезагрузки страницы на клиентской стороне.

4.1.5 Протокол WebSocket

Протокол полнодуплексной связи (может передавать и принимать одновременно) поверх TCP-соединения, предназначенный для обмена сообщениями между браузером и веб-сервером в режиме реального времени.

4.1.6 База данных PostgreSQL

4.2 Используемые модули и фреймворки

4.2.1 NodeJS

4.2.2 ReactJS

4.2.3 Express

4.3 Описание модулей и компонентов

Что в системе используется?

4.3.1 Модель User

4.3.2 Модель Patient

4.3.3 Модель Doctor

4.3.4 Модель Message

4.3.5 Резолвер Doctor

4.3.6 Резолвер Patient

4.3.7 Резровер User

4.3.8 Резолвер Message

5 ТЕСТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

Проведено тестирование программного средства. Целью данного испытания было ознакомление с программным средством и проверка его работоспособности.

Установка и тестирование программного средства производилась на персональном компьютере с установленной операционной системой Windows 8. ПС протестировано в последних версиях наиболее популярных браузеров: Google Chrome, Safari, Mozilla Firefox, Internet Explorer, Opera.

Таблица 5.1 — Набор тест-кейсов модуля «Авторизация»

№ тест-кейса	Тестируемая функциональность	Последовательность действий	Ожидаемый результат	Полученный результат
1	Авторизация. Валидация ошибок	1) Запустить приложение. 2) Нажать на кнопку «Войти».	1) Сообщение под полем ввода логина «Пожалуйста, заполните поле Имя пользователя».	Тест успешно пройден
2	Авторизация. Валидация ошибок	1) Запустить приложение. 2) Ввести заведомо ложный логин и пароль. 3) Нажать на кнопку «Войти».	1) Сообщение под полем ввода логина «Введенные имя и пароль неверны».	Тест успешно пройден

Следующий набор тест-кейсов рассматривается под ролью «Сотрудник приемной комиссии». Шаг ввода корректного пароля и логина выполнен.

Таблица 5.2 — Набор тест-кейсов модуля «Абитуриенты»

№ тест-кейса	Тестируемая функциональность	Последовательность действий	Ожидаемый результат	Полученный результат
3	Абитуриенты. Отображение таблицы абитуриентов.	1) Переход на страницу просмотра списка абитуриентов.	1) Отображение таблицы абитуриентов со статусом «Очередь». 2) Фильтр не заполнен.	Тест успешно пройден

Продолжение таблицы 5.2

№ тест-кейса	Тестируемая функциональность	Последовательность действий	Ожидаемый результат	Полученный результат
4	Абитуриенты. Отображение таблицы абитуриентов.	1) Переход на страницу просмотра списка абитуриентов. 2) Ввод в поле «Фамилия» заведомо некорректных данных. 3) Нажатие кнопки «Получить».	1) Отображение сообщения «Ничего не найдено».	Тест успешно пройден
5	Абитуриенты. Отображение таблицы абитуриентов. Добавление в очередь.	1) Переход на страницу просмотра списка абитуриентов. 2) Переход в категорию «Сайт». 3) Выбор абитуриента из таблицы. 4) Нажатие кнопки «Добавить в очередь». 5) Переход в категорию «Очередь».	1) Добавленный с сайта в очередь абитуриент отображается в таблице категории «Очередь».	Тест успешно пройден
6	Абитуриенты. Отображение таблицы абитуриентов. Добавление в очередь.	1) Переход на страницу просмотра списка абитуриентов. 2) Нажатие кнопки «Создать». 3) Заполнение формы. 4) Нажатие кнопки «Сохранить».	1) Добавленный абитуриент отображается в таблице категории «Очередь».	Тест успешно пройден
7	Абитуриенты. Отображение таблицы абитуриентов. Принятие документов	1) Переход на страницу просмотра списка абитуриентов. 2) Выбор абитуриента из таблицы. 3) Нажатие кнопки «Принять документы». 4) Ввести номер личного дела. 5) Переход в категорию «БГУИР».	1) Добавленный абитуриент отображается в таблице категории «БГУИР».	Тест успешно пройден

Следующий набор тест-кейсов рассматривается под ролью «Сотрудник приемной комиссии». Шаг ввода корректного пароля и логина выполнен. Осуществлен переход на страницу создания абитуриента в очередь.

Таблица 5.3 — Набор тест-кейсов модуля «Абитуриенты»

№ тест-кейса	Тестируемая функциональность	Последовательность действий	Ожидаемый результат	Полученный результат
8	Абитуриенты. Создание записи.	1) Переход на страницу создания записи.	1) Отображение формы создания записи.	Тест успешно пройден
9	Абитуриенты. Создание записи.	1) Нажатие кнопки «Сохранить» без ввода данных.	1) Сообщения о необходимости заполнить обязательные поля.	Тест успешно пройден
10	Абитуриенты. Создание записи.	1) Ввод фамилии, имени и отчества.	1) Сообщение о возможной потере введенных данных.	Тест успешно пройден
11	Абитуриенты. Создание записи.	1) Ввод в «Расчет среднего балла» нескольких значений от 0 до 15.	1) Значение 0 заменяется на 10. 2) Значения больше 9 заменяются на 1.	Тест успешно пройден
12	Абитуриенты. Создание записи.	1) Ввод серии и номера паспорта абитуриента, сдавшего ЦТ. 2) Нажатие кнопки «ЦТ».	1) Отображение предметов и баллов ЦТ в соответствующих полях.	Тест успешно пройден
13	Абитуриенты. Создание записи.	1) Заполнение раздела «Адрес». 2) Нажатие кнопки «Адрес абитуриента».	1) Появление в поле адреса родителя адреса абитуриента.	Тест успешно пройден
14	Абитуриенты. Создание записи.	1) Заполнение информации о матери. 2) Нажатие кнопки «Мать» в графе «Представитель».	1) Появление информации матери в соответствующих полях.	Тест успешно пройден
15	Абитуриенты. Создание записи.	1) Введение мобильного телефона в формате «291234567».	1) Номер телефона преобразован в «+375 (29) 123-45-67».	Тест успешно пройден

Продолжение таблицы 5.3

№ тест-кейса	Тестируемая функциональность	Последовательность действий	Ожидаемый результат	Полученный результат
16	Абитуриенты. Создание записи. Специальности.	1) Переход во вкладку «Специальности».	1) Отображение фильтра специальностей. 2) Отображение списка выбора специальностей.	Тест успешно пройден
17	Абитуриенты. Создание записи. Специальности.	1) Переход во вкладку «Специальности». 2) Выбор факультета.	1) Список специальностей принадлежит только выбранному факультету.	Тест успешно пройден
18	Абитуриенты. Создание записи. Специальности.	1) Переход во вкладку «Специальности». 2) Выбор факультета и специальности. 3) Нажатие кнопки «Очистить форму».	1) Появление предупреждения возможности потери данных.	Тест успешно пройден
19	Абитуриенты. Создание записи. Специальности.	1) Переход во вкладку «Специальности». 2) Выбор параметров специальностей. 3) Выбор факультета. 4) Выбор специальности. 5) Нажатие кнопки «Очистить форму». 6) Нажатие «Очистить».	1) Форма очищена от введенных ранее данных.	Тест успешно пройден
20	Абитуриенты. Создание записи. Льготы.	1) Переход во вкладку «Льготы». 2) Выбор «Без экзаменов».	1) Раскрывшийся список со списком льгот с возможностью выбора чекбоксом.	Тест успешно пройден

Успешность прохождения тестов показывает корректность работы программы с реальными данными и соответствие функциональным требованиям.

6 РУКОВОДСТВО ПО УСТАНОВКЕ И ИСПОЛЬЗОВАНИЮ

6.1 Серверная часть

Для корректной работы данного программного средства необходим сервер следующей минимальной конфигурации:

- ОС: Windows (версии 7 или 8.1);
- Процессор: Pentium® III 800 МГц или AMD Athlon;
- RAM: 1024 Мб;
- HDD: 2 Гб свободного места.

Дополнительно: наличие установленной в операционной системе СУБД Microsoft SQL Server, интерпретатора и компилятора языка программирования C#, предустановленный в ОС набор серверов IIS 7.5, а также пакетный менеджер NuGet.

Эти действия необходимы для развертывания серверной части приложения. После чего необходимо перейти в папку с приложением и в консоли Developer Command Prompt for VS2012 выполнить следующую команду:

```
msbuild Priem_BSUIR.sln /p:DeployOnBuild=true  
/p:PublishProfile=Staging
```

Данная команда выполняет компиляцию, построение, установку и обновление существующих в приложении модулей и фреймворков. После успешного обновления и установки пакетов выполняется запуск сервера. Для этого необходимо выполнить следующие команды:

```
net start iisadmin  
net start w3svc
```

Команда запустит службу IIS. После конфигурирования сервера сайтом можно пользоваться через любой браузер, установленный в системе.

6.2 Клиентская часть

6.2.1 Вход в систему

При первом входе в систему на экран выводится предупреждение о том, что системой разрешено пользоваться только сотрудникам приемной комиссии (рис. 6.1).



Белорусский государственный университет информатики и радиэлектроники

Автоматизированная система подачи заявления и зачисления – Внимание!!!

Вы пытаетесь подключиться к информационной системе приемной комиссии.
Пользоваться этой системой разрешено только работникам приемной комиссии.
Несанкционированное использование системы запрещено.

Вы желаете продолжить подключение к системе?

Да

Нет

Рисунок 6.1 — Экран предупреждения

Для продолжения работы с системой необходимо нажать кнопку «Да». Если дальнейшая работа с системой не предполагается, нажмите кнопку «Нет».

При нажатии на кнопку «Нет» происходит перенаправление на сайт priem.bsuir.by.

При нажатии на кнопку «Да» система перенаправляет пользователя на страницу авторизации (рис. 6.2).

АСПЗиЗ

Имя пользователя

Пароль

Войти

Рисунок 6.2 — Страница авторизации

При вводе некорректных данных система выдаст соответствующее сообщение (рис. 6.3).

Введенные имя и пароль неверны.

Имя пользователя

SomeIncorrectUser

Пароль

.....|

Войти

Рисунок 6.3 — Сообщение об ошибке

6.2.2 Меню системы

При вводе подходящих имени и пароля открывается доступ к странице главного меню системы, из которого можно перейти в необходимые разделы (рис. 6.4).

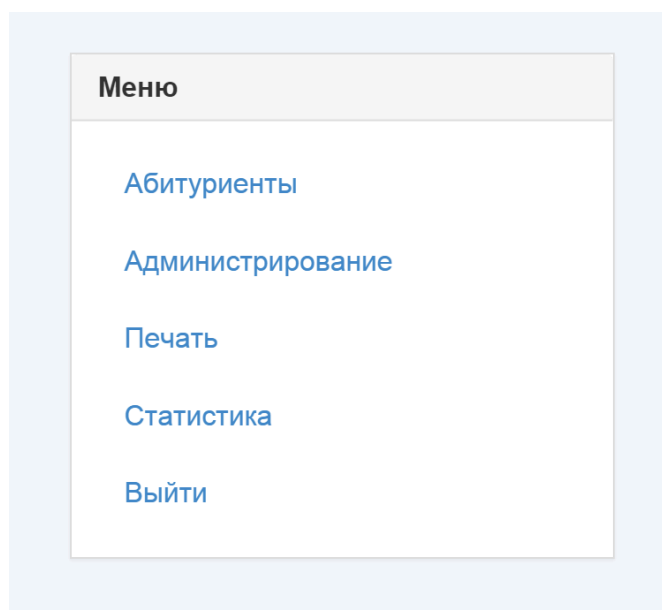


Рисунок 6.4 — Меню системы

6.2.3 Абитуриенты

Для работы с данными абитуриентов необходимо перейти в пункт меню «Абитуриенты». Имя текущего модуля отображается вверху страницы (рис. 6.5).

Рисунок 6.5 — Имя текущего модуля

Работа с абитуриентами возможна из следующих списков:

- Сайт. В данном списке отображаются абитуриенты, зарегистрировавшиеся в «Электронном кабинете БГУИР»

- Очередь. В данном списке показаны пользователи, добавленные в очередь операторами приемной комиссии.

- БГУИР. В данном списке находятся абитуриенты, полностью прошедшие процедуру подачи документов в БГУИР, но еще не зачисленные в университет.

- БГУИР (зачисл.). В этом списке показаны зачисленные системой и решением приемной комиссии абитуриенты.

- БГУИР (не зачисл.). В данном списке отображаются абитуриенты, не прошедшие по конкурсу в университет, но еще не забравшие свои документы из приемной комиссии.

- Забрали. В это списке показаны абитуриенты, забравшие свои документы из приемной комиссии на каком-либо из этапов их подачи в БГУИР.

Выбор режима осуществляется при помощи выпадающего списка сверху страницы (рис. 6.6).

БГУИР АСПЗиЗ Модуль: Абитуриенты

Очередь ▼

☐ Бюджет ☐ МГВРК

☐ Платное

Рисунок 6.6 — Выпадающий список выбора режима работы

Рисунок 6.7 — Фильтр поиска

Для удобства поиска абитуриентов по различным параметрам в системе предусмотрен фильтр поиска (рис. 6.7), доступный вверху страницы модуля «Абитуриенты». Поиск возможен по следующим параметрам, указанным в заявлении абитуриента:

- вид обучения;
- дата подачи заявления;
- срок обучения;
- форма обучения;
- факультет;
- специальность;
- фамилия;
- имя;
- отчество;
- серия и номер документа, удостоверяющего личность;
- признак окончания обучения в МГВРК.

Лн	ФИО	Б/Пл	Срок	МГВРК	ФО	Факл	Спец
ВКВ001	Борисовец Олег Геннадьевич	Пл	Полный	Ц	Вечерняя форма	ФКСиС	Вычислительные машины, системы и сети
ВКВ003	Кирвель Андрей Игоревич	Пл	Полный	Ц	Вечерняя форма	ФКСиС	Вычислительные машины, системы и сети
ВКВ004	Федорченко Владислав Витальевич	Пл	Полный	Ц	Вечерняя форма	ФКСиС	Вычислительные машины, системы и сети
ВКВ005	Соловьев Андрей Сергеевич	Пл	Полный	Ц	Вечерняя форма	ФКСиС	Вычислительные машины, системы и сети
ВКВ006	Осиевский Константин Олегович	Пл	Полный	Ц	Вечерняя форма	ФКСиС	Вычислительные машины, системы и сети
ВКВ007	Дубских Сергей Николаевич	Пл	Полный	Ц	Вечерняя форма	ФКСиС	Вычислительные машины, системы и сети
ВКВ008	Пянко Павел Игоревич	Пл	Полный	Ц	Вечерняя форма	ФКСиС	Вычислительные машины, системы и сети

Рисунок 6.8 — Результаты фильтрации списка абитуриентов

При нажатии на кнопку «Получить данные» найденные результаты отображаются в таблице под фильтром (рис. 6.8).

Количество найденных результатов отображается в правом верхнем углу списка. Для перехода в режим работы с отдельным абитуриентом необходимо нажать кнопку «Просмотреть» после выбора абитуриента кликом мышки из списка.

Для занесения абитуриента в очередь необходимо перейти в режим «Очередь» и нажать кнопку «Создать» (рис. 6.9).

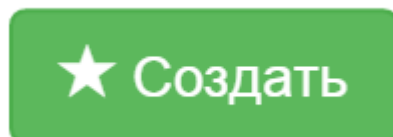


Рисунок 6.9 — Кнопка «Создать»

6.2.4 Заполнение электронного заявления

Заполнение электронного заявления состоит из трех этапов:

а) указание личной информации:

- ФИО (рис. 6.10);
- дата рождения;
- пол;
- семейное положение;
- тип документа, удостоверяющего личность (рис. 6.11);
- идентификационный номер документа;
- серия, номер, дата выдачи и кем документ выдан;
- балл сертификата ЦТ или вступительного испытания (рис. 6.12);
- номер сертификата ЦТ;
- балл по предмету в документе об образовании;
- уровень образования (рис. 6.13);
- учреждение образования;
- документ об окончании УО;
- номер учебного заведения или аббревиатура;
- номер документа об окончании УО;
- дата окончания;
- изучаемый в УО иностранный язык;
- признак окончания подготовительного отделения вуза;
- почтовый индекс;
- страна, область, район, населенный пункт, улица, дом и/или квартира проживания (рис. 6.14);
- признак нужды в проживании в общежитии во время обучения в вузе;
- домашний телефон (рис. 6.15);
- мобильный телефон;
- электронная почта;
- льготы при зачислении (рис. 6.16);
- место работы, должность, стаж и стаж по специальности (рис. 6.17);
- информация о родителях (рис. 6.18);
- данные законного представителя для несовершеннолетних (рис. 6.19).

1. ФИО

Фамилия

Имя

Отчество

Дата рождения

Пол

☐ Мужской

☐ Женский

Семейное положение

☐ Холост/Не замужем

☐ Женат/Замужем

Рисунок 6.10 — Личная информация: ФИО

2. Документ, удостоверяющий личность

Тип документа

Паспорт гражданина Республики | ▼

Идентификационный номер

Серия

Номер

Дата выдачи

Кем выдан

Рисунок 6.11 — Личная информация: документ, удостоверяющий личность

3. Вступительные испытания				
ЦТ	Дисциплина	Кол-во баллов	Номер сертификата	Балл в документе об образовании
<input type="checkbox"/>	Белорусский язык ▼	<input type="text"/>	<input type="text"/>	
<input type="checkbox"/>	Физика	<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="checkbox"/>	Математика	<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="checkbox"/>	Английский язык ▼	<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="checkbox"/>	Осн.информационных технологий	<input type="text"/>		
Ср. балл документа об образовании		0	<input type="button" value="↓ Показать расчет среднего балла"/>	

Рисунок 6.12 — Личная информация: вступительные испытания

3. Образование	
Уровень	<input type="text"/>
Учреждение	<input type="text"/>
Документ	<input type="text"/>
Номер УЗ или аббревиатура	<input type="text"/>
Номер документа	<input type="text"/>
Дата окончания	<input type="text"/>
Иностранный язык	<input type="text"/>
окончил(а) подготовительное отделение ВУЗа <input type="checkbox"/>	

Рисунок 6.13 — Личная информация: образование

4. Адрес	
Индекс	<input type="text"/>
Страна	<input type="text"/>
Область	<input type="text"/>
Район	<input type="text"/>
Тип населенного пункта	<input type="text"/>
Название населенного пункта	<input type="text"/>
Тип улицы	<input type="text"/>
Название улицы	<input type="text"/>
Номер дома	<input type="text"/>
Номер корпуса	<input type="text"/>
Номер квартиры	<input type="text"/>
<input type="checkbox"/> Нуждаюсь в общежитии на время учебы	

Рисунок 6.14 — Личная информация: адрес

5. Телефон и email	
Дом. тел. код города	<input type="text"/>
Дом. телефон	<input type="text"/>
Моб. телефон	<input type="text"/>
Эл.почта	<input type="text" value="prcom@bsuir.by"/>

Рисунок 6.15 — Личная информация: телефон и email

6. Льготы при зачислении
<div><div></div><div></div></div>

Рисунок 6.16 — Личная информация: льготы при зачислении

7. Трудовая деятельность

Место работы и должность:

Стаж (общий):

 лет месяцев,

в том числе по профилю избранной специальности

Рисунок 6.17 — Личная информация: трудовая деятельность

8. Родители

Отец

Тип родства

Фамилия

Имя

Отчество

Адрес

☐ такой же, как у абитуриента

Место работы и должность

Мать

Тип родства

Фамилия

Имя

Отчество

Адрес

☐ такой же, как у абитуриента

Место работы и должность

Рисунок 6.18 — Личная информация: родители

Законный представитель (данные)		Мать
ФИО	<input type="text"/>	Отец
Адрес	<input type="text"/>	

Законный представитель (Документ, удостоверяющий личность)	
Тип документа	<input type="text"/>
Идентификационный номер	<input type="text"/>
Серия	<input type="text"/>
Номер	<input type="text"/>
Дата выдачи	<input type="text"/>
Кем выдан	<input type="text"/>

Рисунок 6.19 — Личная информация: законный представитель

- б) указание специальностей в порядке приоритета (рис. 6.20):
- признак второго высшего образования;
 - вид обучения;
 - срок обучения;
 - приоритет специальности;
 - форма обучения;
 - факультет;
 - специальность.

Личная информация

Специальности

Льготы

☐ Второе высшее на 2-3 курс.

☒ Бюджет

☐ Платное

 Срок обучения:

☐ Сокращенная подготовка по интегрированным планам с МГВРК по дневной форме обучения

Очистить

Приоритет	Факультет / Специальность
1	<div> <div>Форма обучения:</div> <input type="text"/> </div> <div> <div>Факультет:</div> <input type="text"/> </div> <div> <div>Специальность:</div> <input type="text"/> </div>

Рисунок 6.20 — Специальности

- в) данные о льготах при поступлении (рис. 6.21):
- без экзаменов;
 - вне конкурса;
 - преимущество при равном количестве баллов при зачислении.

Личная информация

Специальности

Льготы

☒ По конкурсу

☐ Без вступительных испытаний (Глава 4. Пункт 24 Правил приема)

☐ Вне конкурса (оценки не ниже 6 баллов по предметам вступительных испытаний) (Глава 4. Пункт 26 Правил приема)

☐ Преимущественное право на зачисление при равном общем количестве баллов

Рисунок 6.21 — Льготы

Для сохранения заявления необходимо нажать кнопку «Сохранить». Для отмены заполнения заявления нажмите кнопку «Назад» (рис. 6.22).



Рисунок 6.22 — Кнопки сохранения и отмены

6.2.5 Особенности заполнения заявления

При попытке сохранить заявления без заполнения обязательных полей система выдаст соответствующее сообщение (рис. 6.23).

1. ФИО

Фамилия

*

Имя

*

Отчество

*

Дата рождения

Пол

☐ Мужской

☐ Женский

Укажите пол, пожалуйста.

Семейное положение

☐ Холост/Не замужем

☐ Женат/Замужем

Укажите семейное положение, пожалуйста.

Рисунок 6.23 — Сообщения о необходимости заполнения обязательных полей

При нажатии кнопки «ЦТ» в разделе «3. Вступительные испытания» система автоматически заполнит раздел данными о сертификатах ЦТ из базы данных РИКЗ на основании введенных данных о документе, удостоверяющем личность, в разделе 2.

Кнопка «Показать расчет среднего балла» открывает поле автоматиче-

ского подсчета среднего балла документа об образовании на основании оценок в документе (рис. 6.24).

Ср. балл
документа
об
образовании

81

↑ Показать расчет среднего балла

☐ оценки документа об образовании по пятибалльной шкале

Оценки документа об образовании:

8+6+7+9+7+8+6+10+8+6+7+8+10+10+10+8+8+9+8+

Сумма баллов: 153
Кол-во оценок: 19
Средний балл:
8.052631578947368
Средний балл по 100-балльной:
81

2: 0
3: 0
4: 0
5: 0
6: 3

7: 3
8: 7
9: 2
10: 4

Рисунок 6.24 — Расчет среднего балла

При необходимости перевода оценок из пятибалльной в десятибалльную шкалу нужно поставить галочку в пункте «оценки документа об образовании по пятибалльной шкале». Средний балл считается в режиме реального времени на основании уже введенных баллов.

Если у пользователя имеется электронный кабинет, то в поле email раздела 5 подставляется его электронный адрес из электронного кабинета. Иначе выставляется значение по умолчанию prcom@bsuir.by.

Для ускорения заполнения заявления в разделе «8. Родители» фамилии и адреса родителей заполняются автоматически при выставлении галочки «такой же, как у абитуриента».

Данные законного представителя заполняются автоматически при нажатии кнопок «Отец» или «Мать» на основании введенных ранее данных родителей.

Кнопка «Очистить» во вкладке «Специальности» очищает все введенные пользователем данные в этой вкладке, предварительно требуя подтверждения пользователя (рис. 6.25).

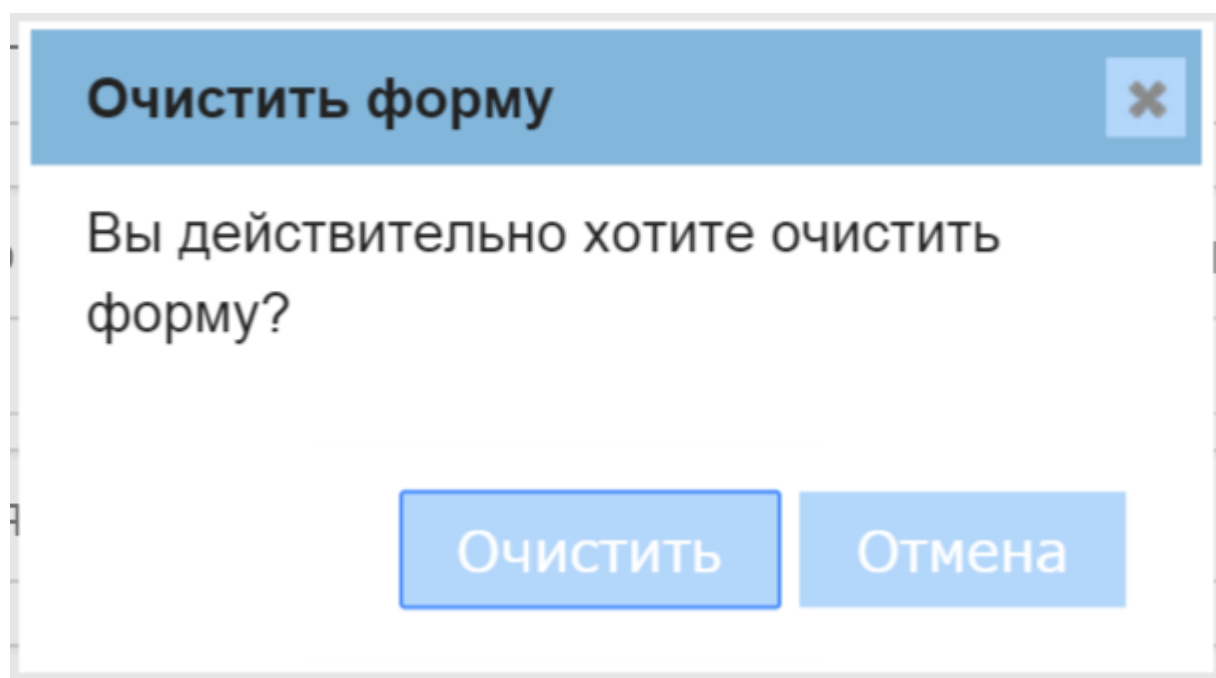


Рисунок 6.25 — Окно подтверждения очистки формы

Приоритет специальности в списке можно менять перетаскиванием с зажатой кнопкой мыши строки специальности в таблице.

При выборе льготы во вкладке «Льготы» необходимо указать в раскрывающемся списке конкретное наименование льготы (рис. 6.26).

☒ Вне конкурса (оценки не ниже 6 баллов по предметам вступительных испытаний) (Глава 4. Пункт 26 Правил приема)

☒ дети-сироты

☐ кадеты

☐ военные

☐ пограничники

☐ почетный караул

☐ второе высшее

☒ Преимущественное право на зачисление при равном общем количестве баллов

☐ выпускники Лицея №1 г. Минска при БГУИР

☐ победители олимпиад БГУИР(диплом 1-ой степени)

☐ победители олимпиад БГУИР(диплом 2-ой степени)

☐ победители олимпиад БГУИР(диплом 3-ой степени)

☐ стаж практической работы соответствующей или родственной специальности не менее двух лет;

☐ закончившие учреждения среднего специального образования по соответствующей специальности;

☐ закончившие учреждения профессионально-технического образования по соответствующей специальности;

Рисунок 6.26 — Выбор льготы

При попытке выйти из режима заполнения заявления с введенными в форму данными нажатием на кнопку «Назад» без сохранения система потребует подтверждения перехода у пользователя (рис. 6.27).

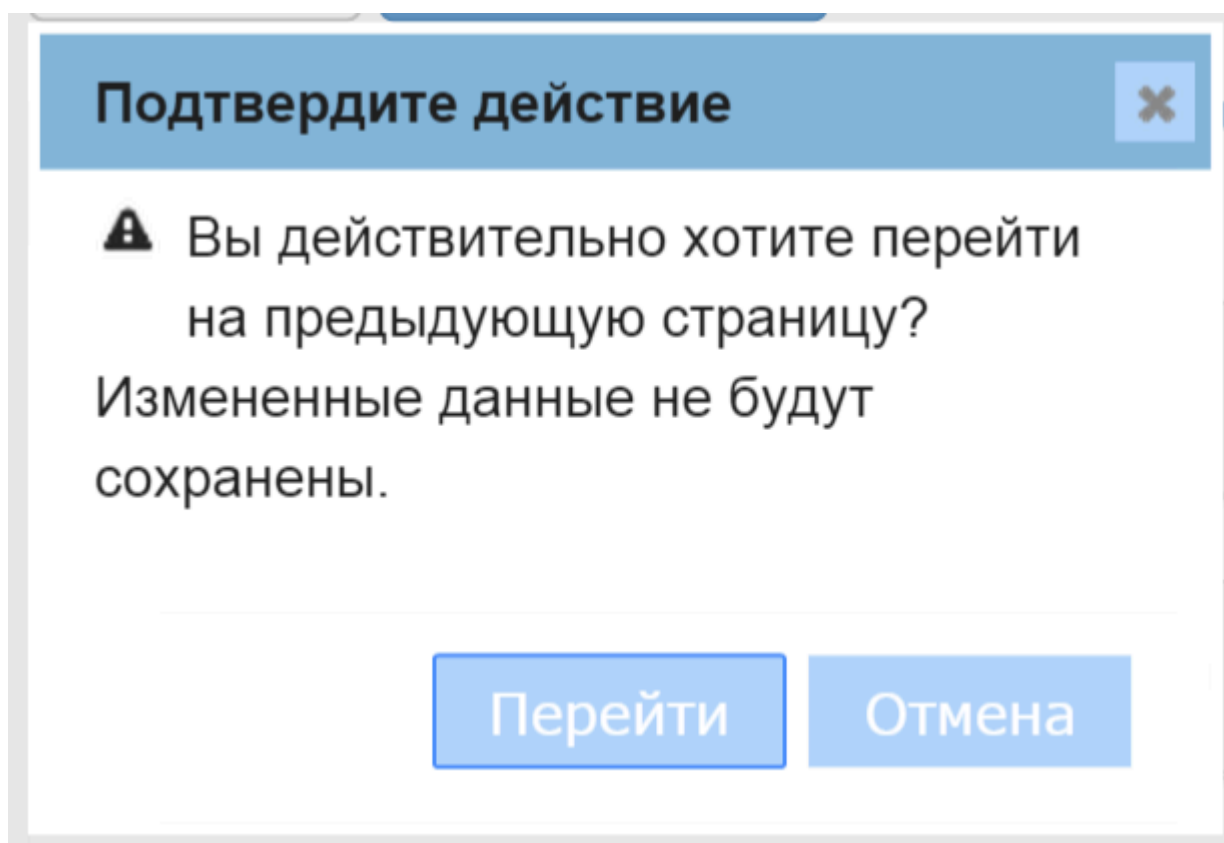


Рисунок 6.27 — Окно подтверждения перехода

При успешном сохранении заявления абитуриент должен появиться в списке «Очередь».

6.2.6 Принятие документов

При дальнейшем рассмотрении заявление абитуриента выбирается из списка «Очередь», после чего становятся доступными следующие действия с заявлением (рис. 6.28):

- редактирование данных заявления;
- удаление заявления;
- создание электронного кабинета на основании заявления;
- принятие документов у абитуриента.



Рисунок 6.28 — Доступные действия с заявлением

Режим редактирования аналогичен режиму создания нового заявления.

После нажатия кнопки «Принять документы» необходимо будет ввести номер личного дела абитуриента во всплывающем окне (рис. 6.29).

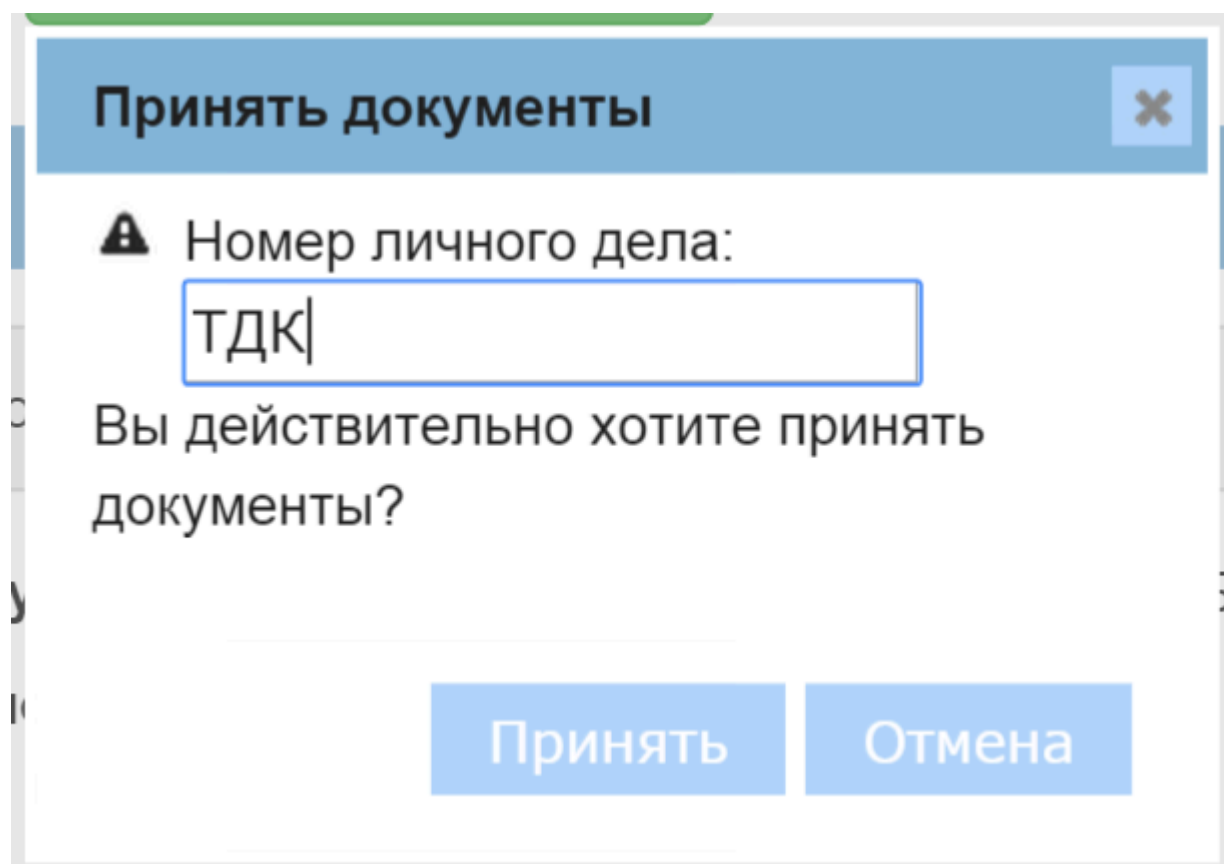


Рисунок 6.29 — Окно принятия документов

Помимо этого в режиме Очереди доступна вкладка «Печать», в которой направить на печатающее устройство как само заявление в формате PDF (рис. 6.30), так и необходимые для оформления личного дела документы (обложка личного дела, расписка, договор).

После принятия документов заявление абитуриента попадает в список «БГУИР», где находится до процедуры выдачи заявления или зачисления.

Допустить к вступительным испытаниям Ректор _____ М.П.Батура "___" _____ 2014г. Ректору БГУИР от _____ (фамилия, имя, отчество абитуриента) который проживает по адресу _____ _____ (указать почтовый индекс, адрес постоянной прописки и телефон) _____ и закончил(а) _____ в 2010 году ГУО СОШ №2 г.Пинск _____ (год окончания, наименование учреждения образования)	Зачислить на _____ курс на факультет _____ специальность _____ Приказ от "___" _____ № _____ Ректор _____ М.П.Батура
---	--

ЗАЯВЛЕНИЕ

Прошу допустить меня к участию в конкурсе для получения высшего образования 1 ступени на условиях оплаты по специальностям в порядке приоритета:

1. Факультет: Факультет непрерывного и дистанционного обучения
 (Дистанционная форма)

Специальность: Программное обеспечение информационных технологий

О себе сообщаю следующие сведения

Число, месяц, год рождения 3 января 1993 г.

Какой иностранный язык изучал(а) в среднем учебном заведении Английский язык

Место работы, занимаемая должность (профессия) _____

Трудовой стаж по профилю избранной специальности _____

нуждаюсь в общежитии (да, нет) нет

Родители проживают по адресу: _____

Рисунок 6.30 — Генерируемое в формате PDF заявление

6.2.7 Зачисление

В момент зачисления становится доступной кнопка «Зачислить» (рис. 6.31).



Рисунок 6.31 — Кнопки управления заявлением

По ее нажатию появляется диалоговое окно указания специальности, на которую зачисляется абитуриент (рис. 6.32).

Зачислить

☐ Второе высшее на 2-3 курс.

☐ Бюджет

☒ Платное

Срок обучения:

Полный

☐ Сокращенная подготовка по интегрированным планам с МГВРК по дневной форме обучения

Факультет / Специальность

Форма обучения:

Дневная форма

Факультет

Факультет компьюте|

Специа

Информационные си

⚠

Зачислить абитуриента?

Зачислить

Отмена

Рисунок 6.32 — Окно выбора специальности, на которую зачисляется абитуриент

При успешном проведении зачисления появляется соответствующее уведомление. Заявление попадает в список «БГУИР (зачисл.)» (рис. 6.33).

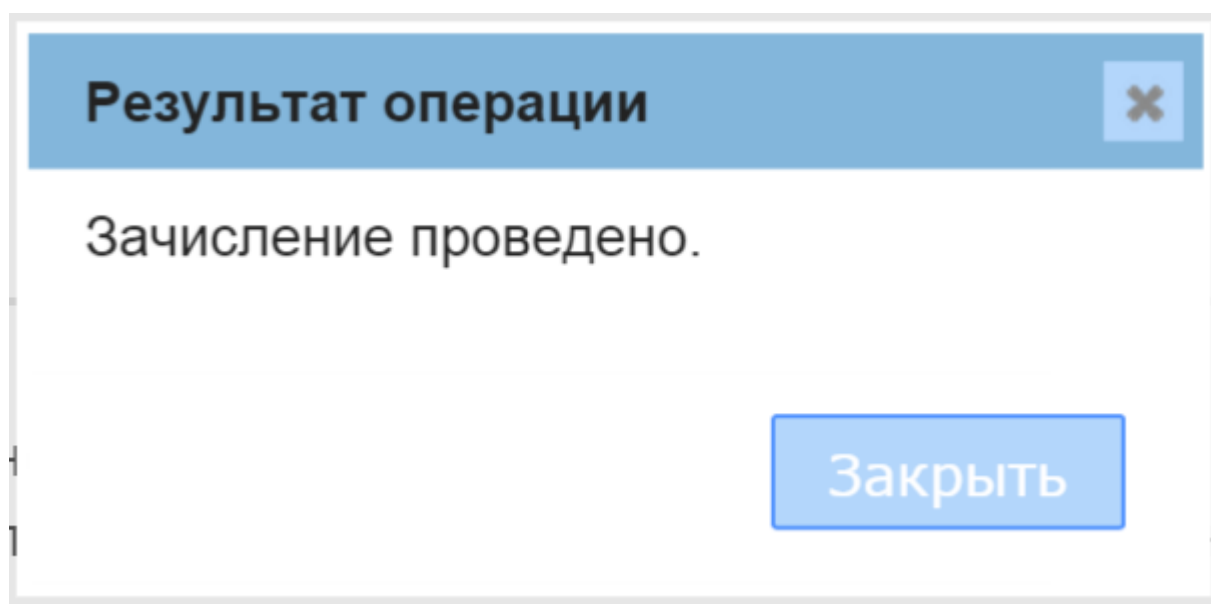


Рисунок 6.33 — Окно уведомления об успешности проведения зачисления

Из списка «Забрали» абитуриента можно вернуть в очередь либо удалить его заявление из базы данных приемной комиссии предназначенными для этого кнопками (рис. 6.34).

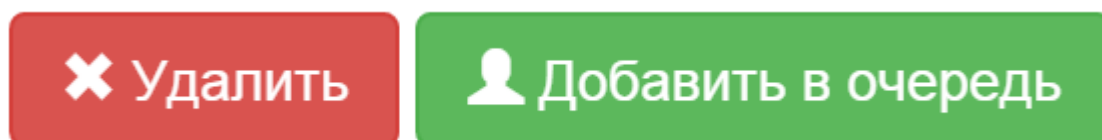


Рисунок 6.34 — Кнопки управления заявлением

6.2.8 Модуль Администрирование

Второй пункт меню, доступный администраторам системы – Администрирование (рис. 6.35).

★Зачисление	★Создать	👤Просмотреть	✖Удалить
UserName ↑↓	Login ↑↓	Locked ↑↓	Last Login ↑↓
Дубов Никита Александрович	nikn_melody	No	18.12.2014 11:11
Корень В. В.	korovsk	No	01.08.2014 10:44
Левина В. М.	lv123	No	31.07.2014 12:42
Морозов И. И.	moroz	No	31.07.2014 17:37
Петров С. И.	pet1234	No	31.07.2014 13:56

Рисунок 6.35 — Страница администрирования

На данной странице есть кнопки создания, просмотра и удаления пользователей системы. При создании каждому пользователю устанавливается логин, пароль, полное имя для печати и набор флагов доступа к определенным функциям системы, среди которых:

- сайт;

- сайт. Редактировать;
- сайт. Удалять;
- сайт. Создать кабинет;
- сайт. Добавить в очередь;
- очередь;
- очередь. Создать;
- очередь. Редактировать;
- очередь. Удалять;
- очередь. Создать кабинет;
- очередь. Принять документы;
- университет;
- университет. Редактировать;
- университет. Создать кабинет;
- университет. Выдать документы;
- университет. Зачислять;
- забранные;
- забранные. Удалять;
- забранные. Добавить в очередь;
- статистика;
- статистика. Для сайта;
- статистика. Подробно;
- статистика. Для печати;
- администрирование.

Помимо этого на этой странице находится кнопка «Зачисление», нажатие на которую вызывает работу алгоритма проведения предварительного зачисления абитуриентов в университет согласно модели зачисления данного ВУЗ-а. Результатов работы алгоритма является Excel-таблица со списком зачисленных и не зачисленных в университет и на конкретные специальности абитуриентов, а также статистика заполнения мест по специальностям.

6.2.9 Печать

Третий пункт меню – Печать (рис. 6.36). В нем доступна работа с документами, которые необходимо подготовить для печати и отправки зачисленных абитуриентам (конверты, письма, справки в военкомат).

Рисунок 6.36 — Страница «Печать»

Каждая кнопка при нажатии формирует необходимые документы для абитуриентов, находящихся в списке после фильтрации, собирает их в один архив в формате zip и отдает его на скачивание пользователю.


6.2.10 Статистика

Пункт меню «Статистика» генерирует и отдает на скачивание статистику поданных заявлений с разбиением по диапазонам баллов в формате Excel (рис. 6.38). Статистика настраивается полями выбора временного диапазона подачи заявлений (рис. 6.37).


Даты подачи

Статистика (полная)

Дата подачи с

08.07.2014

Дата подачи по

18.12.2014

на дату

Рисунок 6.37 — Настройка статистики

		Информация о поданных заявлениях на ДНЕВНУЮ ПЛАТНУЮ форму обучения на 18.12.2014 14:12:31																																			
		Информация о числе поданных заявлений отражает количество поданных заявлений по первой специальности из приоритетного списка, заявляемого абитуриентом.																																			
Факультет / группа специальностей	Специальность	План приема	Подано заявлений			Распределение по баллам для участвующих в конкурсе																															
			Всего	Без оспорительных испытаний	Вне конкурса	301-400	301-390	371-390	301-370	351-380	341-350	331-340	321-330	311-320	301-310	291-300	281-290	271-280	261-270	251-260	241-250	231-240	221-230	211-220	201-210	191-200	181-190	171-180	161-170	151-160	141-150	131-140	121-130	111-120	101-110	90-100	иные
ФКП	МикПРС	5	4			4																	1	3													
	ПыПРУС	5	2			2																				1											
	ПМС	45	24			24								1												2	3	1	4	1	2	1					
	ПУЭОС	10	1			1																		1													
	МедЭ	10	5			5																															
	ИПОМТ	15	8			8																															
	ЭСБ	15	11			11																															
	ИСИТ(ПБ)	15	2			2																															
	ИСИТ(ЕМ)	45	53			53								3	1	1	2	1	6	4	5	3	6	6	1	4	1	2	1	2	3	1					
ЭВС	5	0			0																																
итого	170	110	0	0	110								4	1	1	4	2	6	8	6	6	13	8	5	10	5	7	4	9	8	2	1					
ФКТУ	АСОИ	65	37			37								1	1	1	4																				
	ИИ	40	23			23																															
	ИТКУТС	45	14			14																															
	ПЭ	30	2			2																															
итого	180	76	0	0	76								1	1	1	4																					
ФРЭ	МНЭТС	5	4			4																															
	РТ(ПРС)	10	4			4																															
	РТ(ПДР)	10	3			3																															
	РЭС	5	1			1																															
	РИ	5	1			1																															
	НИС	5	2			2																															
	ННЭ	5	2			2																															
	РЭИ	10	0			0																															
	ЭИИС	5	0			0																															
итого	60	17	0	0	17																																
ФКСИС	ПОИТ	110	135			135																															
	ВМСИС	90	20			20																															
	ИИП	100	118			118																															
	итого	300	273	0	0	273																															
К	ИКТ(СИ)	10	2			2																															
	ИКТ(СРМИ)	10	3			3																					</										

Рисунок 6.38 — Пример сгенерированного документа статистики

6.2.11 Выход из системы

Нажатие на кнопку меню «Выйти» вернет вас на страницу предупреждения о доступе к системе только сотрудникам приемной комиссии.

7 ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ ЭФФЕКТИВНОСТИ РАЗРАБОТКИ И ВНЕДРЕНИЯ ПС

7.1 Введение

Целью данного дипломного проекта является разработка программного средства автоматизации приемной кампании БГУИР.

Программное средство относится к категории продукции, которая реализуется по рыночным отпускным ценам. Выбор целесообразного с финансовой точки зрения проекта связан с его экономической оценкой и расчетом экономического эффекта. Именно поэтому разрабатываемые ПС должны иметь не только совершенную техническую и технологическую основу, но и быть выгодными с экономической точки зрения.

В данном разделе приведен расчет экономической эффективности внедрения автоматизированной системы подачи заявлений и зачисления в приемные комиссии университетов. Целью технико-экономического обоснования программного средства является определение экономической выгоды создания данного программного продукта и его дальнейшего использования в реальной жизни для выполнения бизнес-задач.

Разработанное программное средство позволяет исключить из процесса зачисления в университет человеческое вмешательство, упрощает работу по приему и учету документов в приемной комиссии университета, дает возможность автоматизировать формирование статистической информации приемной кампании вуза, а также предусматривает поэтапную организацию работы с документами, что позволит снизить временные затраты и упростить организацию процесса управления работой приемной комиссии университета. Использование системы значительно увеличит производительность труда сотрудников организации и экономическую эффективность организации.

Программное средство относится ко второй группе сложности. Категория новизны продукта – «В». Дополнительный коэффициент сложности ПО – 0,12.

Расчеты выполнены на основании методического пособия [20].

7.2 Расчет сметы затрат и цены ПС

7.2.1 Расчет трудоемкости разработки ПС и численности исполнителей
Исходные данные для расчета представлены в таблице 7.1.

Таблица 7.1 — Исходные данные

Наименование показателя	Буквенные обозначения	Единицы измерения	Количество
Коэффициент новизны	K_H	единиц	0,70
Группа сложности		единиц	2
Дополнительный коэффициент сложности	K_C	единиц	0,12

8 ОРГАНИЗАЦИЯ БЕЗОПАСНЫХ УСЛОВИЙ ТРУДА ИНЖЕНЕРОВ-ПРОГРАММИСТОВ В БГУИР

Охрана труда – важнейшая государственная задача, т. к. в ее основе лежит забота о здоровье и жизни людей. Значимость этой функции государства подтверждается принятым и действующим «Законом Республики Беларусь Об охране труда» (№356-З от 23.06.2008 г.). Определение охраны труда содержится в этом законе, а также в СТБ 18001 «Системы управления охраной труда. Общие требования». Согласно данным документам, охрана труда – это система обеспечения безопасности жизни и здоровья работающих в процессе трудовой деятельности, включающая правовые, социально-экономические, организационные, технические, психофизиологические, санитарно-гигиенические, лечебно-профилактические, реабилитационные и иные мероприятия и средства.

БГУИР сегодня – крупный научно-образовательный инновационный центр, в структуру которого входят 10 факультетов, 38 кафедр, научно-исследовательская часть, Институт повышения квалификации и переподготовки руководящих работников и специалистов по информационным технологиям и радиоэлектронике.

Общая численность работников университета – более 2000 человек, из них численность профессорско-преподавательского состава – более 750 человек, в том числе более 50 докторов наук и более 270 кандидатов наук. В научно-исследовательской части работают на постоянной основе более 220 человек. Обеспечение безопасных условий труда для всех сотрудников университета – важная часть политики руководства БГУИР.

Одной из стратегических целей вуза в системе менеджмента качества и системе охраны труда является обеспечение безопасных условий труда, сохранение жизни, здоровья и работоспособности работников в процессе их трудовой деятельности, профилактика производственного травматизма и профессиональных заболеваний.

Для достижения этой цели БГУИР определяет следующие основные направления политики университета в области охраны труда:

- предупреждать возникновение происшествий в университете (несчастных случаев, аварийных ситуаций, инцидентов и т.д.);
- обеспечить функционирование, проводить оценку и постоянно улучшать систему управления охраной труда в соответствии с требованиями государственного стандарта СТБ 18001;
- обеспечить соблюдение требований нормативных правовых и технических нормативных правовых актов в области охраны труда Республики Беларусь, а также других требований в области охраны труда, принятых на себя университетом;
- обеспечить снижение рисков производственного травматизма и профессиональных заболеваний.

В БГУИР разработано, принято и функционирует утвержденное ректором «Руководство по системе управления охраной труда» (Р СУОТ 01-2011 от 06.07.2011 г.).

Руководство университета гарантирует обеспечение необходимыми ресурсами, создание всех условий для выполнения Политики в области охраны труда и результативного функционирования системы управления охраной труда.

Выполнение Политики в области охраны труда – обязанность каждого сотрудника университета.

- повышенный уровень электромагнитных излучений;
- повышенная напряженность электрического поля;
- отсутствие или недостаток естественного света;
- недостаточная искусственная освещенность рабочей зоны;
- повышенная яркость света;
- повышенная контрастность;
- прямая и отраженная блескость;
- зрительное напряжение;
- монотонность трудового процесса;
- нервно-эмоциональные перегрузки.

Поэтому необходимо правильно организовать инженеру-программисту рабочее место, что позволит достигнуть максимального комфорта и предотвратит излишнюю нагрузку на организм, а также позволит увеличить производительность труда.

Исходя из общих принципов организации рабочего места инженера-программиста, в нормативно-методологических документах сформулированы требования к конструкции рабочего места. К ним относятся:

- требования к конструкции рабочего места;
- требования к размерам рабочей поверхности;
- требования к пространству для ног;
- требования к параметрам зон, в которых размещаются устройства ввода информации;
- требования к параметрам зон, в которых размещаются средства отображения информации;
- требования к взаимному расположению устройств ввода информации и средств отображения информации;
- требования к рабочему стулу в соответствии с техническими свойствами рабочего места и физиологическими параметрами человека.

Несоблюдение этих требований приведет к снижению работоспособности и утомляемости инженера-программиста.

Основным рабочим положением инженера-программиста является положение сидя. Основным оборудованием рабочего места являются:

- монитор;
- системный блок;
- клавиатура;
- компьютерная мышь;
- рабочий стол;
- стул;
- подставка для ног;
- шкафы и полки.

Согласно ГОСТ 12.2.032-78 рабочее место и взаимное расположение всех его элементов должно соответствовать антропометрическим, физиологическим и психологическим требованиям, как показано на рисунке 8.2.

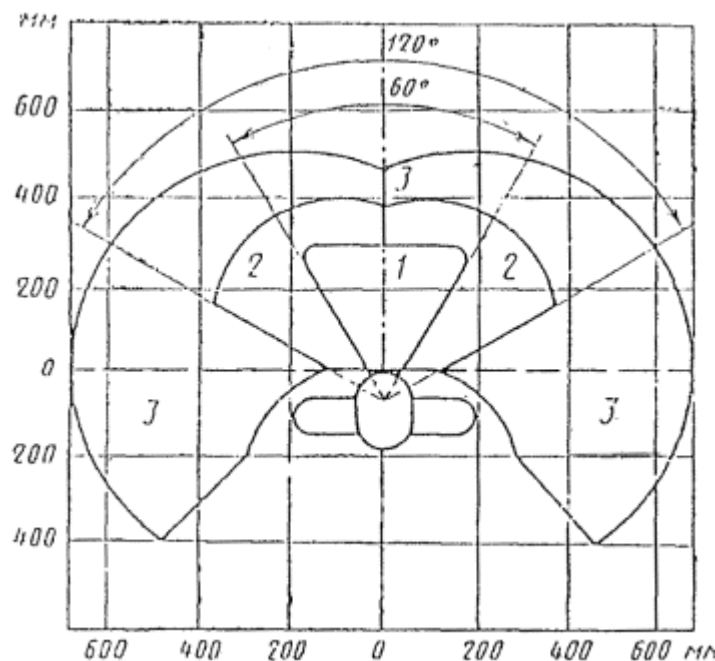


Рисунок 8.2 — Зоны для выполнения ручных операций и размещения органов управления

То, что требуется для выполнения работ чаще, расположено в зоне легкой досягаемости рабочего пространства. Таким образом, размещение оборудования на рабочем месте следующее:

- монитор установлен в центре зоны 3;
- клавиатура находится в зоне 1;
- мышь находится в зоне 1 или 2;
- системный блок расположен в правой части зоны 3;
- литература и документация, постоянно необходимые при работе, располагаются в левой части зоны досягаемости ладони – 2;
- литература, используемая в работе крайне редко, помещена в выдвижные ящики рабочего стола или на полках в зоне 3.

При проектировании письменного стола, изображенного на рисунке 3, в БГУИР учитываются следующие параметры:

- высота стола выбрана с учетом возможности сидеть свободно, в удобной позе (высота рабочих столов в университете 750 мм, длина стола составляет 1500 мм, а ширина 720 мм);
- поверхность для письма имеет не менее 40 мм в глубину и не менее 600 мм в ширину;
- под рабочей поверхностью предусмотрено пространство для ног (высота не менее 600 мм, ширина не менее 500 мм, глубина не менее 400 мм);
- нижняя часть стола сконструирована так, чтобы инженер-программист мог удобно сидеть, не был вынужден поджимать ноги;
- поверхность стола обладает свойствами, исключающими появление бликов в поле зрения работника;
- конструкция стола предусматривает наличие выдвижных ящиков (не менее 3 для хранения документации, листингов, канцелярских принадлежностей, личных вещей).

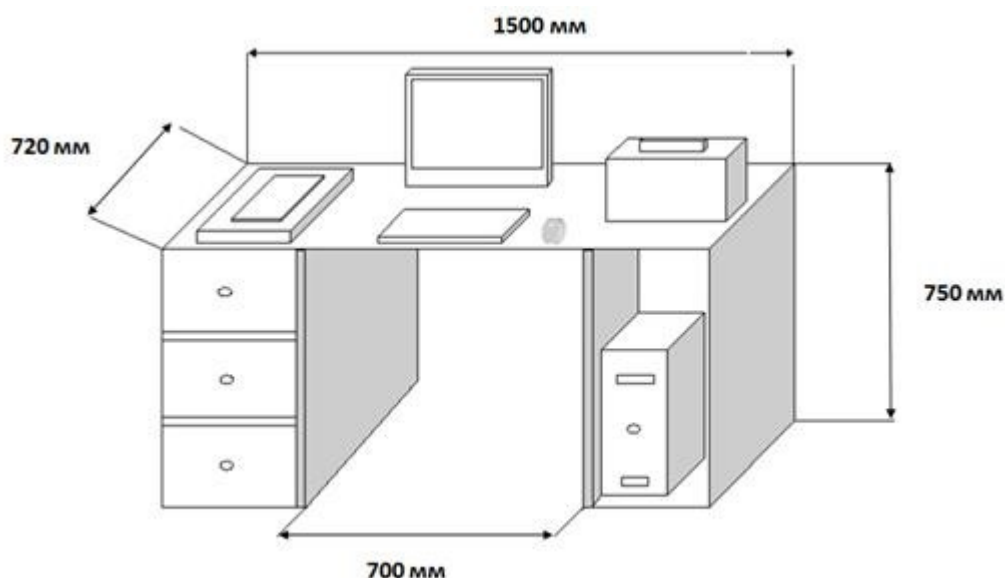


Рисунок 8.3 — Параметры письменного стола для инженера-программиста

Также в БГУИР для создания благоприятных условий труда уделяется внимание правильному эстетическому оформлению рабочих мест. Это делается для облегчения обстановки, влияющей на производительность труда. Окраска помещений и мебели благодаря своей нейтральности – малонасыщенные оттенки голубого цвета – способствует созданию благоприятных условий для зрительного восприятия.

В рабочем помещении для выполнения работником поставленных задач присутствует как естественное, так и искусственное освещение. Из осветительных приборов используются светильники типа ОД. Каждый светильник комплектуется двумя лампами. Размещаются светильники двумя рядами, по четыре в каждом ряду.

В настоящее время в вузе создается оптимальный микроклимат за счёт эффективной организации системы вентиляции, кондиционирования воздуха и отопительной системы.

Уровень шума не превышает допустимые значения.

Таким образом, изложенные выше условия обеспечивают комфортные условия труда для разработки инженерами-программистами БГУИР программного средства автоматизации деятельности приемной комиссии, снижают их утомляемость и риск возникновения профессиональных заболеваний.

9 ЗАКЛЮЧЕНИЕ

В ходе работы над дипломным проектом проанализированы системы упрощения подачи заявлений в различные органы. Исследованы разные направления и подходы к решению задач, связанных с разработкой программного средства для данных систем.

Проведен анализ предметной области, исследованы существующие аналоги. Результатом анализа явилось обобщение достоинств и недостатков существующих решений, которые учтены при разработке функциональных требований к разработанному программному средству. Наиболее часто встречающимся недостатком у имеющихся решений оказалось отсутствие комплексного решения всех поставленных требований. На отечественном рынке на сегодняшний день нет подобных средств, позволяющих гибко производить прием документов и автоматизировать зачисление вузы. Удобный пользовательский интерфейс поможет за короткие сроки обучиться работе с программой.

На основе функциональных требований было произведено проектирование программного средства. В нем представлены разработка архитектуры ПС, разработка модели базы данных, разработка алгоритма программного средства и алгоритмов отдельных модулей. В разделе разработки архитектуры ПС приведена обобщенная схема взаимодействия клиент-сервер и общая архитектура системы. Также детально рассмотрена схема алгоритма входа с блокировкой, схема алгоритма работы с программным средством, представлена диаграмма взаимодействия модулей системы по методологии IDEF0, а также диаграмма развертывания приложения.

Согласно требованиям были сформированы тестовые наборы, которые успешно пройдены в ходе тестовых испытаний программного средства. Успешность прохождения тестов показывает корректность работы программы с реальными данными, соответствие функциональным требованиям.

На завершающем этапе подробно описана методика использования программного средства.

Также в ходе работы над дипломным проектом рассмотрена экономическая сторона проектирования и разработки программного средства, рассчитаны экономический эффект от внедрения программного средства и показатели эффективности использования программного средства у пользователя. В результате расчётов подтвердилась целесообразность разработки. Рентабельность разработки составила почти 40%, а инвестиции, вложенные в разработку, окупаются за два года.

В разделе охраны труда отображены результаты исследования по обеспечению безопасных условий труда и комфортных условий микроклимата для разработки программного средства инженерами-программистами в БГУИР.

Главной целью при разработке программного средства было поставлено устранение основных недостатков существующих аналогов. В ходе работы над дипломным проектом эта цель была успешно достигнута: разработан минималистический, удобный, интуитивно понятный пользовательский интерфейс; программное средство имеет повышенную надежность хранения данных и отказоустойчивость благодаря использованию современных технологий защиты данных и грамотной архитектуре системы; разработанное ПС позволяет автоматизировать деятельность приемной комиссии, связанную с приемом, хранением и обработкой документов, как полученных от абитуриентов, так и внутренних, а также зачислением абитуриентов в ВУЗ.

Т.к. программа является веб-приложением, обеспечена поддержка всех современных браузеров.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- [1] Interfax.by [Электронный ресурс]. – Электронные данные. – Режим доступа: <http://www.interfax.by/article/80878>
- [2] Abitur.bsuir.by [Электронный ресурс]. – Электронные данные. – Режим доступа: <http://abitur.bsuir.by/>
- [3] Кодекс Республики Беларусь от 13.01.2011 N 243-З (ред. от 13.12.2011) "Кодекс Республики Беларусь об образовании" // Консультант Плюс – Беларусь [Электронный ресурс] / Нац. центр правовой информ. Республики Беларусь. – Минск, 2012.
- [4] Правила приема в высшие учебные заведения [Текст]: указ Президента Республики Беларусь от 07.02.2006 № 80 // Собрание законодательства. – Минск, 2006. – 26 с.
- [5] Порядок приема для получения высшего образования I ступени в учреждение образования «Белорусский государственный университет информатики и радиоэлектроники» на 2015 год [Текст]: постановление Министерства Образования Республики Беларусь от 13.04.2015 г. // Собрание законодательства. – Минск, 2015. – 11 с.
- [6] Система программ «1С:Образование 4.1. Школа 2.0» [Электронный ресурс]. – Электронные данные. – Режим доступа: <http://edu.1c.ru/>
- [7] Система электронной подачи заявления [Электронный ресурс]. – Электронные данные. – Режим доступа: <http://beldor.cent.r.by>
- [8] Портал государственных и муниципальных услуг Санкт-Петербурга [Электронный ресурс]. – Электронные данные. – Режим доступа: <http://gu.spb.ru>
- [9] Интернет ресурс приемной комиссии Российского Государственного Геологоразведочного университета [Электронный ресурс]. – Электронные данные. – Режим доступа: <http://mgri-rggru.ru/abitur/>
- [10] Флэнаган, Д. JavaScript. Подробное руководство / Д. Флэнаган. – М.: Символ-Плюс, 2008. – 992 с.
- [11] Agibetov, A. Essence of JSON / A. Agibetov. – L.: LAP Lambert Academic Publishing, 2013. – 52 с.
- [12] Скит, Д. C# для профессионалов: тонкости программирования, 3-е издание, новый перевод / Д. Скит – М.: Вильямс, 2014. – 608 с.
- [13] Ларман, К. Применение UML 2.0 и шаблонов проектирования – 3-е издание / К. Ларман – М.: Вильямс, 2006. – 736 с.
- [14] REST vs WebSocket Comparison and Benchmarks [Электронный ресурс]. – Электронные данные. – Режим доступа: <http://blog.arungupta.me/rest-vs-websocket-comparison-benchmarks>
- [15] Thomas, S. SSL & TLS Essentials: Securing the Web – 1-st. / S. Thomas – L.: Wiley, 2000. – 224 с.
- [16] Фримен, Э. Изучаем HTML, XHTML и CSS / Э. Фримен – П.: Питер, 2010. – 656 с.
- [17] Петкович, Д. Microsoft SQL Server 2008. Руководство для начинающих / Д. Петкович – С.: БХВ-Петербург, 2009. – 752 с.
- [18] Фримен, А. ASP.NET MVC 5 с примерами на C# 5.0 для профессионалов, 5-е издание / А. Фримен – М.: Вильямс, 2014. – 736 с.
- [19] Вулстон, Д. Аяx и платформа .NET 2.0 для профессионалов / Д. Вулстон – М.: Вильямс, 2007. — 464 с.
- [20] Техничко-экономическое обоснование дипломных проектов: Методическое пособие для студентов всех специальностей БГУИР. В 4-ч. Ч. 4: Проекты программного обеспечения / В. А. Палицын. – Минск: БГУИР, 2006 г. –

76 с.

[21] Алексеев, С.В. Гигиена труда / С.В. Алексеев, В.Р. Усенко – Санкт-Петербург, 2009. – 24 с.

ПРИЛОЖЕНИЕ А
(ОБЯЗАТЕЛЬНОЕ)
ТЕКСТ ПРОГРАММНОГО МОДУЛЯ СЕРВЕРА

```
namespace ASPZiZ_PK.Areas.Admin.Controllers
{
    public class SpecialitiesController :
ControllerExtensions
    {
        //
        // GET: /Admin/Specialities/

        public ActionResult Index()
        {
            return RedirectToAction("OKSK");
        }

        #region OKSK
        public ActionResult OKSK(string command,
string id)
        {
            string viewPath = "OKSK/";
            string actionName = CurrentAction;
            short idInt = -1;
            short.TryParse(id, out idInt);
            switch (command)
            {
                case "New": { return View(viewPath +
"Edit", OKSKRepos.New()); }
                case "Edit":
                {
                    OKSK_Speciality_Model model =
OKSKRepos.Get(idInt);
                    if (model == null) return
RedirectToAction("OKSK", new { command = "List" });
                    return View(viewPath + "Edit",
model);
                }
                case "Details":
                {
                    OKSK_Speciality_Model model =
OKSKRepos.Get(idInt);
                    if (model == null) return
RedirectToAction("OKSK", new { command = "List" });
                    ViewBag.Specialities =
SpecRepos.GetByOKSKId(model.Id);
                    return View(viewPath +
"Details", model);
                }
                case "List": return View(viewPath +
"List", OKSKRepos.GetList());
                default: { return
```

```

RedirectToAction("OKSK", new { command = "List" }); }
    }

    [HttpPost]
    [ValidateAntiForgeryToken]
    public ActionResult
OKSK_Save(OKSK_Speciality_Model model)
    {
        OKSKRepos.Save(model);
        return RedirectToAction("OKSK", new {
command = "Details", id = model.Id });
    }

    [HttpPost]
    [ValidateAntiForgeryToken]
    public ActionResult OKSK_Delete(short id)
    {
        KeyValuePair<bool, string> result =
OKSKRepos.Delete(id);
        if (result.Key)
            return RedirectToAction("OKSK", new {
command = "List" });
        else
            return RedirectToAction("OKSK", new {
command = "Details", id = id });
    }
    #endregion

    public ActionResult Faculty(string command,
string id)
    {
        string viewPath = "Faculty/";
        string actionName = CurrentAction;
        short idInt = -1;
        short.TryParse(id, out idInt);
        switch (command)
        {
            //case "New": { return View(viewPath +
"Edit", FacultyRepos.New()); }
            case "Edit":
            {
                FacultyModel model =
FacultyRepos.GetById(idInt);
                if (model == null) return
RedirectToAction(actionName, new { command = "List" });
                return View(viewPath + "Edit",
model);
            }
            case "Details":
            {
                FacultyModel model =

```

```

FacultyRepos.GetById(idInt);
        if (model == null) return
RedirectToAction(actionName, new { command = "List" });
        ViewBag.Specialities =
SpecRepos.GetByFaciltyId(model.Id);
        return View(viewPath +
"Details", model);
    }
    case "List": return View(viewPath +
"List", FacultyRepos.GetList());
    default: { return
RedirectToAction(actionName, new { command = "List"
}); }
    }
}

[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Faculty_Save(FacultyModel
model)
{
    //FacultyRepos.Save(model);
    return RedirectToAction("Faculty", new {
command = "Details", id = model.Id });
}

}

[Authorize]
public class UsersController : ControllerExtensions
{
    private string _moduleName = "Абитуриенты";
    public ActionResult Index()
    {
        return RedirectToAction("List");
    }

    public ActionResult List()
    {
        UserModel userCurr = CurrentUser;
        if (userCurr != null) { if
(!userCurr.Permissions.Adm) return
RedirectToLoginPage(); }
        else RedirectToLoginPage();

        IEnumerable<UserModel> users =
MembershipRepos.GetUsers();
        return View(users);
    }

    public ActionResult Details(string id)
    {

```

```

        UserModel userCurr = CurrentUser;
        if (userCurr != null) { if
(!userCurr.Permissions.Adm) return
RedirectToLoginPage(); }
        else RedirectToLoginPage();

        ViewBag.IsNew = false;
        return
View(MembershipRepos.GetUserById(id));
    }

    [HttpPost]
    public ActionResult Save(UserModel model)
    {
        UserModel userCurr = CurrentUser;
        if (userCurr != null) { if
(!userCurr.Permissions.Adm) return
RedirectToLoginPage(); }
        else RedirectToLoginPage();

        if
(!string.IsNullOrEmpty(model.Password)) {
MembershipRepos.ChangePass(model.Login,
model.Password); }
        ProfileUser profile =
ProfileUser.GetProfile(model.Login);
        profile.IsGroup = false;
        profile.FullName = model.UserName;
        profile.RequirePasswordChange = false;
        profile.Save();
        MembershipRepos.SaveUserPermissions(model);
        return RedirectToAction("List");
    }

    public ActionResult New()
    {
        UserModel userCurr = CurrentUser;
        if (userCurr != null) { if
(!userCurr.Permissions.Adm) return
RedirectToLoginPage(); }
        else RedirectToLoginPage();

        UserModel model = new UserModel();
        model.Permissions = new PermissionsModel();
        ViewBag.IsNew = true;
        return View("Details", model);
    }

    [HttpPost]
    public ActionResult Create(UserModel model)

```



```

        {
            UserModel userCurr = CurrentUser;
            if (userCurr != null) { if
(!userCurr.Permissions.Adm) return
RedirectToLoginPage(); }
            else RedirectToLoginPage();

            MembershipCreateStatus createStatus;
            MembershipUser mu =
Membership.CreateUser(model.Login, model.Password,
model.Login, null, null, true, null, out createStatus);

            if (createStatus ==
MembershipCreateStatus.Success)
            {
                ProfileUser profile =
ProfileUser.GetProfile(model.Login);
                profile.IsGroup = false;
                profile.FullName = model.UserName;
                profile.RequirePasswordChange = false;
                profile.Save();
                model.UserID =
Guid.Parse(mu.ProviderUserKey.ToString());
MembershipRepos.SaveUserPermissions(model);
            }
            else
            {
                ViewBag.IsNew = true;
                return View("Details", model);
            }
            return RedirectToAction("List");
        }

        public ActionResult Delete(string id)
        {
            UserModel userCurr = CurrentUser;
            if (userCurr != null) { if
(!userCurr.Permissions.Adm) return
RedirectToLoginPage(); }
            else RedirectToLoginPage();

            MembershipRepos.DeleteUser(id);
            return RedirectToAction("List");
        }
    }
}

namespace ASPZiZ_PK.Controllers
{
    [Authorize]

```

```

    public class AbiturController :
ControllerExtensions
    {
        private string SessionId_Filter =
"asdfji238jcSIo";
        private bool IsAccess(ref
AbiturListFilterModel filter, UserModel userCurr)
        {
            bool isAccess = false;
            bool isFirst = true;
            while (!isAccess && filter.RegStepId <=
254)
            {
                switch
((RegistrationStep)filter.RegStepId)
                {
                    case RegistrationStep.NotActivate:
                    case RegistrationStep.SiteActivate:
                    case RegistrationStep.SiteFull:
                    case RegistrationStep.SiteNotFull:
if (userCurr.Permissions.Site) isAccess = true; else
if (isFirst) { isFirst = false; filter.RegStepId =
(byte)RegistrationStep.SiteActivate; } else
filter.RegStepId++; break;

                    case RegistrationStep.Ochered: if
(userCurr.Permissions.Och) isAccess = true; else if
(isFirst) { isFirst = false; filter.RegStepId =
(byte)RegistrationStep.SiteActivate; } else
filter.RegStepId++; break;
                    case RegistrationStep.BSUIR:
                    case RegistrationStep.BSUIRno:
                    case RegistrationStep.BSUIRyes: if
(userCurr.Permissions.Univ) isAccess = true; else if
(isFirst) { isFirst = false; filter.RegStepId =
(byte)RegistrationStep.SiteActivate; } else
filter.RegStepId++; break;

                    case RegistrationStep.ZabralDok:
if (userCurr.Permissions.Zabr) isAccess = true; else
if (isFirst) { isFirst = false; filter.RegStepId =
(byte)RegistrationStep.SiteActivate; } else
filter.RegStepId++; break;
                    default: filter.RegStepId++; break;
                }
            }
            return isAccess;
        }

        public ActionResult Index()
        {

```

```

        return RedirectToAction("List");
    }
    public ActionResult List(string regStep,
string[] sortParams)
    {
        byte rs;
        AbiturListFilterModel filter = null;
        if (Session != null)
        {
            object tmp = Session[SessionId_Filter];
            if (tmp != null)
                filter =
(AbiturListFilterModel)tmp;
        }
        if (byte.TryParse(regStep, out rs) ==
false) return RedirectToAction("List", new { regStep =
(byte)RegistrationStep.Ochered }); // rs =
(byte)RegistrationStep.Ochered;
        if (filter == null) { filter = new
AbiturListFilterModel() { RegStepId = rs }; }

        UserModel userCurr = CurrentUser;
        if (userCurr != null) { if (!IsAccess(ref
filter, userCurr)) return RedirectToLoginPage(); }
        else RedirectToLoginPage();

        AbiturListModel model = new
AbiturListModel()
        {
            Abiturs = AbiturRepos.GetList(filter),
            Filter = filter
        };
        FillViewData_ListAction(filter, userCurr);
        if (Session != null)
        {
            Session.Remove(SessionId_Filter);
            Session.Add(SessionId_Filter, filter);
        }
        return View(model);
    }

    private void
FillViewData_ListAction(AbiturListFilterModel filter,
UserModel currUser)
    {
        ViewData["RegStepList"] = new
SelectList(DDL_Model.GetEnum_RegistrationStep(currUser),
"Id", "Name", filter.RegStepId.ToString());
        ViewData["SrokObuchList"] = new
SelectList(SrokObuchRepos.GetList(true), "Id", "Name",
(filter.SrokObId.HasValue ?
filter.SrokObId.Value.ToString() : "0"));
    }

```

```

        ViewData["FormaObuchList"] = new
SelectList(FormaObuchRepos.GetList_Filter(filter.IsPlatn,
filter.SrokObId, filter.isCelev, true), "Id", "Name",
(filter.FormObId.HasValue ?
filter.FormObId.Value.ToString() : "0"));
        ViewData["FacultyList"] = new
SelectList(FacultyRepos.GetList_Filter(filter.IsPlatn,
filter.FormObId, filter.SrokObId, filter.isCelev),
"Id", "Name", (filter.FacId.HasValue ?
filter.FacId.Value.ToString() : "0"));
        ViewData["SpecialityList"] = new
SelectList(SpecRepos.GetList_Filter(filter.IsPlatn,
filter.FormObId, filter.SrokObId, filter.isCelev,
filter.FacId), "Id", "Name", (filter.SpecId.HasValue
? filter.SpecId.Value.ToString() : "0"));
    }

    [HttpPost]
    public ActionResult
PostActionToUpdateAbiturs(AbiturListFilterModel
filter, string[] sortParams)
    {
        UserModel userCurr = CurrentUser;
        if (userCurr != null) { if (!IsAccess(ref
filter, userCurr)) return RedirectToLoginPage(); }
        else RedirectToLoginPage();

        AbiturListModel model = new
AbiturListModel()
        {
            Abiturs = AbiturRepos.GetList(filter),
            Filter = filter
        };
        if (Session != null)
        {
            Session.Remove(SessionId_Filter);
            Session.Add(SessionId_Filter, filter);
        }
        return PartialView("partial_AbiturList",
model);
    }

    [HttpPost]
    public ActionResult ReloadExams(short specId)
    {
        List<AbiturExamModel> exams =
ExamRepos.GetExams(new short[1] { specId });
        ViewData["ExamTypes"] = new
SelectList(ExamRepos.GetExamTypes(), "Id", "Name");
        return PartialView("Edit_Exams_Exams",
exams);
    }

```

```

    }

    public ActionResult New(string regStep)
    {
        UserModel userCurr = CurrentUser;
        if (userCurr != null) { if
(!userCurr.Permissions.Och ||
!userCurr.Permissions.Och_New) return
RedirectToLoginPage(); }
        else RedirectToLoginPage();

        ProfileAbiturModel model =
AbiturRepos.New();
        FillViewData(model);
        ViewBag.tabIndex = "0";
        return View("Edit", model);
    }

    public ActionResult Delete(string regStep,
string id)
    {
        byte rs = 0;
        if (byte.TryParse(regStep, out rs) ==
false) { return RedirectToAction("List"); }

        Guid abtId = Guid.Empty;
        if (Guid.TryParse(id, out abtId) == false)
return RedirectToAction("List");

        UserModel userCurr = CurrentUser;
        if (userCurr != null)
        {
            if (((RegistrationStep)rs ==
RegistrationStep.Ochered && userCurr.Permissions.Och
&& userCurr.Permissions.Och_Del)
                || ((RegistrationStep)rs ==
RegistrationStep.ZabralDok &&
userCurr.Permissions.Zabr &&
userCurr.Permissions.Zabr_Del))
            {
                KeyValuePair<bool, string> res =
AbiturRepos.Delete(abtId, (RegistrationStep)rs);
                return RedirectToAction("List");
            }
        }

        return RedirectToLoginPage();
    }

    public ActionResult Details(string regStep,
string id, string tab)
    {

```

```

        byte rs = 0;
        if (byte.TryParse(regStep, out rs) ==
false) { return RedirectToAction("List"); }

        Guid abtId = Guid.Empty;
        if (Guid.TryParse(id, out abtId) == false)
return RedirectToAction("List");

        ProfileAbiturModel model =
AbiturRepos.Details(abtId, (RegistrationStep)rs);

        if (model.Predstavitel == null)
            model.Predstavitel = new
Dogovor_Predstavitel();
        ViewData["PassportList"] = new
SelectList(PassportRepos.GetList_Types(), "Id",
"Name", model.Predstavitel.PassportTypeId.ToString());
        ViewBag.tabIndex =
(string.IsNullOrEmpty(tab) == true) ? "0" : tab;
        if (model.RegStep == RegistrationStep.BSUIR
            || model.RegStep ==
RegistrationStep.BSUIRno
            || model.RegStep ==
RegistrationStep.BSUIRyes)
            FillViewData_Zachisl(model);
        return View(model);
    }

    [HttpGet]
    public ActionResult Edit(byte regStep, string
id, string tab)
    {
        Guid abtId = Guid.Empty;
        if (Guid.TryParse(id, out abtId) == false)
return RedirectToAction("List");
        UserModel userCurr = CurrentUser;
        RegistrationStep rs =
(RegistrationStep)regStep;
        if (userCurr != null)
        {
            if ((regStep <=
(byte)RegistrationStep.SiteFull &&
userCurr.Permissions.Site &&
userCurr.Permissions.Site_Edit)
                || (rs == RegistrationStep.Ochered
&& userCurr.Permissions.Och &&
userCurr.Permissions.Och_Edit)
                || (rs == RegistrationStep.BSUIR
|| rs == RegistrationStep.BSUIRno || rs ==
RegistrationStep.BSUIRyes)
                    &&
userCurr.Permissions.Univ &&

```

```

userCurr.Permissions.Univ_Edit))
    {
        ProfileAbiturModel model =
AbiturRepos.Edit(abtId, rs);
        FillViewData(model);
        ViewBag.tabIndex =
(string.IsNullOrEmpty(tab) == true) ? "0" : tab;
        return View(model);
    }
    }
    return RedirectToLoginPage();
}
private void FillViewData(ProfileAbiturModel
model)
{
    if (model == null) return;
    ViewData["ObrUrovenList"] = new
SelectList(ObrazovRepos.GetList_ObrUroven(), "Id",
"Name", model.Obr_UrovenId.ToString());
    ViewData["ObrUchregdList"] = new
SelectList(ObrazovRepos.GetList_ObrUchregd(model.Obr_UrovenId
"Id", "Name", model.Obr_UchregId.ToString());
    ViewData["ObrDokTypeList"] = new
SelectList(ObrazovRepos.GetList_ObrDokType(model.Obr_UrovenId
"Id", "Name", model.Obr_DokTypeId.ToString());

    ViewData["PassportList"] = new
SelectList(PassportRepos.GetList_Types(), "Id",
"Name", model.PassportTypeId.ToString());

    ViewData["TipRodstva_Otec"] =
RoditelModel.GetTipRodstvaList(true, (model.Otec !=
null ? (model.Otec.TipRodstva ?? " ") : " "));
    ViewData["TipRodstva_Mat"] =
RoditelModel.GetTipRodstvaList(false, (model.Mat !=
null ? (model.Mat.TipRodstva ?? " ") : " "));

    ViewData["ExamTypes"] = new
SelectList(ExamRepos.GetExamTypes(), "Id", "Name");

    FillViewData_Zayavlenie(model);
}

private void
FillViewData_Zayavlenie(ProfileAbiturModel model)
{
    if (model.Zayvlenie == null) return;
    bool isAnySpeciality =
model.Zayvlenie.Any();
    ViewData["FormaObuchList"] = new
SelectList(FormaObuchRepos.GetList(), "Id", "Name",
(isAnySpeciality ?

```

```

model.Zayvlenie[0].FormaObuchId.ToString(CultureInfo.InvariantCulture
: "0"));
        ViewData["SrokObuchList"] = new
SelectList(SrokObuchRepos.GetList(true), "Id", "Name",
(isAnySpeciality ?
model.Zayvlenie[0].SrokObuchId.ToString(CultureInfo.InvariantCulture
: "0"));
        ViewData["SpecSSUZList"] = new
SelectList(SSUZRepos.GetList(true), "Id", "Name",
(model.SSUZ_SpecId.HasValue ?
model.SSUZ_SpecId.Value.ToString(CultureInfo.InvariantCulture
: "0"));

        if (isAnySpeciality == false) return;
        SpecialnostModel spDef =
model.Zayvlenie[0];
        List<FormaObuchModel> foList =
FormaObuchRepos.GetList(spDef.isPlatn,
spDef.SrokObuchId, spDef.isCelev, false,
model.SSUZ_SpecId ?? 0, spDef.SpecialnostId);
        List<FacultyModel> fcltList;
        List<SpecModelDDL> specList;

        if (foList.Count == 1 && foList[0].Id !=
0) fcltList = FacultyRepos.GetList(spDef.isPlatn,
spDef.FormaObuchId, spDef.SrokObuchId, spDef.isCelev,
model.SSUZ_SpecId ?? 0, spDef.SpecialnostId);
        else fcltList = new List<FacultyModel>() {
new FacultyModel() { Id = 0, Name = "" } };

        if (fcltList.Count == 1 && fcltList[0].Id
!= 0) specList = SpecRepos.GetList(spDef.isPlatn,
spDef.FormaObuchId, spDef.SrokObuchId, spDef.isCelev,
spDef.FacultetId, model.SSUZ_SpecId ?? 0,
spDef.SpecialnostId);
        else specList = new List<SpecModelDDL>() {
new SpecModelDDL() { Id = 0, Name = "" } };

        foreach (SpecialnostModel z in
model.Zayvlenie)
        {
            if (z.SpecialnostId != 0)
            {
                //ViewData["FormaObuch_" +
z.Prioritet] = new SelectList(foList, "Id", "Name",
z.FormaObuchId.ToString());
                ViewData["Faculty_" + z.Prioritet]
= new SelectList(FacultyRepos.GetList(z.isPlatn,
z.FormaObuchId, z.SrokObuchId, z.isCelev,
model.SSUZ_SpecId ?? 0, z.SpecialnostId), "Id",
"Name", z.FacultetId.ToString());
                ViewData["Speciality_" +

```



```

z.Prioritet] = new
SelectList(SpecRepos.GetList(z.isPlatn,
z.FormaObuchId, z.SrokObuchId, z.isCelev,
z.FacultetId, model.SSUZ_SpecId ?? 0,
z.SpecialnostId), "Id", "Name",
z.SpecialnostId.ToString());
        }
        else
        {
            //ViewData["FormaObuch_" +
z.Prioritet] = new SelectList(foList, "Id", "Name",
"0");
            ViewData["Faculty_" + z.Prioritet]
= new SelectList(fcltList, "Id", "Name", "0");
            ViewData["Speciality_" +
z.Prioritet] = new SelectList(specList, "Id", "Name",
"0");
        }
    }

    private void
FillViewData_Zachisl(ProfileAbiturModel model)
    {
        if (model.Zayvlenie == null) return;

        if (model.Zayvlenie.Count <= 0) return;
        SpecialnostModel spDef =
(model.Zach_Specialnost == null ? model.Zayvlenie[0] :
model.Zach_Specialnost);

        ViewData["SrokObuchList"] = new
SelectList(SrokObuchRepos.GetList(true), "Id", "Name",
spDef.SrokObuchId.ToString());

        List<FormaObuchModel> foList =
FormaObuchRepos.GetList(spDef.isPlatn,
spDef.SrokObuchId, spDef.isCelev, false, 0,
spDef.SpecialnostId);
        List<FacultyModel> fcltList;
        List<SpecModelDDL> specList = new
List<SpecModelDDL>();

        if ((foList.Count == 1 && foList[0].Id !=
0) || spDef != null) fcltList =
FacultyRepos.GetList(spDef.isPlatn,
spDef.FormaObuchId, spDef.SrokObuchId, spDef.isCelev,
0, spDef.SpecialnostId);
        else fcltList = new List<FacultyModel>() {
new FacultyModel() { Id = 0, Name = "" } };

        if ((fcltList.Count == 1 && fcltList[0].Id

```

```

!= 0) || spDef != null) specList =
SpecRepos.GetList(spDef.isPlatn, spDef.FormaObuchId,
spDef.SrokObuchId, spDef.isCelev, spDef.FacultetId,
model.SSUZ_SpecId ?? 0, spDef.SpecialnostId);
    else specList = new List<SpecModelDDL>() {
new SpecModelDDL() { Id = 0, Name = "" } };

        ViewData["FormaObuch_Zach"] = new
SelectList(foList, "Id", "Name",
(model.Zach_Specialnost != null ?
model.Zach_Specialnost.FormaObuchId.ToString() : "0"));
        ViewData["Faculty_Zach"] = new
SelectList(fcList, "Id", "Name",
(model.Zach_Specialnost != null ?
model.Zach_Specialnost.FacultetId.ToString() : "0"));
        ViewData["Speciality_Zach"] = new
SelectList(specList, "Id", "Name",
(model.Zach_Specialnost != null ?
model.Zach_Specialnost.SpecialnostId.ToString() :
"0"));
    }

    [HttpPost]
    [ActionName("Edit")]
    public ActionResult Save(ProfileAbiturModel
model, string tabIndex)
    {
        UserModel userCurr = CurrentUser;
        if (userCurr != null)
        {
            RegistrationStep rs =
(RegistrationStep)model.RegStepId;
            if (MembershipRepos.CanEditInSite(rs,
userCurr) || MembershipRepos.CanEditInOchered(rs,
userCurr) || MembershipRepos.CanEditInVuz(rs,
userCurr))
            {
                model.AbiturId = model.AbiturId;
                model.Zayvlenie =
AbiturRepos.FixErrorsInSpecialities(model.Zayvlenie);

                if (Request != null &&
Request.Form != null)
                {
                    short ssuzSpecId = 0;
                    if
(short.TryParse(Request.Form["SpecSSUZ"], out
ssuzSpecId)) { model.SSUZ_SpecId = ssuzSpecId; } else
{ model.SSUZ_SpecId = (short?)null; }
                }
                model =
AbiturRepos.PrepareForSave(model);

```

```

        if (ModelState.IsValid)
        {
            KeyValuePair<bool, string>
result = AbiturRepos.Save(model, rs);
            return
RedirectToAction("Details", new { regStep = (byte)rs,
id = model.AbiturId, tab = tabIndex });
        }
        else
        {
            model =
AbiturRepos.PrepareForEdit(model);
            FillViewData(model);
            return View("Edit", model);
        }
    }
}
return RedirectToLoginPage();
}

[OutputCache(Duration = 3600, VaryByParam =
"urovenId")]
[HttpGet]
public JsonResult GetObrDDL(byte urovenId)
{
    var resDokType = new
SelectList(ObrazovRepos.GetList_ObrDokType(urovenId),
"Id", "Name", "0");
    var resUchr = new
SelectList(ObrazovRepos.GetList_ObrUchregd(urovenId),
"Id", "Name", "0");
    var resList = new[] { resDokType, resUchr
};
    var jr = Json(resList,
JsonRequestBehavior.AllowGet);
    return jr;
}

//[OutputCache(Duration = 3600, VaryByParam =
"platn;srokObuchId;celev;specId;ekzId_1;ekzId_2")]
[HttpGet]
public JsonResult GetFormObuch(bool platn,
byte srokObuchId, bool celev, byte? specId, short?
ssuzSpecId)
{
    var resList =
FormaObuchRepos.GetList(platn, srokObuchId, celev,
true, ssuzSpecId ?? 0, specId);
    JsonResult jr = Json(new
SelectList(resList, "Id", "Name", 0),
JsonRequestBehavior.AllowGet);
    return jr;
}

```



```

        RegistrationStep regStepE =
(RegistrationStep)rs;
        if ((regStepE <=
RegistrationStep.SiteFull && userCurr.Permissions.Och
&& userCurr.Permissions.Site_AddToOcher)
            || (regStepE ==
RegistrationStep.ZabralDok &&
userCurr.Permissions.Zabr &&
userCurr.Permissions.Zabr_AddToOcher))
        {
            KeyValuePair<bool, string> res =
AbiturRepos.AddToOchered((RegistrationStep)rs, abtId);
            JsonResult js = Json(new
KeyValuePair<bool, string>(true, "Абитуриент добавлен
в очередь."), JsonRequestBehavior.AllowGet);
            return js;
        }
    }
    JsonResult jsEr = Json(new
KeyValuePair<bool, string>(false, "В доступе
отказано."), JsonRequestBehavior.AllowGet);
    return jsEr;
}

[HttpGet]
public JsonResult GetSpecCode(string regStep,
string id, string s)
{
    byte rs = 0;
    if (byte.TryParse(regStep, out rs) ==
false) { return Json(new KeyValuePair<bool,
string>(false, "Ошибка. Неверный шаг регистрации.)); }

    Guid abtId = Guid.Empty;
    if (Guid.TryParse(id, out abtId) == false)
return Json(new KeyValuePair<bool, string>(false,
"Ошибка. Неверный идентификатор абитуриента.));

    UserModel userCurr = CurrentUser;
    if (userCurr != null)
    {
        RegistrationStep regStepE =
(RegistrationStep)rs;
        if (regStepE ==
RegistrationStep.Ochered && userCurr.Permissions.Och
&& userCurr.Permissions.Och_AddToBsuir)
        {
            KeyValuePair<bool, string> res =
AbiturRepos.GetSpecCodeFirstSpec((RegistrationStep)rs,
abtId);

            JsonResult js = Json(new

```

```

KeyValuePair<bool, string>(res.Key, res.Value),
JsonRequestBehavior.AllowGet);
        return js;
    }
}
JsonResult jsEr = Json(new
KeyValuePair<bool, string>(false, "В доступе
отказано."), JsonRequestBehavior.AllowGet);
return jsEr;
}

[HttpGet]
public JsonResult ZachToBsuir(string regStep,
string id, string specId, string s)
{
    byte rs = 0;
    if (byte.TryParse(regStep, out rs) ==
false) { return Json(new KeyValuePair<bool,
string>(false, "Ошибка. Неверный шаг регистрации.)); }

    Guid abtId = Guid.Empty;
    if (Guid.TryParse(id, out abtId) == false)
return Json(new KeyValuePair<bool, string>(false,
"Ошибка. Неверный идентификатор абитуриента.));

    UserModel userCurr = CurrentUser;
    if (userCurr != null)
    {
        RegistrationStep regStepE =
(RegistrationStep)rs;
        if ((regStepE ==
RegistrationStep.BSUIR || regStepE ==
RegistrationStep.BSUIRno || regStepE ==
RegistrationStep.BSUIRyes)
            && userCurr.Permissions.Univ &&
userCurr.Permissions.Univ_BsuirZach)
        {
            short specIdZach;
            KeyValuePair<bool, string> res =
AbiturRepos.ZachToBSUIR(abtId, (short.TryParse(specId,
out specIdZach) ? specIdZach : (short?)null));
            JsonResult js = Json(new
KeyValuePair<bool, string>(true, "Зачисление
проведено."), JsonRequestBehavior.AllowGet);
            return js;
        }
    }
    JsonResult jsEr = Json(new
KeyValuePair<bool, string>(false, "В доступе
отказано."), JsonRequestBehavior.AllowGet);
    return jsEr;
}

```

```

[HttpGet]
public JsonResult AddToBsuir(string regStep,
string id, string abtCode, string s)
{
    byte rs = 0;
    if (byte.TryParse(regStep, out rs) ==
false) { return Json(new KeyValuePair<bool,
string>(false, "Ошибка. Неверный шаг регистрации.)); }

    Guid abtId = Guid.Empty;
    if (Guid.TryParse(id, out abtId) == false)
return Json(new KeyValuePair<bool, string>(false,
"Ошибка. Неверный идентификатор абитуриента.));

    UserModel userCurr = CurrentUser;
    if (userCurr != null)
    {
        RegistrationStep regStepE =
(RegistrationStep)rs;
        if (regStepE ==
RegistrationStep.Ochered && userCurr.Permissions.Och
&& userCurr.Permissions.Och_AddToBsuir)
        {
            KeyValuePair<bool, string> res =
AbiturRepos.AddToBSUIR((RegistrationStep)rs, abtId,
abtCode);

            JsonResult js = Json(new
KeyValuePair<bool, string>(true, "Документы
приняты."), JsonRequestBehavior.AllowGet);
            return js;
        }
    }
    JsonResult jsEr = Json(new
KeyValuePair<bool, string>(false, "В доступе
отказано."), JsonRequestBehavior.AllowGet);
    return jsEr;
}

[HttpGet]
public JsonResult RemoveFromBsuir(string
regStep, string id, string s)
{
    byte rs = 0;
    if (byte.TryParse(regStep, out rs) ==
false) { return Json(new KeyValuePair<bool,
string>(false, "Ошибка. Неверный шаг регистрации.)); }

    Guid abtId = Guid.Empty;
    if (Guid.TryParse(id, out abtId) == false)
return Json(new KeyValuePair<bool, string>(false,
"Ошибка. Неверный идентификатор абитуриента.));

```

```

        UserModel userCurr = CurrentUser;
        if (userCurr != null)
        {
            RegistrationStep regStepE =
(RegistrationStep)rs;
            if ((regStepE ==
RegistrationStep.BSUIR || regStepE ==
RegistrationStep.BSUIRno || regStepE ==
RegistrationStep.BSUIRyes)
                && userCurr.Permissions.Univ
&& userCurr.Permissions.Univ_RemoveFromBsuir)
            {
                KeyValuePair<bool, string> res =
AbiturRepos.RemoveFromBsuir((RegistrationStep)rs,
abtid);
                JsonResult js = Json(new
KeyValuePair<bool, string>(false, "Документы
выданы."), JsonRequestBehavior.AllowGet);
                return js;
            }
        }
        JsonResult jsEr = Json(new
KeyValuePair<bool, string>(false, "В доступе
отказано."), JsonRequestBehavior.AllowGet);
        return jsEr;
    }

    [HttpGet]
    public JsonResult GetCertfCT(string id, string
passSer, string passNom, string s)
    {
        List<CertificatModel> cert =
AbiturRepos.GetCertf(passSer, passNom);
        JsonResult js = Json(cert,
JsonRequestBehavior.AllowGet);
        return js;
    }

    [HttpPost]
    public ActionResult
PostActionToUpdateSpec(byte specId)
    {
        SpecialnostModel spModel =
SpecRepos.GetSpec(specId);
        var prof = new ProfileAbiturModel() {
Zayvlenie = new List<SpecialnostModel>() };
        if (spModel != null)
        {
            int countSpec =

```



```

SpecRepos.GetCountSpec(spModel.GroupId);
    prof.isCelev = spModel.isCelev;
    prof.isPlatn = spModel.isPlatn;
    prof.CountSpecInGroup = countSpec;
    spModel.Prioritet = 1;
    prof.Zayvlenie.Add(spModel);
    prof =
AbiturRepos.PrepareForEdit(prof);

        FillViewData_Zayavlenie(prof);
    }
    return
 PartialView("Edit_Specialties_Specs", prof);
}

        public ActionResult Print_Zayvlenie(string s,
string regStep, string id)
    {
        byte rs = 0;
        if (byte.TryParse(regStep, out rs) ==
false) { return RedirectToAction("List"); }

        Guid abtId = Guid.Empty;
        if (Guid.TryParse(id, out abtId) == false)
return RedirectToAction("List");
        ProfileAbiturModel model =
AbiturRepos.Edit(abtId, (RegistrationStep)rs);

        byte[] result =
PrintRepos.Zayvlenie(model);

        var output = new MemoryStream();
        output.Write(result, 0, result.Length);
        output.Position = 0;
        string documentName = "Заявление в " +
Constants.VuzShortName + "_" + model.AbiturFam + ".pdf";
        return new
UnicodeFileContentResult(result, "application/pdf",
documentName);
        //return new BinaryContentResult(result,
"application/pdf", documentName);
    }

        public UnicodeFileContentResult
Print_Raspiska(string s,
        string regStep,
        string id,
        string fio,
        string adr,
        string dokType,

```

```

        string dokTypeNot,
        string dokLN,
        string dokSer,
        string dokNom,
        string dokData,
        string dokKem
    )
    {
        byte rs = 0;
        byte.TryParse(regStep, out rs);

        Guid abtId = Guid.Empty;
        Guid.TryParse(id, out abtId);
        ProfileAbiturModel model =
AbiturRepos.Edit(abtId, (RegistrationStep)rs);

        Dogovor_Predstavitel predstv = null;
        if (fio != null) predstv = new
Dogovor_Predstavitel()
        {
            FIO = fio,
            Dokument =
(!string.IsNullOrEmpty(dokType) ? dokType :
dokTypeNot),
            DokumentSeriya = dokSer,
            DokumentNomer = dokNom,
            DokumentVidan_Data = dokData,
            DokumentVidan_Kem = dokKem,
            DokumentIdentification = dokLN,
            Adres = adr
        };

        string path =
HttpContext.Server.MapPath(ASPZiZ_PK.Constants.TemplatesPath)
        ExcelExportData fsed =
PrintRepos.CreatePrint(model, CurrentUser.UserName,
path, predstv);
        return new
UnicodeFileContentResult(fsed.FileStream,
fsed.FileType, fsed.FileName);

        //byte rs = 0;
        //if (byte.TryParse(regStep, out rs) ==
false) { return RedirectToAction("List"); }

        //Guid abtId = Guid.Empty;
        //if (Guid.TryParse(id, out abtId) ==
false) return RedirectToAction("List");
        //ProfileAbiturModel model =
AbiturRepos.Edit(abtId, (RegistrationStep)rs);

```

```

        //byte[] result =
PrintRepos.Raspiska(model);

        //var output = new MemoryStream();
        //output.Write(result, 0, result.Length);
        //output.Position = 0;
        //string documentName = "Расписка в
БГУИР_" + model.AbiturFam + ".pdf";
        //return new BinaryContentResult(result,
"application/pdf", documentName);
    }

    public class BinaryContentResult : ActionResult
    {
        private string ContentType;
        private byte[] ContentBytes;
        private string FileName;

        public BinaryContentResult(byte[]
contentBytes, string contentType, string filename)
        {
            this.ContentBytes = contentBytes;
            this.ContentType = contentType;
            this.FileName = filename;
        }

        public override void
ExecuteResult(ControllerContext context)
        {
            var response =
context.HttpContext.Response;
            response.Clear();

            response.Cache.SetCacheability(HttpCacheability.NoCache);
            response.ContentType =
this.ContentType;

            response.AddHeader("Content-Disposition", "inline;
filename=" + this.FileName);

            var stream = new
MemoryStream(this.ContentBytes);
            stream.WriteTo(response.OutputStream);
            stream.Dispose();
        }
    }

    public class AccountController : Controller
    {
        public ActionResult Index() { return View(); }
    }

```

```

        public ActionResult LogOn() { return View(); }

        [Authorize]
        public ActionResult LogOff()
        {
            FormsAuthentication.SignOut();
            return RedirectToAction("Index", "Home");
        }

        public RedirectToRouteResult
        RedirectToAfterLonOn()
        {
            return RedirectToAction("Menu", "Home");
        }

        [ValidateAntiForgeryToken]
        [HttpPost]
        public ActionResult LogOn(LogOnModel model,
string returnUrl)
        {
            if (ModelState.IsValid)
            {
                if
                (Membership.ValidateUser(model.UserName,
model.UserPassword))
                {
                    ChangeSessionId();

FormsAuthentication.SetAuthCookie(model.UserName,
false);

                    if (Url.IsLocalUrl(returnUrl) &&
returnUrl.Length > 1 && returnUrl.StartsWith("/")
                        && !returnUrl.StartsWith("//") &&
!returnUrl.StartsWith("/\\\")) return
Redirect(returnUrl);
                    else return RedirectToAfterLonOn();
                }
                else
                {
                    ModelState.AddModelError("", "The
user name or password provided is incorrect.");
                }
            }

            return View(model);
        }

        private void ChangeSessionId()
        {
            /* изменение id сессии */
            SessionIDManager manager = new
SessionIDManager();

```

```

        HttpApplication ctx =
HttpContext.ApplicationInstance;

        string oldId =
manager.GetSessionID(ctx.Context);
        string newId =
manager.CreateSessionID(ctx.Context);

        bool isAdd = false;
        bool isRedir = false;

        manager.SaveSessionID(ctx.Context, newId,
out isRedir, out isAdd);

        HttpModuleCollection mods = ctx.Modules;
        SessionStateModule ssm =
(SessionStateModule)mods.Get("Session");
        FieldInfo[] fields =
ssm.GetType().GetFields(BindingFlags.NonPublic |
BindingFlags.Instance);
        SessionStateStoreProviderBase store = null;
        FieldInfo rqIdField = null, rqLockIdField
= null, rqStateNotFoundField = null;
        foreach (FieldInfo field in fields)
        {
            if (field.Name.Equals("_store")) store
= (SessionStateStoreProviderBase)field.GetValue(ssm);
            if (field.Name.Equals("_rqId"))
rqIdField = field;
            if (field.Name.Equals("_rqLockId"))
rqLockIdField = field;
            if
(field.Name.Equals("_rqSessionStateNotFound"))
rqStateNotFoundField = field;
        }
        object lockId =
rqLockIdField.GetValue(ssm);
        if ((lockId != null) && (oldId != null))
store.ReleaseItemExclusive(ctx.Context, oldId, lockId);
        rqStateNotFoundField.SetValue(ssm, true);
        rqIdField.SetValue(ssm, newId);
    }

    [Authorize]
    [HttpPost]
    public ActionResult ContinueSession()
    {
        return Json(new { continueSession = true
});
    }

```

```

        #region Status Codes
        private static string
        ErrorCodeToString(MembershipCreateStatus createStatus)
        {
            // See http://go.microsoft.com/
            fwlink/?LinkID=177550 for
            // a full list of status codes.
            switch (createStatus)
            {
                case
                MembershipCreateStatus.DuplicateUserName:
                    return "User name already exists.
                    Please enter a different user name.";

                case
                MembershipCreateStatus.DuplicateEmail:
                    return "A user name for that
                    e-mail address already exists. Please enter a
                    different e-mail address.";

                case
                MembershipCreateStatus.InvalidPassword:
                    return "The password provided is
                    invalid. Please enter a valid password value.";

                case
                MembershipCreateStatus.InvalidEmail:
                    return "The e-mail address
                    provided is invalid. Please check the value and try
                    again.";

                case
                MembershipCreateStatus.InvalidAnswer:
                    return "The password retrieval
                    answer provided is invalid. Please check the value and
                    try again.";

                case
                MembershipCreateStatus.InvalidQuestion:
                    return "The password retrieval
                    question provided is invalid. Please check the value
                    and try again.";

                case
                MembershipCreateStatus.InvalidUserName:
                    return "The user name provided is
                    invalid. Please check the value and try again.";

                case
                MembershipCreateStatus.ProviderError:
                    return "The authentication

```

provider returned an error. Please verify your entry and try again. If the problem persists, please contact your system administrator.";

```
        case
MembershipCreateStatus.UserRejected:
            return "The user creation request
has been canceled. Please verify your entry and try
again. If the problem persists, please contact your
system administrator.";

        default:
            return "An unknown error occurred.
Please verify your entry and try again. If the problem
persists, please contact your system administrator.";
    }
}
#endregion
}

[Authorize]
public class AdminController : ControllerExtensions
{
    private string _moduleName = "Абитуриенты";
    public ActionResult Index()
    {
        return RedirectToAction("Users");
    }

    public ActionResult Users()
    {
        UserModel userCurr = CurrentUser;
        if (userCurr != null) { if
(!userCurr.Permissions.Adm) return
RedirectToLoginPage(); }
        else RedirectToLoginPage();

        IEnumerable<UserModel> users =
MembershipRepos.GetUsers();
        return View(users);
    }

    public ActionResult Details(string id)
    {
        UserModel userCurr = CurrentUser;
        if (userCurr != null) { if
(!userCurr.Permissions.Adm) return
RedirectToLoginPage(); }
        else RedirectToLoginPage();
    }
}
```

```

        ViewBag.IsNew = false;
        return
View(MembershipRepos.GetUserById(id));
    }

    [HttpPost]
    public ActionResult Save(UserModel model)
    {
        UserModel userCurr = CurrentUser;
        if (userCurr != null) { if
(!userCurr.Permissions.Adm) return
RedirectToLoginPage(); }
        else RedirectToLoginPage();

        if
(!string.IsNullOrEmpty(model.Password)) {
MembershipRepos.ChangePass(model.Login,
model.Password); }
        ProfileUser profile =
ProfileUser.GetProfile(model.Login);
        profile.IsGroup = false;
        profile.FullName = model.UserName;
        profile.RequirePasswordChange = false;
        profile.Save();
        MembershipRepos.SaveUserPermissions(model);
        return RedirectToAction("Users");
    }

    public ActionResult New()
    {
        UserModel userCurr = CurrentUser;
        if (userCurr != null) { if
(!userCurr.Permissions.Adm) return
RedirectToLoginPage(); }
        else RedirectToLoginPage();

        UserModel model = new UserModel();
        model.Permissions = new PermissionsModel();
        ViewBag.IsNew = true;
        return View("Details", model);
    }

    [HttpPost]
    public ActionResult Create(UserModel model)
    {
        UserModel userCurr = CurrentUser;
        if (userCurr != null) { if
(!userCurr.Permissions.Adm) return
RedirectToLoginPage(); }

```



```

        else RedirectToLoginPage();

        MembershipCreateStatus createStatus;
        MembershipUser mu =
Membership.CreateUser(model.Login, model.Password,
model.Login, null, null, true, null, out createStatus);

        if (createStatus ==
MembershipCreateStatus.Success)
        {
            ProfileUser profile =
ProfileUser.GetProfile(model.Login);
            profile.IsGroup = false;
            profile.FullName = model.UserName;
            profile.RequirePasswordChange = false;
            profile.Save();
            model.UserID =
Guid.Parse(mu.ProviderUserKey.ToString());
MembershipRepos.SaveUserPermissions(model);
        }
        else
        {
            ViewBag.IsNew = true;
            return View("Details", model);
        }
        return RedirectToAction("Users");
    }

    public ActionResult Delete(string id)
    {
        UserModel userCurr = CurrentUser;
        if (userCurr != null) { if
(!userCurr.Permissions.Adm) return
RedirectToLoginPage(); }
        else RedirectToLoginPage();

        MembershipRepos.DeleteUser(id);
        return RedirectToAction("Users");
    }
}

public class HomeController : ControllerExtensions
{
    public ActionResult Index()
    {
        return View();
    }

    [Authorize]
    public ActionResult Menu()

```

```

        {
            UserModel user = UserCurrent.Get(User);
            if ((user.Permissions.Site ||
user.Permissions.Och || user.Permissions.Univ ||
user.Permissions.Zabr)
                && !user.Permissions.Adm &&
!user.Permissions.Stat &&
!user.Permissions.Zachislenie)
                return RedirectToAction("List",
"Abitur");

            return View();
        }

        [Authorize]
        public ActionResult Turn()
        {
            UserModel user = UserCurrent.Get(User);
            if (user.Permissions.Site ||
user.Permissions.Och || user.Permissions.Univ ||
user.Permissions.Zabr)
            {
                return RedirectToAction("List",
"Abitur");
            }
            return RedirectToLoginPage();
        }
    }

    public class PrintController : ControllerExtensions
    {
        //
        // GET: /Print/

        private string SessionId_Filter = "tmp";

        public ActionResult Index()
        {
            return RedirectToAction("List");
        }

        public ActionResult List(string regStep,
string[] sortParams)
        {
            byte rs = (byte)RegistrationStep.BSUIRyes;
            AbiturListFilterModel filter = null;
            if (Session != null)
            {
                object tmp = Session[SessionId_Filter];
                if (tmp != null)

```

```

        filter =
(AbiturListFilterModel)tmp;
    }
    if (filter == null)
    {
        filter = new AbiturListFilterModel() {
RegStepId = rs };
    }

    UserModel userCurr = CurrentUser;
    AbiturListModel model = new
AbiturListModel()
    {
        Abiturs =
AbiturRepos.GetListByZach(filter),
        Filter = filter
    };
    FillViewData_ListAction(filter, userCurr);
    if (Session != null)
    {
        Session.Remove(SessionId_Filter);
        Session.Add(SessionId_Filter, filter);
    }
    return View(model);
}

    public void
ExportToLotus(AbiturListFilterModel filter)
    {
        List<AbiturPrintModel> abiturs =
AbiturRepos.GetListByZachForPrint_Spravki(filter);
        string path =
HttpContext.Server.MapPath(ASPZiZ_PK.Constants.TemplatesPath)
        Core.Print.ExportToLotus.ExportData(path);
    }

    public FileContentResult
Spravki(AbiturListFilterModel filter)
    {
        List<AbiturPrintModel> abiturs =
AbiturRepos.GetListByZachForPrint_Spravki(filter);
        string path =
HttpContext.Server.MapPath(ASPZiZ_PK.Constants.TemplatesPath)
        ExcelExportData fsed =
PrintRepos.Spravki(abiturs, path);
        return File(fsed.FileStream,
fsed.FileType, fsed.FileName);
    }

    public FileContentResult
Letters(AbiturListFilterModel filter)
    {

```

```

        List<AbiturPrintModel> abiturs =
AbiturRepos.GetListByZachForPrint_Letters(filter);
        string path =
HttpContext.Server.MapPath(ASPZiZ_PK.Constants.TemplatesPath)
        ExcelExportData fsed =
PrintRepos.Letters(abiturs, path);
        return File(fsed.FileStream,
fsed.FileType, fsed.FileName);
        //return RedirectToAction("List");
    }

    public FileContentResult
Envelops(AbiturListFilterModel filter)
    {
        List<AbiturPrintModel> abiturs =
AbiturRepos.GetListByZachForPrint_Letters(filter);
        string path =
HttpContext.Server.MapPath(ASPZiZ_PK.Constants.TemplatesPath)
        ExcelExportData fsed =
PrintRepos.Envelops(abiturs, path);
        return File(fsed.FileStream,
fsed.FileType, fsed.FileName);
        //return RedirectToAction("List");
    }

    private void
FillViewData_ListAction(AbiturListFilterModel filter,
UserModel currUser)
    {
        ViewData["RegStepList"] = new
SelectList(DDL_Model.GetEnum_RegistrationStep(currUser),
        "Id", "Name", filter.RegStepId.ToString());
        ViewData["SrokObuchList"] = new
SelectList(SrokObuchRepos.GetList(true), "Id", "Name",
        (filter.SrokObId.HasValue ?
filter.SrokObId.Value.ToString() : "0"));
        ViewData["FormaObuchList"] = new
SelectList(FormaObuchRepos.GetList_Filter(filter.IsPlatn,
        filter.SrokObId, filter.isCelev, true), "Id", "Name",
        (filter.FormObId.HasValue ?
filter.FormObId.Value.ToString() : "0"));
        ViewData["FacultyList"] = new
SelectList(FacultyRepos.GetList_Filter(filter.IsPlatn,
        filter.FormObId, filter.SrokObId, filter.isCelev),
        "Id", "Name", (filter.FacultId.HasValue ?
filter.FacultId.Value.ToString() : "0"));
        ViewData["SpecialityList"] = new
SelectList(SpecRepos.GetList_Filter(filter.IsPlatn,
        filter.FormObId, filter.SrokObId, filter.isCelev,
        filter.FacultId), "Id", "Name", (filter.SpecId.HasValue
        ? filter.SpecId.Value.ToString() : "0"));
    }

```

```

        [HttpPost]
        public ActionResult
PostActionToUpdateAbiturs(AbiturListFilterModel
filter, string[] sortParams)
        {
            UserModel userCurr = CurrentUser;
            if(filter == null)
                filter = new AbiturListFilterModel() {
RegStepId = (byte)RegistrationStep.BSUIRyes };
            AbiturListModel model = new
AbiturListModel()
            {
                Abiturs =
AbiturRepos.GetListByZach(filter),
                Filter = filter
            };
            if (Session != null)
            {
                Session.Remove(SessionId_Filter);
                Session.Add(SessionId_Filter, filter);
            }
            return PartialView("partial_AbiturList",
model);
        }

```

```

        public ActionResult
PostActionToSpravki(AbiturListFilterModel filter,
string[] sortParams)
        {
            UserModel userCurr = CurrentUser;
            if (filter == null)
                filter = new AbiturListFilterModel() {
RegStepId = (byte)RegistrationStep.BSUIRyes };
            AbiturListModel model = new
AbiturListModel()
            {
                Abiturs =
AbiturRepos.GetListByZach(filter),
                Filter = filter
            };
            if (Session != null)
            {
                Session.Remove(SessionId_Filter);
                Session.Add(SessionId_Filter, filter);
            }
            return PartialView("partial_AbiturList",
model);
        }

```

```

    }

    public class ProgressController : Controller
    {
        protected readonly ProgressManager
ProgressManager;
        public ProgressController()
        {
            ProgressManager = new ProgressManager();
        }

        public String GetTaskId()
        {
            var id =
Request.Headers["X-SimpleProgress-TaskId"];
            return id ?? String.Empty;
        }

        public String Progress()
        {
            var taskId = GetTaskId();
            return ProgressManager.GetStatus(taskId);
        }
    }

    public interface IProgressManager
    {
        void SetCompleted(String taskId, Int32
percentage);
        void SetCompleted(String taskId, String step);
        String GetStatus(String taskId);
    }

    public class ProgressManager : IProgressManager
    {
        public void SetCompleted(string taskId, int
percentage)
        {
            throw new NotImplementedException();
        }

        public void SetCompleted(string taskId, string
step)
        {
            AspNetProgressProvider prov = new
AspNetProgressProvider();
            prov.Set(taskId, step);
        }

        public string GetStatus(string taskId)

```

```

        {
            AspNetProgressProvider prov = new
AspNetProgressProvider();
            return prov.Get(taskId);
        }
        public static int HeaderNameTaskId { get; set; }
    }
}

public interface IProgressDataProvider
{
    void Set(String taskId, String progress, Int32
durationInSeconds = 300);
    String Get(String taskId);
}

public class AspNetProgressProvider :
IProgressDataProvider
{
    public void Set(String taskId, String progress,
Int32 durationInSeconds = 3)
    {
        HttpContext.Current.Cache.Insert(
            taskId,
            progress,
            null,
            DateTime.Now.AddSeconds(durationInSeconds),
            Cache.NoSlidingExpiration);
    }

    public String Get(String taskId)
    {
        var o = HttpContext.Current.Cache[taskId];
        if (o == null)
            return String.Empty;

        return (String)o;
    }
}

public class StatController : Controller
{
    //
    // GET: /Stat/

    public ActionResult Index()
    {
        AbiturListFilterModel model = new
AbiturListFilterModel();
        string s = "2014-07-08 00:00";
    }
}

```

```

        model.DataPodachiS =
DateTime.ParseExact(s, "yyyy-MM-dd HH:mm",
CultureInfo.InvariantCulture);
        model.DataPodachiPo = DateTime.Now;
        return View(model);
    }

    public ActionResult ExportStat()
    {
        string path =
HttpContext.Server.MapPath(ASPZiZ_PK.Constants.TemplatesPath)

        string s = "2014-07-08 00:00";
        DateTime dateS = DateTime.ParseExact(s,
"yyyy-MM-dd HH:mm", CultureInfo.InvariantCulture);
        DateTime datePo = DateTime.Now;

        ExcelExportData fsed =
Statistics.CreateStatistics(path, dateS, datePo);
        return File(fsed.FileStream,
fsed.FileType, fsed.FileName);
    }

    public ActionResult
ExportStatDate(AbiturListFilterModel filter)
    {
        string path =
HttpContext.Server.MapPath(ASPZiZ_PK.Constants.TemplatesPath)
        ExcelExportData fsed =
Statistics.CreateStatistics(path, filter.DataPodachiS,
filter.DataPodachiPo );
        return File(fsed.FileStream,
fsed.FileType, fsed.FileName);
    }
}

    public class ZachislenieController :
ProgressController
    {
        public ActionResult Index()
        {
            return RedirectToAction("GSList");
        }

        public ActionResult List()
        {
            return View(SpecGroupRepos.GetList());
        }

        public ActionResult Delete(string id)
        {
            SpecGroupRepos.Delete(int.Parse(id));
        }
    }

```



```

        return RedirectToAction("GSList");
    }

    public ActionResult Details(string id)
    {
        return
View(SpecGroupRepos.Details(int.Parse(id)));
    }

    public ActionResult Preview(string id)
    {
        //var taskId = GetTaskId();
        return
View(ZachRepos.Zachislit(int.Parse(id), true));
    }

    public ActionResult PreviewReport(string id)
    {
        ExcelExportData fsed =
ZachRepos.GetReport(int.Parse(id));
        return File(fsed.FileStream,
fsed.FileType, fsed.FileName);
    }

    public ActionResult Submit(string id)
    {
        return View("Preview",
ZachRepos.Zachislit(int.Parse(id), false));
    }
}

```