МИНИСТЕРСТВО ОБРАЗОВАНИЯ ОРЕНБУРГСКОЙ ОБЛАСТИ **ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ «ОРЕНБУРГСКИЙ КОЛЛЕДЖ ЭКОНОМИКИ И ИНФОРМАТИКИ»**

**(ГАПОУ ОКЭИ)**

**ОТЧЁТ**

ОКЭИ 09.02.07. 7021 7 У

УП.03.01 Учебная практика

По модулю ПМ.03 Ревьюирование программных модулей

ГАПОУ «ОКЭИ»

(Место прохождения практики)

Количество листов: 41

Дата готовности: 05.03.2021

Разработал: студент группы 4ис1 Корнев Д.С.

Руководитель: Асташева Т.С.

Соответствие отчета с заданием на практику\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Защищён\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_с оценкой\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оренбург 2021

**Содержание**

Содержание 2  
Введение 3  
1 Техническое задание 4  
2 Информационное обеспечение 10  
3 Логика программ и программных модулей 13  
4 Ревьюирование программного кода 15

5 Тестирование и отладка 20

6 Метрики программного кода 23

7 Установка, настройка и работа с системой контроля версий 26  
Заключение 30  
Список использованных источников 31  
Приложение А (обязательное) Входные документы 33  
Приложение Б (обязательное) Выходные документы 35  
Приложение В (обязательное) Информационная модель 36  
Приложение Г (обязательное) Схема взаимосвязи программ и программных модулей 37

Приложение Д (обязательное) Журнал опытной эксплуатации 38  
Приложение Е (обязательное) Тест – кейсы 39  
Приложение Ж (обязательное) Тестирование 40  
Приложение И (обязательное) Метрики кода 41

**Введение**

Осознавая важность подачи документов в академию, была разработана **информационная система "Абитуриент".** Система позволяет отслеживать анкеты абитуриентов, их средний балл и место на поступление. Эти задачи можно решить при помощи современных компьютерных систем управления бизнесом.

Для того чтобы реализовать разработку системы «**Абитуриент»** необходимо решить задачи:

- выбрать стратегии автоматизации;

- изучить системный подход к автоматизации работ по созданию алгоритмов и программ;

- подготовить к автоматизации профессиональных задач с использованием персонального компьютера;

- изучить методологию построения и стандарты написания программ;

- выбрать язык программирования с учетом требований заказчика в зависимости от решаемой задачи;

- определить основные этапы работ по созданию алгоритмов и написанию программ.

Данная система будет осуществляется через программный продукт «1С: Предприятие». Внедрение системы 1С в организации обеспечит повышение эффективности поиска по данных. Система обладает отличным функционалом. Повышается не только скорость выполнения работы специалистам организации, но и качество результатов.

Автоматизация 1С – является ее невероятная гибкость и как следствие возможность индивидуальной настройки под требования конкретной организации. В целом, использование данного программного продукта позволяет существенно снизить нагрузку на отдел организации, путем автоматизации типовых задач и исключения медленного бумажного оборота, облегчить контроль над ее деятельностью при помощи разнообразных форм отчетности и снизить вероятность возникновения ошибки по вине конечного пользователя к минимуму.

**1 Техническое задание**

1.1 Общие сведения о проекте

Полное наименование – Автоматизированная информационная система «Абитуриент» в «РАНХиГС».

* Краткое наименование – 1С «Абитуриент».
* Заказчиком данной системы является: Российская Академия Народного Хозяйства и Государственной Службы «РАНХиГС».
* Разработчиком данной системы является обучающийся «Оренбургского колледжа экономики и информатики», отделения информационных систем, группы 4ис1 Корнев Данил Сергеевич Чкалова 11.
* Разработка ведется на основании методических рекомендаций для написания дипломного проекта.
* Данная подсистема будет создаваться с 28 февраля 2021 года по 06 марта 2021.
* Работы по созданию автоматизированной информационной системы сдаются разработчиком поэтапно в соответствии с планом проекта.

1.2 Назначение и цели создания системы

Данная подсистема ведения базы данных «Абитуриент» является автоматизация деятельности предприятия по набору абитуриентов в академию, а также их учету в базе данных.

Объектами автоматизации будут служить:

- приемная комиссия;

- академия;

Целью создания подсистемы является автоматизация процесса ведения базы данных о абитуриентах «РАНХиГС».

1.3 Требования к системе

1.3.1 Требования к системе в целом

Подсистема «Абитуриент» должна хранить все данные в центральном хранилище, чтобы система выполняла как можно больше операций, при этом быстродействие должно быть на высоком уровне, а также она должна быть понятной и приятной, с точки зрения пользователя, для удобной работы с ней.

Система должна обладать мерами защиты данных и возможностью резервного копирования.

Входные данные:

- приказ о заселение абитуриента в общежитие;

- приказ о зачисление.

Входные данные представлены в приложении А.

Подсистема должна обладать следующими качествами:

- простота в обслуживании;

- эффективность;

- простота в пользовании;

- эргономичность;

- модифицируемость;

- структурированность;

- сохранность данных;

- безопасность данных.

1.3.2 Требования к функциям, выполняемым системой

- авторизация пользователя;

- ввод данных;

- ведение справочников;

- формирование отчетов;

- поиск данных.

Разрабатываемая подсистема позволит формировать выходные документы:

- отчет об абитуриенте.

1.3.3 Требования к видам обеспечения

Данная подсистема не требует дополнительно технического обеспечения. Для функционирования системы достаточно обычного рабочего стола и компьютера. Параметры этого компьютера должны соответствовать современным нормам.

- процессор Intel Pentium Celeron 1800 Мгц и выше;

- оперативная память 256 Мб и выше;

- жесткий диск 40Гб и выше;

- SVGA-видеокарта;

Монитор. Жесткий диск объёмом от 500 ГБ, стандартные устройства ввода и вывода: мышь, монитор, клавиатура, принтер, устройства обеспечения доступа в локальную сеть.

Программное обеспечение.

Для функционирования подсистемы на компьютере пользователя должны быть установлены:

–  операционная система Windows 10 со всеми необходимыми драйверами устройств;

– установленная система 1С Предприятие;

– установленное программное обеспечение для корректной работы принтера;

– установленный MS Excel.

Организационное обеспечение.

С системой работа осуществляется по мере возникновения потребности специалиста.

1.4 Состав и содержание работ по созданию системы

Создание подсистемы разработка системы 1С «Абитуриент» «РАНХиГС» должна пройти перечень стадий и этапов разработки по созданию АС в соответствии с принятым ГОСТ 34.601–90.4

Таблица 1 – состав и содержание работ по созданию системы

|  |  |  |  |
| --- | --- | --- | --- |
| Стадия | Этап | Начало выполнения | Конец выполнения |
| Формирование требований | Исследование проектной области | 28.02.2021 | 28.02.2021 |
| Формирование технического задания | Разработка ТЗ | 28.02.2021 | 28.02.2021 |
| Разработка автоматизированной системы | Планирование системы | 01.03.2021 | 01.03.2021 |
| Реализация системы | 01.03.2021 | 01.03.2021 |
| Разработка системы и кода | 02.03.2021 | 02.03.2021 |
| Тестирование системы | 03.03.2021 | 04.03.2021 |
| Ввод в эксплуатацию | Приемка системы | 04.03.2021 | 06.03.2021 |

1.5 Порядок контроля и приемки системы

Данная стадия по создания автоматизированной системы должна быть выполнена в сроки 28.02.21 года по 06.03.21 года.

В зависимости от взаимосвязей испытываемых в системе объектов испытания могут быть автономные или комплексные.

Автономные испытания охватывают части подсистемы. Их проводят по мере готовности частей приложения к сдаче в опытную эксплуатацию. В программе автономных испытаний указывают:

– перечень функций, подлежащих испытаниям;

– описание взаимосвязей объекта испытаний с другими частями системы;

– условия, порядок и методы проведения испытаний и обработки результатов;

– критерии приемки частей по результатам испытаний.

Комплексные испытания проводят для групп, взаимосвязанных частей подсистемы или для системы в целом.

В программе комплексных испытаний системы или частей подсистемы

указывают:

– перечень объектов испытания;

– состав предъявляемой документации;

– описание проверяемых взаимосвязей между объектами испытаний;

– очередность испытаний частей системы;

– порядок и методы испытаний, в том числе состав программных средств и оборудования, необходимых для проведения испытаний, включая специальные стенды и полигоны.

При комплексных испытаниях допускается использовать в качестве исходной информацию, полученную на автономных испытаниях частей системы.

Протокол комплексных испытаний должен содержать заключение о возможности (невозможности) приемки системы в опытную эксплуатацию, а также перечень необходимых доработок и рекомендуемые сроки и выполнения. После устранения недостатков проводят повторные комплексные испытания в необходимом объеме.

В зависимости от вида требований, предъявляемых к подсистеме на испытаниях, проверке или аттестации в ней подвергают:

– комплекс программных и технических средств.

– персонал;

– эксплуатационную документацию, регламентирующую деятельность персонала при функционировании приложения;

– системы в целом.

При испытаниях системы проверяют:

– качество выполнения комплексом программных и технических средств автоматических функций во всех режимах функционирования системы согласно ТЗ на создание подсистемы;

– знание персоналом эксплуатационной документации, наличие у него навыков, необходимых для выполнения установленных функций во всех режимах функционирования системы, согласно ТЗ на создание подсистемы;

– полноту содержащихся в эксплуатационной документации указаний персоналу по выполнению им функций во всех режимах функционирования системы согласно ТЗ на создание подсистемы;

– количественные и (или) качественные характеристики выполнения автоматических и автоматизированных функций системы в соответствии с ТЗ;

– другие свойства системы, которым она должна соответствовать по ТЗ.

1.6 Требования к составу и содержанию работ по подготовке объекта автоматизации к вводу системы в действие

Проверку работы персонала в диалоговом режиме проводят с учетом полноты и качества выполнения функций системы в целом.

Проверке подлежит:

– полнота сообщений, директив, запросов, доступных оператору и их достаточность для эксплуатации подсистемы;

– сложность процедур диалога, возможность работы персонала без специальной подготовки;

– реакция системы и ее частей на ошибки оператора, средства сервиса.

Проверка средств восстановления работоспособности системы после отказов ПК должна включать:

– проверку наличия в эксплуатационной документации рекомендаций по восстановлению работоспособности и полноту их описания;

– практическую выполнимость рекомендованных процедур;

– работоспособность средств автоматического восстановления функций (при их наличии).

Проверку комплектности и качества эксплуатационной документации следует проводить путем анализа документации на соответствие требованиям нормативно-технических документов ТЗ.

Результаты испытаний объектов, предусмотренных программой, фиксируют в протоколах, содержащих следующие разделы:

– назначение испытаний и номер раздела требований ТЗ на подсистему, по которому проводят испытание;

– состав технических и программных средств, используемых при испытаниях;

– указание методик, в соответствии с которыми проводились испытания, обработка и оценка результатов;

– условия проведения испытаний и характеристики исходных данных;

– средства хранения и условия доступа к конечной, тестирующей программе;

– обобщенные результаты испытаний;

– выводы о результатах испытаний и соответствии созданной системы или ее частей определенному разделу требований ТЗ на разработку подсистемы.

При внедрении подсистемы в «РАНХиГС», специалист должен научиться работать с ней.

При необходимости производится обучение специалиста для работы с данной системой. Необходимо оборудовать место для работы пользователя с системой, отрегулировать соответствующие показатели.

При создании подсистемы используется технико-экономическое обоснование, где описывается база данных «РАНХиГС», для которого разрабатывается подсистема. Перечисляются выполняемые функции, круг решаемых задач. Техническое задание, где указано полное наименование системы, её условное обозначение для «РАНХиГС». ТЭО – первый документ, создаваемый на пред проектные стадии разработки подсистемы, подтверждающий её экономическую и производственную необходимость. ТЭО должно содержать: обоснование цели проектирования и состава комплекса подсистем и задач, перечень организационно–технических мероприятий по разработке и внедрению сайта, оценка экономической эффективности. ТЗ – документ, завершающий пред проектную стадию создания подсистемы и должен включать в себя требования к задачам, техническому комплексу, обеспечивающим подсистемам и их составу. В ТЗ должна быть определена очередность проектирования и внедрение подсистемы.

1.7 Требования к документированию

Для работы приложения необходимо создать руководство пользователя и руководство программиста. Руководство пользователя нужно для ознакомления пользователя с приложением и его инсталляцией на рабочую станцию. Руководство программиста содержит принцип работы программы.

**2 Информационное обеспечение**

Данные, используемые системой, должны храниться в виде базы данных, состоящей из нескольких связанных между собой таблиц.

Данные берутся из документов:

- договор на платные услуги;

- договоров.

На основании этих документов производится ведение внутренних справочников:

- анкета абитуриента;

- специальность;

- платные услуги;

- организация.

Для решения задачи использованы:

- справочники;

- документы;

- перечисления;

- отчеты;

Справочник «анкета абитуриента» содержит полную информацию о абитуриенте. Структура справочника представлена в таблице 1.

Таблица 1 – Структура справочника «анкета абитуриента»

|  |  |  |
| --- | --- | --- |
| Имя реквизита | Тип данных | Длина |
| ФИО | Строка | 75 |
| ПолученноеОбразование | ПеречислениеСсылка.Образование | - |
| ВидОбучения | ПеречислениеСсылка.ВидОбучения | - |
| ФормаОбучения | ПеречислениеСсылка.ФормаОбучения | - |
| Специальность | СправочникСсылка.Специальность | - |
| Пол | ПеречислениеСсылка.Пол | - |
| ДатаРождения | Дата | - |
| МестоРождения | Строка | 75 |
| КонтактныйНомер | Строка | 18 |
| Почта | Строка | 30 |
| СреднийБалл | Число | 5 |
| Серия | Число | 4 |
| Номер | Число | 6 |
| ДатаВыдачи | Дата | - |
| ОрганВыдавшийДокумент | Строка | 110 |
| ДанныеКартинки | ХранилищеЗначения | - |
| Организация | СправочникСсылка.Организация | - |
| Цена | СправочникСсылка.Специальность | - |

Справочник «специальность» содержит полную информацию о специальности. Структура справочника представлена в таблице 2.

Таблица 2 – Структура справочника «специальность»

|  |  |  |
| --- | --- | --- |
| Имя реквизита | Тип данных | Длина |
| Специальность | Строка | 75 |
| СрокОбучения | ПеречислениеСсылка.СрокОбучения | - |
| ФормаОбучения | ПеречислениеСсылка.ФормаОбучения | - |
| ВидОбучения | ПеречислениеСсылка.ВидОбучения | - |
| ЦенаОбучения | Число | 10 |
| ДополнительныеКурсы | СправочникСсылка.ПлатныеУслуги | - |
| КоличествоМест | Число | 10 |

Справочник «организация» содержит полную информацию о организации. Структура справочника представлена в таблице 3.

Таблица 3 – Структура справочника «организация»

|  |  |  |
| --- | --- | --- |
| Имя реквизита | Тип данных | Длина |
| Организация | Строка | 75 |
| Адрес | Строка | 50 |
| Почта | Строка | 30 |
| КонтактныйНомер | Число | 11 |

Справочник «платные услуги» содержит полную информацию о платных услугах. Структура справочника представлена в таблице 4.

Таблица 4 – Структура справочника «платные услуги»

|  |  |  |
| --- | --- | --- |
| Имя реквизита | Тип данных | Длина |
| ДополнительныеКурсы | Строка | 40 |
| КоличествоЗанятий | Число | 10 |
| Цена | Число | 10 |

Документ «договор на обучение» содержит полную информацию о заключенном договоре на обучение. Структура справочника представлена в таблице 5.

Таблица 5 – Структура документа «договор на обучение»

|  |  |  |
| --- | --- | --- |
| Имя реквизита | Тип данных | Длина |
| АнкетаАбитуриента | СправочникСсылка. АнкетаАбитуриента | - |
| Организация | СправочникСсылка.Организация | - |
| ДатаПодачиДокументов | Дата | - |

Документ «договор на платные услуги» содержит информацию на заключение договора на дополнительные курсы. Структура справочника представлена в таблице 6.

Таблица 6 – Структура документа «договор на платные услуги»

|  |  |  |
| --- | --- | --- |
| Имя реквизита | Тип данных | Длина |
| АнкетаАбитуриента | СправочникСсылка. АнкетаАбитуриента | - |
| Организация | СправочникСсылка.Организация | - |
| ДополнительныеКурсы | СправочникСсылка.ПлатныеУслуги | - |

**3 Логика программ и программных модулей**

В системе «1С: Предприятие 8.3» существуют несколько видов программных модулей. Они различаются по месту размещения и доступному контексту.

Общие модули располагаются в отдельной ветке дерева метаданных. Основным назначением общих модулей является содержание общих алгоритмов конфигурации, доступных из разных модулей.

Набор прикладных объектов имеет собственные модули. К таким объектам относятся: справочники, документы, отчеты.

Модули располагаются в ветках конфигурации, в которых содержатся сами объекты и являются свойствами объектов. Каждый объект имеет свой индивидуальный модуль. В этих модулях возможно объявление переменных, процедур и функций, которые будут доступны при работе с объектом извне во встроенном языке, дополняя контекст объекта.

Модули форм содержатся в формах конфигурации. Каждая форма имеет свой индивидуальный модуль. В этих модулях возможно объявление переменных, процедур и функций, которые будут доступны при работе с формой извне во встроенном языке, дополняя контекст формы.

Автоматизированная информационная система «Абитуриент», имеет в своем наличии модули прикладных объектов и модули форм.

Меню: программы будет состоять из следующих разделов:

* справочники;
* документы;
* отчеты.

При выборе конкретного раздела меню, будет происходить вызов нужной формы для обработки данных.

Форма справочника «АнкетаАбитуриента» содержит следующие объекты:

* ФИО, предназначен для хранения ФИО абитуриента;
* полученное образование абитуриента;
* пол, для уточнения пола абитуриента;
* дата рождения, для информации о возрасте абитуриента;
* телефон, для хранения телефонного номера абитуриента;
* специальность, для информации о абитуриента.

Форма справочника «Специальность» содержит следующие объекты:

* специальность, предназначен для хранения наименования специальности;
* цена обучения, для хранения стоимости обучения специальности.

Форма справочника «Организация» содержит следующие объекты:

* организация, предназначен для хранения наименования организации;
* адрес, предназначен для хранения адреса организации;
* контактный номер, для хранения номера организации.

Форма справочника «Платные услуги» содержит следующие объекты:

* дополнительные курсы, предназначен для хранения наименования курсов;
* количество занятий, предназначен для информации о количестве занятий;
* цена, предназначен для хранения информации цене дополнительного курса.

Форма договора «Договор на обучение» содержит следующие объекты:

* анкета абитуриента, предназначен для хранения договора заключенного с абитуриентом;
* организация, предназначен для хранения наименования организации.

Форма договора «Договор на платные услуги» содержит следующие объекты:

* анкета абитуриента, предназначен для хранения договора заключенного с абитуриентом;
* организация, предназначен для хранения наименования организации.
* платные услуги, предназначен для хранения информации о платных услугах.

Модуль справочника «Анкета абитуриента» содержит следующую

процедуру – добавление картинки.

**4 Ревьюирование программного кода**

Инспекциякода(Codereview) – систематический и периодический анализ программного кода, направленный на поиск необнаруженных на ранних стадиях разработки программного продукта ошибок, а также, на выявление некачественных архитектурных решений и критических мест в программе.

Задачи и цели проведения формальных инспекций

Не всегда возможна разработка автоматических или хотя бы четко формализованных ручных тестов для проверки функциональности программной системы. В некоторых случаях выполнение тестируемого программного кода невозможно в условиях, создаваемых тестовым окружением. Такая ситуация возможна во встроенных системах, если программный код предназначен для обработки исключительных ситуаций, создаваемых только на реальном оборудование.

В тех случаях, когда верифицируется не программный код, а проектная документация на систему, которую нельзя "выполнить" или создать для нее отдельные тестовые примеры, также обычно прибегают к методу экспертных исследований программного кода или документации на корректность или непротиворечивость.

Такие экспертные исследования называют инспекциями или просмотрами. Существует два типа инспекций - неформальные и формальные.

При неформальной инспекции автор некоторого документа или части программной системы передает его эксперту, а тот, ознакомившись с документом, передает автору список замечаний, которые тот исправляет. Сам факт проведения инспекции и замечания нигде отдельно не сохраняются, состояние исправлений по замечаниям также нигде не отслеживается.

Формальная инспекция является четко управляемым процессом, структура которого обычно четко определяется соответствующим стандартом проекта. Таким образом, все формальные инспекции имеют одинаковую структуру и одинаковые выходные документы, которые затем используются при разработке.

Факт начала формальной инспекции четко фиксируется в общей базе данных проекта. Также фиксируются документы, подвергаемые инспекции, и списки замечаний, отслеживаются внесенные по замечаниям изменения. Этим формальная инспекция похожа на автоматизированное тестирование: списки замечаний имеют много общего с отчетами о выполнении тестовых примеров.

В ходе формальной инспекции группой специалистов осуществляется независимая проверка соответствия инспектируемых документов исходным документам. Независимость проверки обеспечивается тем, что она осуществляется инспекторами, не участвовавшими в разработке инспектируемого документа.

Входами процесса формальной инспекции являются инспектируемые документы и исходные документы, а выходами - материалы инспекции, включающие список обнаруженных несоответствий и решение об изменении статуса инспектируемых документов.

Этапы формальной инспекции и роли ее участников

Процесс формальной инспекции состоит из пяти фаз: инициализация, планирование, подготовка (экспертиза), обсуждение, завершение. В некоторых случаях подготовку и обсуждение целесообразно рассматривать не как последовательные этапы, а как параллельные подпроцессы. В частности, такая ситуация может сложиться при использовании автоматизированной системы поддержки проведения формальных инспекций.

Процедура формальной инспекции проекта должна точно описывать порядок проведения формальных инспекций в данном проекте.

После устранения обнаруженных в ходе формальной инспекции несоответствий процесс формальной инспекции повторяется, возможно, в другой форме и с другим составом участников. Процедура формальной инспекции должна регламентировать возможные формы проведения повторной инспекции в зависимости от объема и характера изменений, внесенных в объект инспекции. Как правило, допускается упрощение процесса повторной инспекции (проведение инспекции одним инспектором, отсутствие фазы обсуждения) при внесении в объект инспекции незначительных изменений относительно ранее инспектировавшейся версии.

Инициализация

Руководитель проекта или его заместитель запрашивает из базы, хранящей все данные проекта (например, из системы конфигурационного управления), список объектов, готовых к инспекции, выбирает объект инспекции, затем назначает участников формальной инспекции: автора, ведущего и одного или нескольких инспекторов.

Ведущий также может выполнять роль инспектора; остальные участники выполняют только одну роль. На роль ведущего или инспектора не допускается назначать сотрудников, участвовавших в разработке объекта инспекции.

В роли автора выступает один из разработчиков объекта инспекции, но возможны ситуации, когда разработчик недоступен - например, переведен в другой проект или находится в отпуске. Тогда на роль автора назначается сотрудник, который будет исправлять обнаруженные несоответствия в инспектируемых документах. При инспектировании документов, разработанных заказчиком, автор может не назначаться.

Рекомендуется назначать не менее двух инспекторов. Их количество может быть увеличено, если инспектируются документы особой сложности или новизны понятий, а также, если в качестве инспекторов привлекаются сотрудники с недостаточным опытом. Рекомендуемое общее число участников инспекции не должно превышать пяти.

В обоснованных случаях процедура формальной инспекции проекта может допускать проведение инспекции единственным инспектором, например, когда объект инспекции отличается особой простотой и оцениваемые характеристики такого объекта инспекции тривиальны.

В случае, если проводится повторная инспекция по сокращенной форме, ведущий самостоятельно инициирует процесс повторной инспекции без участия руководителя проекта. Процедура формальной инспекции проекта может разрешать ведущему самостоятельно инициировать процесс повторной инспекции (в том же составе участников), даже когда она проводится в полной форме, если это диктуется спецификой проекта.

Планирование

По завершению процесса инициализации ведущий проверяет, что инспектируемые документы размещены в базе данных проекта, а их статус соответствует готовности к формальной инспекции. Если это не так, инспекция откладывается.

Затем ведущий должен изменить статус инспектируемых документов так, чтобы отметить факт начала инспекции и ограничить доступ к инспектируемой документации. Во время инспекции изменение документов невозможно, а соответствующий статус сохраняется до конца инспекции. Этот статус называется Review.

После этого ведущий должен скопировать из базы данных проекта бланк инспекции и занести в него идентификаторы инспектируемых и исходных документов и номера их версий, список участников с указанием их ролей и дату фактического начала процесса инспекции, т.е. того момента, когда инспектируемые документы были переведены в состояние Review

Ведущий должен оценить время, необходимое инспекторам для подготовки, и продолжительность обсуждения. Время, отводимое на этап подготовки, не может быть менее одного часа. Также ведущий должен определить дату, время и место обсуждения, если оно будет проходить в форме собрания. При этом может потребоваться согласование с другими участниками инспекции. Если оценка продолжительности обсуждения в форме собрания превышает 2 часа, то необходимо запланировать несколько собраний, каждое из которых будет длиться не более двух часов.

Процедура формальной инспекции проекта может допускать проведение повторной инспекции без собрания, если итогом предыдущей инспекции было решение о проведении повторной инспекции в сокращенной форме. Также допускается не проводить собрание, если результаты формальной инспекции ведутся и хранятся в электронном виде.

В этом случае процедура формальной инспекции проекта должна регламентировать взаимодействия участников формальной инспекции между собой. Кроме того, процедура формальной инспекции проекта должна определять механизм подготовки, проведения обсуждения и принятия решения.

Система 1С «Абитуриент» сначала прошла ревьюирование программного кода разработчиком. Код разработан и разделен по тематике задач. В среде разработки Arduino IDE не подразумевается автоматическое ревьюирование программного кода. После выявления ошибок код прошел модернизацию, которая привела код к стандартизированному виду.

Объект инспекции: программный код системы 1С «Абитуриент».

Разработчик – Корнев Д.С., студент группы 4ис1.

Инспектор – Асташева Т.С., руководитель учебной практики по ПМ.03

Место проведения ГАПОУ «ОКЭИ». Все несоответствия были обозначены и исправлены.

Результат ревьюирования – код программы читабельный и исправен.

На рисунке 1 представлен код справочника «АнкетаАбитуриента». Используется для добавления картинки в справочнике.

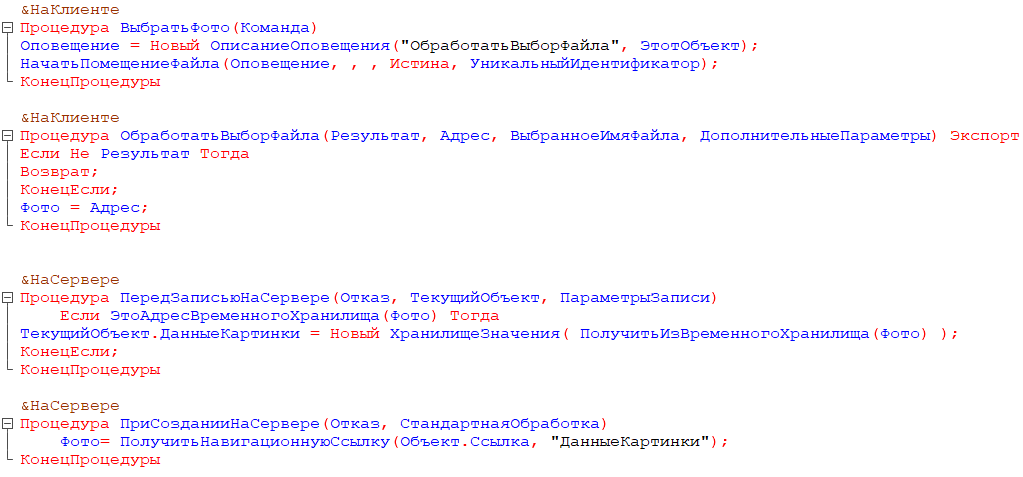


Рисунок 1 – код модуля справочника «Сотрудники»

На рисунке 2 представлен код отчета «Отчет абитуриента». Позволяет выводить необходимые данные из справочника «АнкетаАбитуриента».

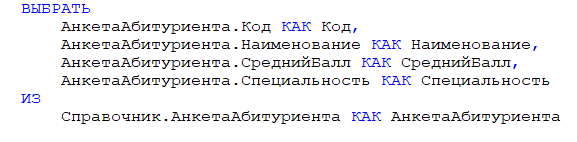


Рисунок 2 – код отчета «Отчет абитуриента»

На рисунке 3 представлен код заполнения документа «ДоговорОбОбучении».

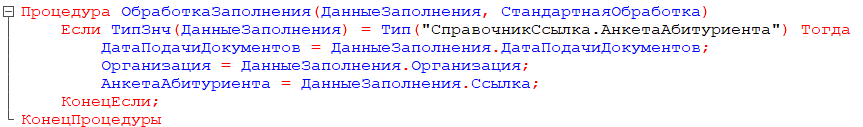


Рисунок 3 – код заполнения «ДоговорОбОбучении»

На рисунке 4 представлен код заполнения документа «ДоговорНаПлатныеУслуги».

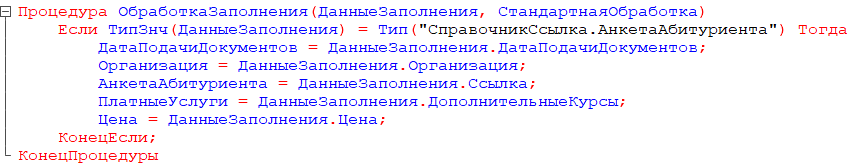


Рисунок 4 – код заполнения «ДоговорНаПлатныеУслуги»

**5 Тестирование и отладка**

Тестирование программного обеспечения – проверка соответствия между реальным и ожидаемым поведением программы, осуществляемая на конечном наборе тестов, выбранном определенным образом. В более широком смысле, тестирование – это одна из техник контроля качества, включающая в себя активности по планированию работ, проектированию тестов, выполнению тестирования и анализу полученных результатов

План Тестирования – это документ, описывающий весь, объем работ по тестированию, начиная с описания объекта, стратегии, расписания, критериев начала и окончания тестирования, до необходимого в процессе работы оборудования, специальных знаний, а также оценки рисков с вариантами их разрешения.

Тест дизайн – это этап процесса тестирования ПО, на котором проектируются и создаются тестовые случаи (тест кейсы), в соответствии с определёнными ранее критериями качества и целями тестирования.

Тестовый случай - это артефакт, описывающий совокупность шагов, конкретных условий и параметров, необходимых для проверки реализации тестируемой функции или её части.

Баг/Дефект Репорт – это документ, описывающий ситуацию или последовательность действий, приведшую к некорректной работе объекта тестирования, с указанием причин и ожидаемого результата.

Тестовое Покрытие – это одна из метрик оценки качества тестирования, представляющая из себя плотность покрытия тестами требований либо исполняемого кода.

Детализация Тест Кейсов – это уровень детализации описания тестовых шагов и требуемого результата, при котором обеспечивается разумное соотношение времени прохождения к тестовому покрытию

Время прохождения Тест Кейса – это время от начала прохождения шагов тест кейса до получения результата теста.

* тестирование компонентов — тестируется минимально возможный для тестирования компонент, например, отдельный класс или функция. часто тестирование компонентов осуществляется разработчиками программного обеспечения.
* интеграционное тестирование — тестируются интерфейсы между компонентами, подсистемами или системами. при наличии резерва времени на данной стадии тестирование ведётся итерационно, с постепенным подключением последующих подсистем.
* системное тестирование — тестируется интегрированная система на её соответствие требованиям.
* альфа-тестирование — имитация реальной работы с системой штатными разработчиками, либо реальная работа с системой потенциальными пользователями/заказчиком. Чаще всего альфа-тестирование проводится на ранней стадии разработки продукта, но в некоторых случаях может применяться для законченного продукта в качестве внутреннего приёмочного тестирования. Иногда альфа-тестирование выполняется под отладчиком или с использованием окружения, которое помогает быстро выявлять найденные ошибки. Обнаруженные ошибки могут быть переданы тестировщикам для дополнительного исследования в окружении, подобном тому, в котором будет использоваться программа.
* бета-тестирование — в некоторых случаях выполняется распространение предварительной версии (в случае проприетарного программного обеспечения иногда с ограничениями по функциональности или времени работы) для некоторой большей группы лиц с тем, чтобы убедиться, что продукт содержит достаточно мало ошибок. Иногда бета-тестирование выполняется для того, чтобы получить обратную связь о продукте от его будущих пользователей.

Часто для свободного и открытого программного обеспечения стадия альфа-тестирования характеризует функциональное наполнение кода, а бета-тестирования — стадию исправления ошибок. При этом как правило на каждом этапе разработки промежуточные результаты работы доступны конечным пользователям.

При статическом тестировании программный код не выполняется — анализ программы происходит на основе исходного кода, который вычитывается вручную, либо анализируется специальными инструментами. В некоторых случаях анализируется не исходный, а промежуточный код.

После внесения изменений в очередную версию программы, регрессионные тесты подтверждают, что сделанные изменения не повлияли на работоспособность остальной функциональности приложения. Регрессионное тестирование может выполняться как вручную, так и средствами автоматизации тестирования.

При тестировании белого ящика, разработчик теста имеет доступ к исходному коду программ и может писать код, который связан с библиотеками тестируемого программного обеспечения. Это типично для компонентного тестирования, при котором тестируются только отдельные части системы. Оно обеспечивает то, что компоненты конструкции работоспособны и устойчивы, до определённой степени. При тестировании белого ящика используются метрики покрытия кода или мутационное тестирование.

При тестировании чёрного ящика тестировщик имеет доступ к программе только через те же интерфейсы, что и заказчик или пользователь, либо через внешние интерфейсы, позволяющие другому компьютеру либо другому процессу подключиться к системе для тестирования.

Например, тестирующий компонент может виртуально нажимать клавиши или кнопки мыши в тестируемой программе с помощью механизма взаимодействия процессов, с уверенностью в том, все ли идёт правильно, что эти события вызывают тот же отклик, что и реальные нажатия клавиш и кнопок мыши. Как правило, тестирование чёрного ящика ведётся с использованием спецификаций или иных документов, описывающих требования к системе. Обычно в данном виде тестирования критерий покрытия складывается из покрытия структуры входных данных, покрытия требований и покрытия модели.

При тестировании серого ящика разработчик теста имеет доступ к исходному коду, но при непосредственном выполнении тестов доступ к коду, как правило, не требуется.

Если «альфа-» и «бета-тестирование» относятся к стадиям до выпуска продукта (а также, неявно, к объёму тестирующего сообщества и ограничениям на методы тестирования), тестирование «белого ящика» и «чёрного ящика» имеет отношение к способам, которыми тестировщик достигает цели.

Бета-тестирование в целом ограничено техникой чёрного ящика (хотя постоянная часть тестировщиков обычно продолжает тестирование белого ящика параллельно бета-тестированию). Таким образом, термин «бета-тестирование» может указывать на состояние программы (ближе к выпуску, чем «альфа»), или может указывать на некоторую группу тестировщиков и процесс, выполняемый этой группой. То есть, тестировщик может продолжать работу по тестированию белого ящика, хотя программа уже «бета-стадии», но в этом случае он не является частью «бета-тестирования».

К разрабатываемому программному продукту применялись разные методы тестирования. После окончания написания кода было проведено функциональное тестирование, которое показало соответствие программного приложения требованиям. Далее, после выявления и исправления различных ошибок, было проведено регрессионное тестирование, которое выявило, как влияют исправленные дефекты на существующую функциональность приложения и не возникают ли старые дефекты. После проведения выше перечисленных методов тестирования, было проведено модульное тестирование для того, чтобы удостовериться в корректной работе всей системы, для этого каждый модуля, входящий в систему был протестирован по отдельности. Тестирование локализации и юзабилити тестирование были проведены в последний момент для выявления степени удобности пользования программным продуктом, а также для выявления ошибок в локализации программы.

Тест кейсы для проведения тестирования были описаны в Приложении Е. Ход тестирования был описан в Приложении Ж. При тестировании выявленные ошибки были занесены в журнал опытной эксплуатации, который описан в Приложении Д.

# 6 Метрики программного кода

В общем случае применение метрик позволяет руководителям проектов и предприятий изучить сложность разработанного или даже разрабатываемого проекта, оценить объем работ, стилистику разрабатываемой программы и усилия, потраченные каждым разработчиком для реализации того или иного решения. Однако метрики могут служить лишь рекомендательными характеристиками, ими нельзя полностью руководствоваться, так как при разработке ПО программистам, стремясь минимизировать или максимизировать ту или иную меру для своей программы, могут прибегать к хитростям вплоть до снижения эффективности работы программы. Кроме того, если, к примеру, программист написал малое количество строк кода или внес небольшое число структурных изменений, это вовсе не значит, что он ничего не делал, а может означать, что дефект программы было очень сложно отыскать. Последняя проблема, однако, частично может быть решена при использовании метрик сложности, т.к. в более сложной программе ошибку найти сложнее.

Метрики программного обеспечения можно разделить на три категории:

- Метрики продукта — описывает характеристики продукта, такие как размер, сложность, особенности дизайна, производительность и уровень качества.

- Метрики процесса — эти характеристики могут использоваться для улучшения деятельности по разработке и сопровождению программного обеспечения.

- Метрики проекта — эти метрики описывают характеристики и исполнение проекта. Примеры включают число разработчиков программного обеспечения, штатное расписание в течение жизненного цикла программного обеспечения, стоимость, график и производительность.

Метрики качества программного обеспечения представляют собой подмножество метрик программного обеспечения, которые фокусируются на аспектах качества продукта, процесса и проекта. Они более тесно связаны с метриками процесса и продукта, чем с метриками проекта.

Метрики качества программного обеспечения можно разделить на три категории:

- метрики качества продукции;

- показатели качества в процессе;

- метрики качества обслуживания.

Метрики качества продукции

Эти показатели включают в себя следующее:

- среднее время до отказа;

- плотность дефектов;

- проблемы с клиентами;

- удовлетворенность клиентов.

У нас в ДО встроена серьёзная подсистема замеров показателей производительности и вычислений различных метрик. Это нужно для того, чтобы и в текущий момент времени, и в исторической перспективе понимать, что в системе происходит, что становится хуже, что становится лучше. Средства мониторинга – метрики и замеры времени – входят в типовую поставку «1С: Документооборот 8». Метрики требуют настройки на внедрении, но сам механизм типовой.  
Метрики – это замеры различных бизнес-показателей в те или иные моменты времени (например, среднее время доставки почты в моменте 10 минут).  
Одна из метрик показывает количество активных пользователей в базе. В среднем их 1000-1400 в течение дня. На графике видно, что на момент скриншота в базе было 2144 активных пользователя.

Ещё одна прикладная метрика для нас – среднее время ожидания загрузки писем с почтового сервера (на скриншоте показана красным). Грубо говоря, сколько будет письмо гулять по Интернету, прежде чем оно окажется у нашего сотрудника. На скриншоте видно, что это время тоже никак не изменилось за последнее время. Есть отдельные всплески – но они связаны не с задержками, а с тем, что время сбивается на почтовых серверах.

Метрики – это, в основном, инструмент администратора для мониторинга системы, для быстрого реагирования на какие-то изменения в поведении системы. На скриншоте – метрики внутреннего ДО за год. Скачок на графиках обусловлен тем, что перед нами поставили задачи по развитию внутреннего ДО.

Метрики качества в использовании (метрики эксплуатационного качества) измеряют степень, в которой программный продукт, установленный и эксплуатируемый в определенной среде, удовлетворяет потребности пользователей в эффективном, продуктивном и безопасном решении задач.

Метрики качества в использовании помогают оценить не свойства самой ПС, а видимые результаты ее эксплуатации - эксплуатационное качество.

Очевидно, что для правильного измерения эксплуатационного качества важно учитывать контекст применения ПС – особенности категорий ее пользователей, специфику решаемых ими задач, а также физические и социальные факторы среды их работы.

Метрики качества в использовании (метрики эксплуатационного качества) измеряют степень, в которой программный продукт, установленный и эксплуатируемый в определенной среде, удовлетворяет потребности пользователей в эффективном, продуктивном и безопасном решении задач.

Метрики качества в использовании помогают оценить не свойства самой ПС, а видимые результаты ее эксплуатации - эксплуатационное качество.

Очевидно, что для правильного измерения эксплуатационного качества важно учитывать контекст применения ПС – особенности категорий ее пользователей, специфику решаемых ими задач, а также физические и социальные факторы среды их работы. Метрика производительности представлена на рисунке 5.

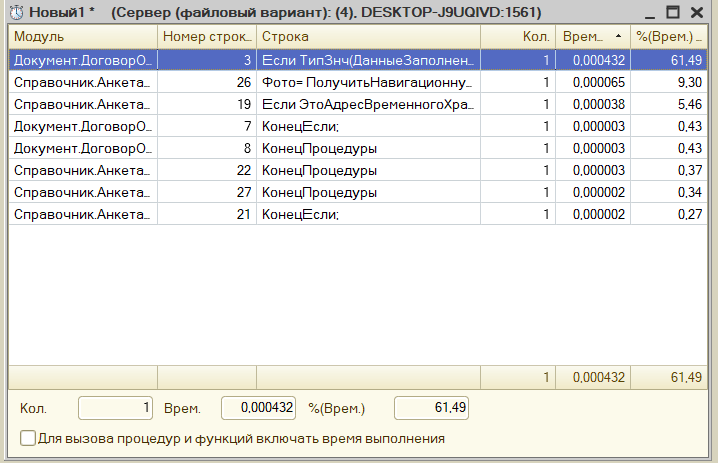


Рисунок 5 – Метрика производительности

# 7 Установка, настройка и работы с системой контроля версий

Git - одна из самых популярных систем контроля версий, используется разработчиками, для контроля изменений в своих разработках и проектах. Изначально создан для использования на Linux-подобных операционных системах, но позднее, из-за удобства и популярности для Windows был написан специальный эмулятор, поддерживающий функционал Git’a.

Настройка Git:

Откройте терминал и запустите команды. Действие представлено на рисунке 6.

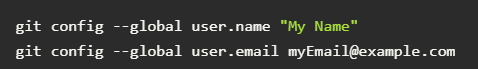


Рисунок 6 - Команда настройки Gita

Теперь каждое наше действие будет отмечено именем и почтой. Таким образом, пользователи всегда будут в курсе, кто отвечает за какие изменения — это вносит порядок.

Чтобы создать новый репозиторий, нам нужно открыть терминал, зайти в папку нашего проекта и выполнить команду init. Это включит приложение в этой конкретной папке и создаст скрытую директорию .git, где будет храниться история репозитория и настройки.

Создайте на рабочем столе папку под названием git\_exercise. Действие представлено на рисунке 7.

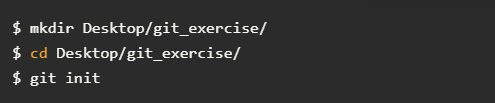


Рисунок 7 - Команда создания папки на рабочем столе

Командная строка должна вернуть. Действие представлено на рисунке 8.



Рисунок 8 - ответ программы на запрос

Определение состояния

status — это еще одна важнейшая команда, которая показывает информацию о текущем состоянии репозитория: актуальна ли информация на нём, нет ли чего-то нового, что поменялось, и так далее. Действие представлено на рисунке 9.

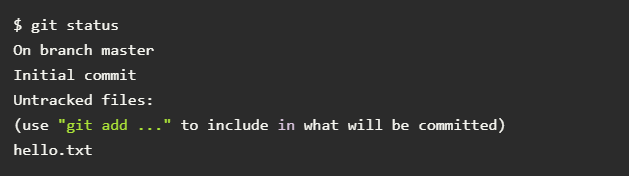


Рисунок 9 - Команда определения статуса.

Подготовка файлов

В git есть концепция области подготовленных файлов. Можно представить ее как холст, на который наносят изменения, которые нужны в коммите. Сперва он пустой, но затем мы добавляем на него файлы. Действие представлено на рисунке 10.



Рисунок 10 - Команда добавления файла.

Если нам нужно добавить все, что находится в директории, мы можем использовать. Действие представлено на рисунке 11.



Рисунок 11 - Команда добавление всей информации из директорий

Удаленные репозитории:

Подключение к удаленному репозиторию. Действие представлено на рисунке 12.

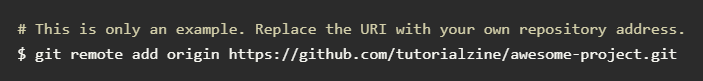


Рисунок 12 - Команда подключения к репозиторию

Отправка изменений на сервер. Действие представлено на рисунке 13.

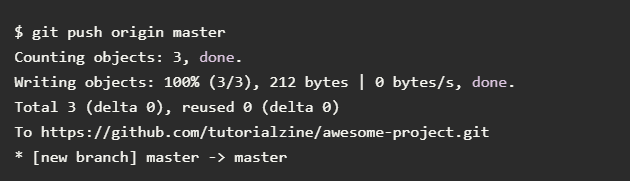


Рисунок 13 - Команда отправка изменений на сервер

Клонирование репозитория. Действие представлено на рисунке 14.



Рисунок 14 - Команда клонирования репозитория

Запрос изменений с сервера. Действие представлено на рисунке 15.

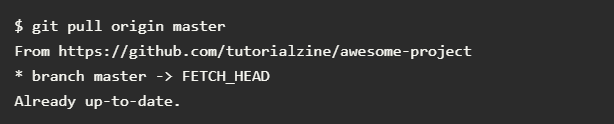


Рисунок 15 - Команда запроса изменений с сервера.

Github — это очень известная платформа для хранения, распространения и управления исходным кодом открытых проектов. Github использует множество разработчиков по всему миру, среди которых есть и крупные компании, такие как Microsoft, RedHat и другие. Github предоставляет возможности не только по просмотру кода и его распространения, но также историю версий, инструменты совместной разработки, средства для предоставления документации, выпуска релизов и обратной связи. И самое интересное, что вы можете размещать на Gihub как открытые, так и приватные проекты. В этой статье мы рассмотрим, как пользоваться Github для размещения своего проекта. Так сказать, github для начинающих.

Для создания репозитория необходимо:

* на открывшейся странице, это главная страница для авторизованных пользователей, нажмите кнопку «Start a project»;
* ввести название и описание репозитория;
* можно сразу же инициализировать репозиторий, создав файл Readme, для этого нужно отметить галочку «Initialize this repository with a README» внизу страницы. Также можно выбрать лицензию;
* когда все будет готово, выберите «Create project», будет создан новый проект с файлом README, в котором находится описание и файлом лицензии.

Ветки Github позволяют работать с несколькими версиями проекта одновременно. По умолчанию при создании репозитория создается ветка master, это основная рабочая ветка. Можно создать дополнительные ветки, например, для того, чтобы тестировать программное обеспечение перед тем, как оно будет опубликовано в ветке master. Таким образом, можно одновременно разрабатывать продукт и предоставлять пользователям стабильную версию. Также можно создавать отдельные ветки для версии программы для разных систем.

Любые изменения файлов на Github делаются с помощью коммитов. Коммит выполняется путем внесения самих исправлений и описания этих исправлений. Это необходимо для того, чтобы знали, что и когда вы меняли, а также позволяет легко отслеживать работу команды. Слово коммит можно перевести как «фиксировать». То есть мы можем внести изменения в несколько файлов, а затем их зафиксировать.

# Заключение

При прохождении учебной практики особых трудностей не возникло.

Автоматизированная информационная система – одна из важнейших частей организации. Она удобна в использовании. В АИС описывается практически вся деятельность компании, что в свою очередь способствует повышению качества работы персонала и администрации, а также наглядному представлению информации.

При прохождении учебной практики, были поставлены следующие задачи:

- изучить предметную область, входную и выходную информацию;

- познакомиться с функциями, выполняемыми данной задачей;

- провести предпроектные исследования: создать информационную

выстроить логику работы системы;

- разработать техническое задание;

- разработать информационную базу;

- разработать интерфейс и функции программного обеспечения на

выбранном языке программирования;

- провести ревьюирование программного кода;

- написать тест-кейсы для проведения тестирования программного

обеспечения;

- создать пользовательские тесты методами встроенного языка;

- создать метрики программного кода;

- установить, настроить и выполнить виды работ в системе контроля

версий;

- обработать исключения;

- провести тестирование и отладку;

- оформить отчёт.

Поставленные требования были выполнены успешно без особых затруднений.

**Список использованных источников**

1. ГОСТ 2.104-2006 ЕСКД. Основные надписи;
2. ГОСТ 2.105-95 ЕСКД. Общие требования к текстовым документам;
3. ГОСТ 2.106-96 ЕСКД. Текстовые документы;
4. ГОСТ 2.109-73 ЕСКД. Основные требования к чертежам;
5. ГОСТ 2.304-81 ЕСКД. Шрифты чертежные;
6. «1С:Предприятие 8.3. Практическое пособие разработчика».: М.Г. Радченко, Е.Ю. Хрусталева;
7. Книга «1С:Программирование для начинающих. Детям и родителям, менеджерам и руководителям. Разработка в системе "1С:Предприятие 8.3».: М. Радченко;
8. «Книга «1C:Предприятие 8.3. Практическое пособие разработчика. Примеры и типовые приемы».: Глоссарий. М.Г. Радченко, Е.Ю. Хрусталева;
9. Книга «101 совет начинающим разработчикам в системе «1С:Предприятие 8».: Е. Ю. Хрусталева;
10. «Книга «Разработка сложных отчетов в «1С:Предприятии 8.2». Система компоновки данных». Издание 2».: Е. Ю. Хрусталева;
11. «Медицина».: статья, 2015 года – Режим доступа: https://ru.wikipedia.org/wiki – 01.09.20 г.;
12. «Лекция №4. Этапы моделирования базы данных». : статья Миледи Осипова, 2014 года Режим доступа: https://pandia.ru/text/78/539/95475.php - 01.09.20 г.;
13. «Практическое руководство по созданию UML-диаграмм». : статья, 2015 года – Режим доступа: https://flexberry.github.io/ru/gpg\_practical-guides-uml.html;
14. «Медицинская регистратура».: стать – Режим доступа: https://www.cvmt-fili.ru/deputy/registry;
15. «Информационное обеспечение процессов управления в системе контроллинга».: статья Нечеухина Н.С., ГОУ ВПО Уральский государственный экономический университет (УРГЭУ) – Режим доступа:https://1c.ru/rus/partners/training/edu/theses/?y=2016&s=55&t=1428;
16. «Установка 1С».: статья – Режим доступа: https://wiseadvice-it.ru/o-kompanii/blog/articles/ustanovka-1s/
17. «Тестирование программного обеспечения».: статья – Режим доступа: https://ru.wikipedia.org/wiki;
18. «Виды тестирования ПО».: статья – Режим доступа: http://wiki.rosalab.ru/ru/index.php;
19. «Константы 1С».: статья – Режим доступа: https://programmist1s.ru/konstantyi-v-1s/;
20. «Информационная модель».: статья – Режим доступа: https://ru.wikipedia.org/wiki;
21. «Функциональная модель организации и влияние ее элементов на конкурентоспособность».: статья – Режим доступа: https://www.cfin.ru/management/strategy/competit/functional\_model.shtml;
22. «Диаграмма прецедентов».: статья – Режим доступа: https://ru.wikipedia.org/wiki;
23. «Диаграмма деятельности».: статья – Режим доступа: http://khpi-iip.mipk.kharkiv.edu/library/case/leon/;
24. «Создание печатных форм в 1С 8.2 и 8.3 с помощью конструктора».: статья – Режим доступа: https://programmist1s.ru/sozdanie-i-nastroyka-pechatnyih-form-v-1s/;
25. «Пример разработки отчета в системе компоновки данных».: статья – Режим доступа: https://its.1c.ru/db/metod8dev/content/1452/hdoc.

**Приложение А**

*(обязательное)*

**Входные данные**

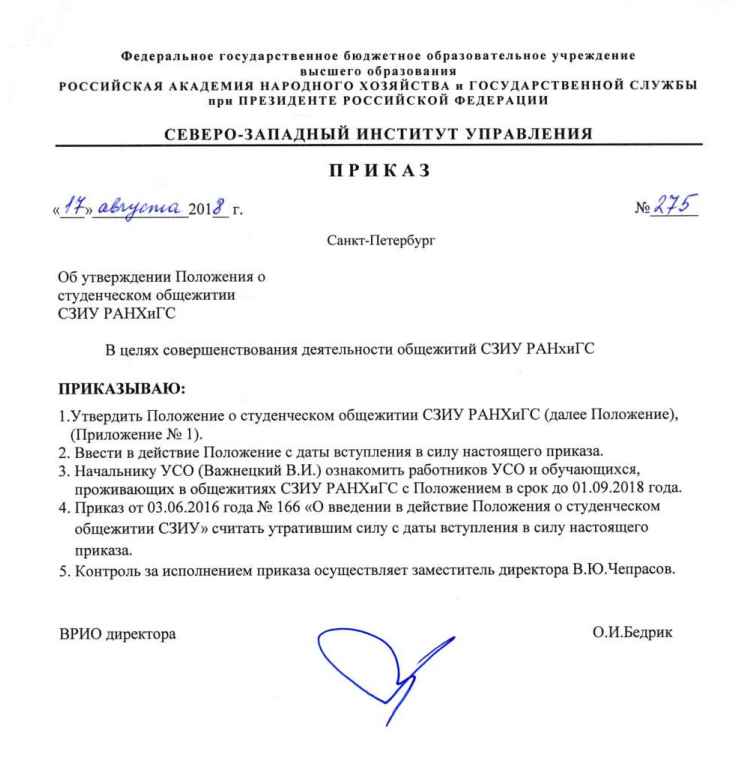


Рисунок А.1 - Приказ о заселении в общежитие

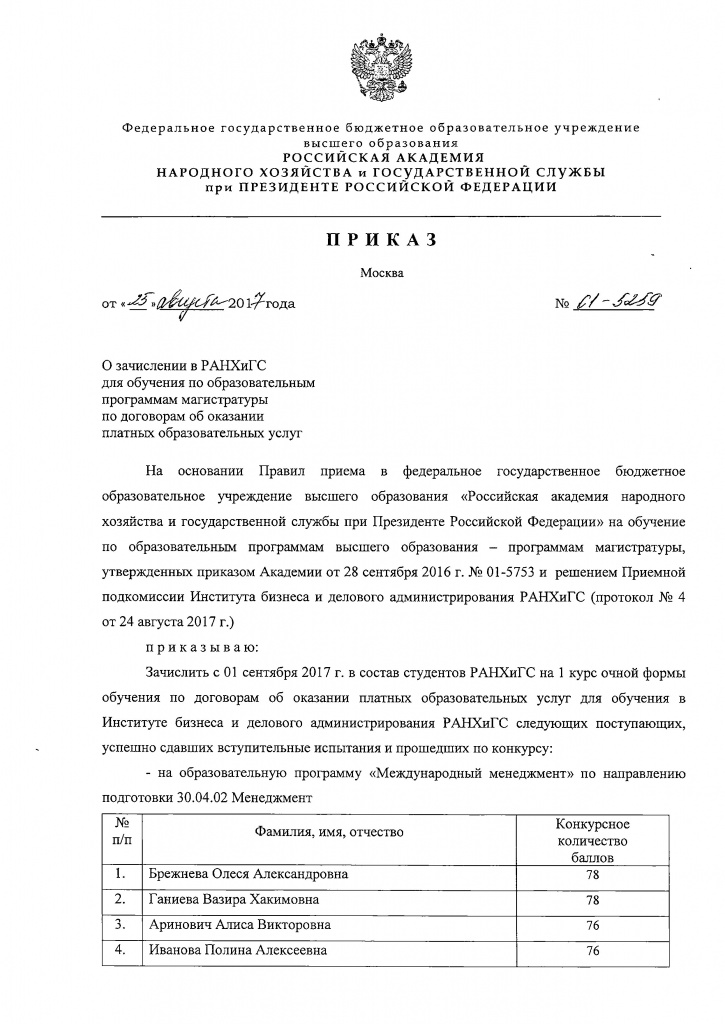


Рисунок А.2 - Приказ на зачисление

**Приложение Б**

*(обязательное)*

**Выходные данные**

Выходная информация на экран:

- отчет по абитуриенту.

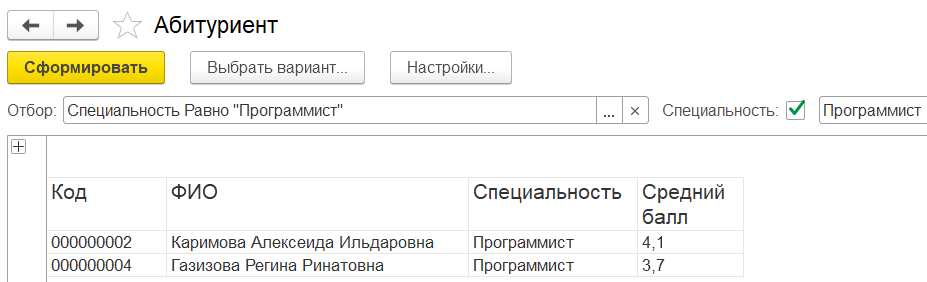


Рисунок Б.1 – Отчет по абитуриенту

**Приложение В**

*(обязательное)*

**Информационная модель**

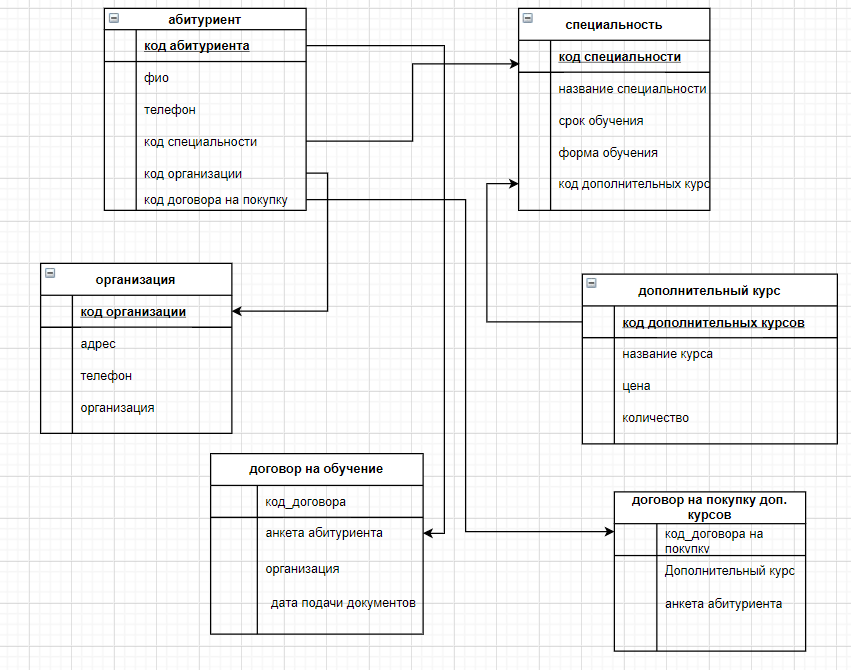


Рисунок В.1 – Информационная модель

**Приложение Г**

*(обязательное)*

**Схема взаимодействия программ и программных модулей**

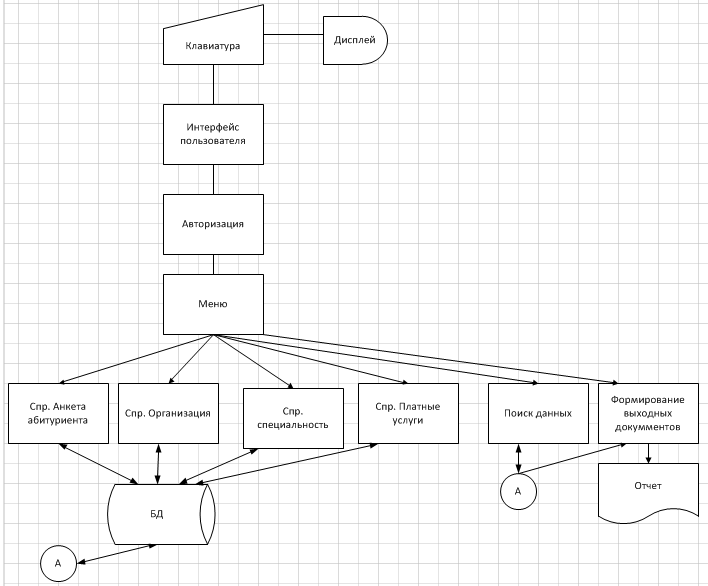


Рисунок Г.1 – Схема взаимодействия программ и программных модулей

**Приложение Д**

*(обязательное)*

**Журнал опытной эксплуатации**

Таблица Д.1 – Журнал опытной эксплуатациии

|  |  |  |  |
| --- | --- | --- | --- |
| Дата | Используемые функции | Возникающие проблемы | Решение |
| 03.03.2021 | Запуск базы | База не открывается | - обновление платформы;  - очистка временных (cash) файлов: удалить базу из списка и подключить её же заново;  - вход в базу от другой учётной записи. |
| 04.03.2021 | Сохранение документа или записи | Недостаточно памяти | Увеличить размер выделенной памяти. Для этого запускается адресная строка (Пуск – Выполнить, вводиться фраза cmd). После нажатия «Enter» достаточно ввести фразу bcdedit /set increaseuserva 4096 и подтвердить действие (клавиша «Enter»). Цифра 4096 – новый выделяемый объем «оперативки». Выполняется перезагрузка системы. Проблема должна быть устранена. |
| 04.03.2021 | Отсутствуют права доступа | Ошибка доступа | Для исправления ситуации достаточно изменить роль выбранного профиля в конфигураторе (вкладка «Администрирование – Пользователи»). |

**Приложение Е**

*(обязательное)*

**Тест кейсы**

Таблица Е.1 – Тест кейсы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер | Название | Предварительные шаги | Шаги | Ожидаемый результат |
| 1 | Тестирование авторизации | Выбрать необходимого пользователя и ввести пароль | Выбор пользователя | Пользователь зайдет под своей учеткой |
| 2 | Тестирование создания нового абитуриента | Выбор пункта «Анкета абитуриента», выбор вкалдки «Абитуриент». Создание нового элемента в справочнике | Создание нового элемента в справчонике | Система создаст новый элемент в справочнике |
| 3 | Тестирование заполнения данными нового справочника | В созданном справочнике заполняем все нужные поля | Заполение полей | Пользователь запонит все необходмые поля справчника |
| 4 | Тестирование создания документа | Выбор нужного пункта, вкладки. Создание нового нового элемента в документе | Создание нового элемента в документе | Система создаст новый элемент в документе |
| 5 | Тестирование заполнения данными нового документа | В созданном документе заполняем все нужные поля | Заполнение поля | Пользователь запонит все необходмые поля документа |
| 6 | Тестирование вывода отчета на экран | Выбор нужного отчета | Формирование отчета | Система сформирует отчет |

**Приложение Ж**

*(обязательное)*

**Тестирование**

Тестирование вывода отчета «Абитуриент».

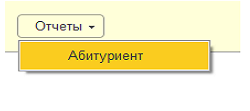


Рисунок Ж.1 – выбор отчета "Абитуриент"

Экранная форма отчета «абитуриент».

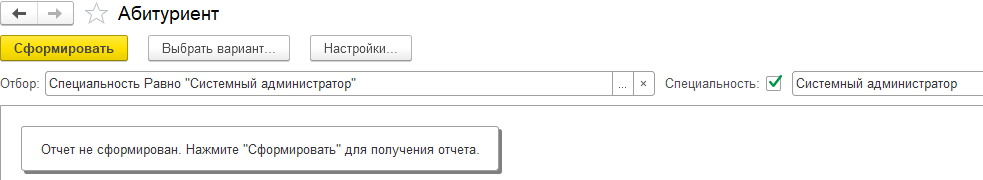


Рисунок Ж.2 - Экранная форма отчета "Абитуриент"

Формирование отчета «Абитуриент».

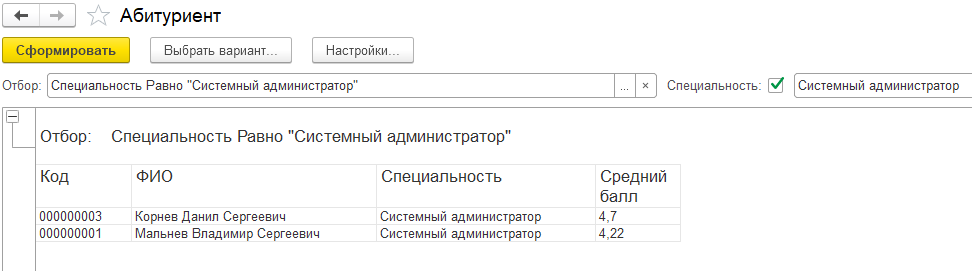


Рисунок Ж.3 – Формирование отчета

**Приложение И**

*(обязательное)*

**Метрики кода**

&НаКлиенте

Процедура ВыбратьФото(Команда)

Оповещение = Новый ОписаниеОповещения("ОбработатьВыборФайла", ЭтотОбъект);

НачатьПомещениеФайла (Оповещение, , , Истина,

УникальныйИдентификатор);

КонецПроцедуры

&НаКлиенте

Процедура ОбработатьВыборФайла(Результат, Адрес, ВыбранноеИмяФайла, ДополнительныеПараметры) Экспорт

Если Не Результат Тогда

Возврат;

КонецЕсли;

Фото = Адрес;

КонецПроцедуры

&НаСервере

Процедура ПередЗаписьюНаСервере(Отказ, ТекущийОбъект, ПараметрыЗаписи)

Если ЭтоАдресВременногоХранилища(Фото) Тогда

ТекущийОбъект.ДанныеКартинки = Новый ХранилищеЗначения

( ПолучитьИзВременногоХранилища(Фото) );

КонецЕсли;

КонецПроцедуры

&НаСервере

Процедура ПриСозданииНаСервере(Отказ, СтандартнаяОбработка)

Фото=ПолучитьНавигационнуюСсылку(Объект.Ссылка, "ДанныеКартинки");

КонецПроцедуры

Процедура ОбработкаЗаполнения(ДанныеЗаполнения, СтандартнаяОбработка)Если ТипЗнч(ДанныеЗаполнения) =Тип("СправочникСсылка.АнкетаАбитуриента") Тогда

ДатаПодачиДокументов = ДанныеЗаполнения.ДатаПодачиДокументов;

Организация = ДанныеЗаполнения.Организация;

АнкетаАбитуриента = ДанныеЗаполнения.Ссылка;

КонецЕсли;

КонецПроцедуры