**АННОТАЦИЯ**

Отчет о курсовой работе: 50 с., 18 рис., 2 приложения, 7 источников.

Объект исследования – характеристика.

Предмет исследования – система генерации характеристик

Цель работы – разработка приложения для генерации характеристик, которая позволяла бы генерировать характеристики по заданному шаблону.

Метод исследования – анализ возможностей языка С# для создания и обработки входных и выходных данных .

В работе были использованы возможности программных продуктов Visual Studio.

В результате решения задачи было разработано приложение для автоматической генерации характеристик. Приложение позволяет задавать необходимую информацию о компании, о человеке, на которого пишется характеристика: ФИО, дата рождения, дата принятия на работу и должность, награды, особые качества и нарекания.

Приложение автоматическая генирация характеристик может использоваться работниками отдела кадров, либо стоящим выше руководством для быстрого оформления характеристик работника по готовым шаблонам.

С#, ПРОГРАММНЫЙ ПРОДУКТ, ХАРАКТЕРИСТИКА, VISUAL STUDIO, WPF

**СОДЕРЖАНИЕ**

[1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 7](#_Toc36986932)

[1.1 Состояние вопроса 7](#_Toc36986933)

[1.2 Актуальность и цель работы 12](#_Toc36986934)

[2 ТЕХНИЧЕСКОЕ ЗАДАНИЕ 14](#_Toc36986935)

[2.1 Описание области применения и исходных данных приложения 14](#_Toc36986936)

[2.2 Требования к пользовательским интерфейсам 14](#_Toc36986937)

[2.3 Требования к аппаратным, программным и коммуникационным интерфейсам 18](#_Toc36986938)

[2.4 Требования к пользователям продукта 19](#_Toc36986939)

[2.5 Требования к адаптации на месте 19](#_Toc36986940)

[2.6 Функции продукта 19](#_Toc36986941)

[2.7 Ограничения 20](#_Toc36986942)

[3 ОБОСНОВАНИЕ ВЫБОРА ИНСТРУМЕНТАЛЬНЫХ СРЕДСТВ 21](#_Toc36986943)

[3.1 Интегрированная среда разработки Visual Studio 21](#_Toc36986944)

[3.2 Язык программирования С# 23](#_Toc36986945)

[3.3 Система для построения клиентских приложений Windows WPF 25](#_Toc36986946)

[3.4 Архитектурный паттерн MVVM 26](#_Toc36986947)

[4 РАЗРАБОТКА ПРИЛОЖЕНИЯ АВТОМАТИЧЕСКАЯ ГЕНЕРАЦИЯ ХАРАКТЕРИСТИК 28](#_Toc36986948)

[4.1 Входные и выходные данные приложения 28](#_Toc36986949)

[4.2 Проектирование структуры приложения 28](#_Toc36986950)

[4.3 Описание объектов и их взаимодействия для ООП 28](#_Toc36986951)

[5 ТЕСТИРОВАНИЕ ПРОГРАММНОГО ПРОДУКТА 29](#_Toc36986952)

[5.1 Аппаратные и программные средства создания и эксплуатации интернет-приложения 29](#_Toc36986953)

[5.2 Описание контрольных примеров 30](#_Toc36986954)

[ЗАКЛЮЧЕНИЕ 31](#_Toc36986955)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 32](#_Toc36986956)

[ПРИЛОЖЕНИЕ А Экранные формы 33](#_Toc36986957)

[ПРИЛОЖЕНИЕ Б Фрагменты листинга 34](#_Toc36986958)

**ВВЕДЕНИЕ**

Ведение бизнеса - это процесс по ходу, которого предприниматель сталкивается с миллионом разнообразных нюансов и вопросов прямо не связанных с его основной деятельностью. Особенно если речь идет о малом бизнесе, где все приходится делать самому. Примеров такого совместительства много, одним из них является составление характеристики на сотрудника.

Характеристика - это краткий отзыв (в нашем случаи работодателя) о конкретном человеке (сотруднике) с уточнением его профессиональных, деловых, личных качеств, а также описание его трудового стажа на конкретном рабочем месте.

Необходимо отметить, что в характеристике с места работы не указывается общая биография или достижения и этапы трудовой деятельности вне данного предприятия. То есть пишем только о работе в конкретной фирме, другие этапы жизни указываются в резюме или отдельных характеристиках. Аналогичная ситуация с указанием семейного состояния или наличие образования.

В большинстве случаев характеристику работнику выдают на фирменном бланке предприятия, если такого нет, то обязательно указание полных реквизитов фирмы или предпринимателя в первом разделе. Подписывают такую бумагу или непосредственно руководитель, или же уполномоченной лицо с проставлением мокрой печати [1].

В данной курсовой работе будет реализовано приложение, которое позволит автоматически генерировать характеристики.

# 1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Состояние вопроса

Характеристика с места работы считается деловым документом, который предоставляется сотруднику последним работодателем. Главная цель этого документа — описать деловые качества сотрудника и оценить его как специалиста. Во многих компаниях характеристика с места работы является весомым фактором во время принятия решения о приеме на работу.

Характеристики можно разделить на два основных типа:

1. Внутренняя (характеристика с предыдущей должности /отдела)

Когда сотрудника переводят внутри компании в другой отдел или на другую должность, то ему предоставляется внутренняя характеристика. Также она может быть написана в случае поощрений или дисциплинарных нарушений.

2. Внешняя (характеристика с места работы)

Под внешней характеристикой имеется в виду документ, который составляется по запросу сотрудника или нового работодателя. Кроме этого, характеристику еще могут запросить такие учреждения, как суд, военкомат, представители банка и другие.

Именно поэтому, к написанию характеристики стоит относиться очень внимательно и учитывать все необходимые пункты [6].

Характеристику часто путают с рекомендательным письмом, но на деле это два разных документа. Первый кратко описывает положительные и отрицательные черты характера человека, его навыки и способности. Второй же является больше положительным отзывом о сотруднике и его работе.

Для написания характеристики нет единого формата, только определенные требования к содержанию. Вы можете уточнить у сотрудника куда необходимо предоставить документ, и по возможности узнать требования организации к характеристике. Особенно, если это касается государственных учреждений.

Работодатель несет полную ответственность за данные, указанные в характеристике. Поэтому всегда описывайте реальное положение вещей. При составлении документа в суды, характеристика приобщается к материалам дела. Некорректные данные могут квалифицировать как «заведомо ложные показания», а на работодателя может быть наложен штраф [6].

В качестве примера, для ознакомления с предметной областью, к сожалению, невозможно что-либо представить, так как по подобным запросам в поисковой системе Google, как показано на рисунке 1.1 отображаются только готовые шаблоны и примеры для самостоятельного написания и составления характеристик.

Рисунок 1.1 – Результаты запроса в поисковой системе Google касательно автоматеческой генерации характеристик

1.2 Актуальность и цель работы

Внешние характеристики составляются по просьбам самих работников (для представления по месту запроса), требованиям государственных и иных органов, сторонних организаций. Цели, для которых от работодателя затребуются характеристики, могут быть разнообразными: как для решения бытовых вопросов (например, поступления работника в учебное заведение, получение кредита, пр.), так и для решения вопросов, связанных с принятием в отношении работника государственным (в т.ч. юрисдикционным) или муниципальным органом властного решения (например, о выдаче различных разрешений, применении в отношении работника мер государственного воздействия (награждения или наказания), пр.).

Наиболее сложным для сотрудников кадровых служб является составление характеристик, затребованных адвокатами, органами (должностными лицами), в производстве которых находится дело о правонарушении, совершение которого вменяется работнику, судами (судьями) [12].

Во всех случаях привлечения работника к административной или уголовной ответственности суд и орган, уполномоченный на применение административных наказаний, при разрешении дела и назначении наказания должны учитывать личность виновного, его имущественное положение и другие обстоятельства, имеющие значение для правильного разрешения дела. Для этого они наделены правом истребовать сведения, характеризующие работника, его семейное и имущественное положение, другие. В большинстве случаев в документах с требованиями представить необходимые сведения не указывается, какая именно информация нужна суду или иному юрисдикционному органу. Ввиду этого, сотрудник кадровой службы в лучшем случае может посоветоваться с адвокатом или самим работником, а в худшем — должен самостоятельно определить, какие сведения необходимы запрашивающему характеристику [12].

Многие граждане думают, что характеристика с места работы уже понятие устаревшее, но это ошибочно – документ все еще востребован. В чем же ее важность? Характеристика с места работы – это своеобразная оценка руководством работающего или уволившегося сотрудника. В ней он должен охарактеризовать его профессиональные и личностные качества. Но сегодня отношение к этому документу несколько неоднозначное, так как чаще всего для руководителя это простая формальность, и реальной индивидуальной принадлежности характеристика не раскрывает. В итоге, содержание документа может вызвать сомнение.

При составлении следует придерживаться общепринятых правил, хотя четко установленной формы нет.

На многих предприятиях внутренними правилами предусмотрено предоставление характеристики сотрудникам на бланках с реквизитами организации. Если подобный бланк отсутствует, то в характеристике все равно должны присутствовать реквизиты фирмы, и если документ был затребован официальным запросом, необходимо прописать, куда именно он предоставляется [7].

Целью данной курсовой работы будет разработка приложения, которое позволит просто и легко создавать характеристики работников, используя готовые шаблоны для их составления.

Задачи, которые необходимо решить для реализации проекта:

1) изучить источники относительно характеристик работников;

2) изучить подходы к реализации написания и составления хактеристики;

3) составить предполагаемый функционал приложения;

4) сформировать техническое задание(ТЗ), на основе которого будет разработано приложение;

5) выбрать и изучить технологии для реализации поставленных в ТЗ задач;

6) спроектировать дизайн приложения;

7) разработать приложение;

8) провести тестирование программного продукта;

9) написать руководство пользователя;

Предполагаемый функционал проекта:

1) создание характеристики;

2) возможность задать направление характеристики;

3) запись информации в отформатированный docx-файл;

4) возможность настройки шаблонов.

# 2 ТЕХНИЧЕСКОЕ ЗАДАНИЕ

2.1 Описание области применения и исходных данных приложения

Данное приложение предназначено для удобства работы при составлении характеристик и поддержания общей структурированности, так как персоналу больше можно будет не беспокоиться об оформлении документа, потому данное приложение может применяться в любой сфере деятельности человека, где необходима работа с кадрами.

Исходными данными для приложения являются:

1. Предустановленные ограничения: максимальное и минимальное количество символов в названиях, максимальное количество ярлыков на рабочем столе;
2. Настройки по умолчанию: шрифт, цвет, размер шрифта, его стиль;
3. Сообщения об ошибках выводимые в программе;
4. Изображение ярлыка программы, а также изображения всех инструментальных элементов внутри программного обеспечения;
5. Шаблоны для формирования характеристик.

2.2 Требования к пользовательским интерфейсам

Основные требования к пользовательскому интерфейсу:

1. функциональность;
2. быстрое обучение пользователя;
3. обеспечение высокой скорости работы пользователя;
4. обеспечение защиты от человеческих ошибок;
5. понятность и логичность.

Пользовательский интерфейс должен быть ориентирован на самых обычных пользователей, быть достаточно понятным и удобным. Все элементы управления должны иметь свои всплывающие подсказки, чтобы упростить работу с приложением и его отдельными элементами.

На формах генерации характеристик и настройки не должно быть лишних элементов.

Кнопки смены подпунктов для формирования характеристики, должны быть доступны всегда, чтобы пользователь мог в любой момент вернуться и изменить какие-либо данные. Они не должны перекрывать поля для ввода текста, labels, cheackboxes и должны располагаться в верхних частях формы.

Кнопки «Закрыть» и «Свернуть в трей» должны располагаться на привычных для пользователей местах в выпадающем меню. На дополнительных формахвсё так же должно присутствовать.

Основные части функционального элемента «форма генерации характеристики», представленного на рисунке 2.1:

1) вкладки «Вид», «Общая информация», «Стаж» и «Личностная характеристика»;

2) текстовые поля для заполнения информации;

3) checkboxes;

4) кнопки: «Назад» (возврат к главноме меню), «Вперёд»;

5) кнопки «закрыть» и «свернуть в трей» на привычном для пользователя месте.

Рисунок 2.1 – Примерный макет формы генерации характеристики

Основные части функционального элемента «форма настройки», представленного на рисунке 2.2:

1) поля для изменения шаблонов;

2) вкладки «Вид», «Общая информация», «Стаж» и «Личностная характеристика»;

3) кнопка «Назад»;

4) кнопка «Сохранить изменения».

Рисунок 2.2 – Примерный макет формы авторизации

Основные части функционального элемента «форма главное меню», представленного на рисунке 2.2:

1) кнопка «Генерация характеристики»

3) кнопка «Настройки»;

4) кнопка «Выход».

Рисунок 2.2 – Примерный макет формы главное меню

2.3 Требования к аппаратным, программным и коммуникационным интерфейсам

Для установки и работы приложения необходимо иметь вычислительную систему следующей минимальной аппаратной конфигурации:

1) процессор: 1.0 ГГц;

2) оперативная память: 512 Мб;

3) свободное место на жёстком диске: 5 Мб;

4) видеоадаптер: 64 Мб;

5) разрешение экрана 1920х1080 пикселей;

6) наличие основных устройств ввода-вывода (мышь, клавиатура, монитор).

Необходимое программное обеспечение:

1. операционная система Windows не ниже версии 7;
2. установленный .NET Framework не ниже версии 4.7.2.

2.4 Требования к пользователям продукта

Приложение предназначено для любой категории пользователей. Для пользования приложением необязательно обладать какими-либо специальными знаниями. Всё что необходимо пользователю, это умение пользоваться устройствами ввода-вывода, такими как компьютерная мышь и клавиатура.

2.5 Требования к адаптации на месте

Для установки приложения не требуются инсталляторы. Достаточно скачать директорию с программным обеспечением на свой компьютер и запустить приложение.

2.6 Функции продукта

1. Создание характеристики: для создания характеристики необходимо нажать кнопку «Генерация характеристики» и заполнить советующие поля всех 4-х вкладок;
2. Редактирование шаблонов: для редактирования шаблона характеристики необходимо нажать кнопку «Настройки» и заполнить советующие поля 4-х вкладок (заполнение может быть частичным, не обязательно изменять все параметры).

2.7 Ограничения

1. При некорректном изменении шаблона в настройках – результирующий документ также будет некорректно оформлен.
2. При частичном либо не полном заполнении формы для генерации характеристики – результитующий документ также будет частично, либо не полностью заполнен.
3. Есть поля, что обязательны к заполнению, их содержимое не может быть опущено, при их не заполнении будет появляться уведомление, требующее заполнить все поля.
4. При не корректном заполнении полей появляется уведомление об их не корректности, а сами формы подсвечиваются красным цветом.3 ОБОСНОВАНИЕ ВЫБОРА ИНСТРУМЕНТАЛЬНЫХ СРЕДСТВ

# 3.1 Интегрированная среда разработки Visual Studio

Система Microsoft Visual Studio является самой сложной интегрированной средой разработки (integrated development enviroment - IDE), доступной для программистов в настоящее время. Она является результатом долгой истории развития языков программирования и интерфейсов и вобрала в себя достижения многих сред разработки программного обеспечения [7].

Разработана Visual Studio была в 1997 году и тогда это была первая среда разработки, в которой были собраны различные средства разработки ПО. Сама по себе, студия написана на языках С++ и С#.

Существует огромное множество сред разработки программного обеспечения, однако, почему именно Visual Studio является такой удобной при разработке приложения с использованием C подобного языка. Рассмотрим основные достоинства Microsoft Visual Studio перед другими средами разработки:

1. Огромная библиотека расширений которая постоянно увеличивается;
2. Настраиваемая панель и закрепляемые окна: возможность легко подстроить визуальный интерфейс сильно влияет на скорость и комфортабельность разработки;
3. Простой рабочий процесс и файловая иерархия: отличные помощники при проектировании приложения в стиле ООП;
4. Статистика мониторинга производительности в режиме реального времени: возможность на ходу проверить количество ресурсов которые тратит компьютер для поддержания работы программного обеспечения;
5. Инструменты автоматизации;
6. Легкий рефакторинг и вставка фрагментов кода: незаменимая помощь при разработке приложений;
7. Поддержка разделенного экрана;
8. Список ошибок, который упрощает отладку: возможность осуществлять быстрый и эффективный поиск ошибок, а так же упрощает их устранение.

По состоянию на 2019 год, Visual Studio и её аналог, созданный Microsoft занимают лидирующие позиции среди IDE. Это обуславливается доступностью и гибкостью этих сред разработки, широкими возможностями для расширения функционала под конкретные потребности разработчика за счет плагинов [2].

Рисунок 3.1 – Рейтинг сред разработки

Так же, Visual Studio лучшая среда разработки для работы на таких языках программирования как C# и С++.

# 3.2 Язык программирования С#

С# появился в 2000 году и был разработан компанией Microsoft, чтобы составить конкуренцию SunMicrosystem и их объектно-ориентированному языку Java. Он стал частью новой технологии Microsoft, названной .NET. В рамках этой технологии предусмотрена единая среда выполнения программ (Common Language Runtime, CLR), написанных на разных языках программирования. Одним из таких языков, основным в этой среде, и является C#. Этот язык был создан на основе языка С++ и стал одним из лучших объектно-ориентированных языков.

С# — простой, современный, объектно-ориентированный язык с безопасной системой типов, происходящий от С и С++. С# будет удобен и понятен для программистов, знающих С и С++. С# сочетает продуктивность Visual Basic и мощность С++.

Каковы же преимущества этого языка перед другими ООП языками, рассмотрим следующие достоинства:

1. Единицей компиляции является файл. Файл может содержать одно или несколько описаний типов: классов, интерфейсов, структур, перечислений, типов-делегатов, с указанием (или без указания) об их распределении по пространствам имен;
2. Структурированные типы: классы и интерфейсы, одномерные и многомерные массивы, строки, структуры, перечисления, несовместимые с целыми;
3. Типы подразделяются на ссылочные (классы, интерфейсы, массивы, делегаты) и типы-значения (элементарные типы, перечисления, структуры). Объекты ссылочных типов размещаются в динамической памяти (куче), а переменные ссылочных типов являются, по сути, указателями на эти объекты;
4. Автоматическая сборка мусора;
5. Обработка исключений;
6. Свойства - элементы классов (объектов), доступ к которым осуществляется так же, как и к полям, но реализуется неявно вызываемыми подпрограммами get и set;
7. События - элементы классов процедурного типа (делегаты), к которым вне класса, где они определены, применимы только операции += и –=, позволяющие добавить или удалить методы-обработчики событий для объектов данного класса [7].

На графике, представленном на рисунке 3.2 можно увидеть, что С# находится в тройке лидеров среди всех языков программирования. Это говорит о том, что язык является одним из самых признанных языков в мире на данный момент и огромное множество разрабатываемых приложений создается именно на этом языке программирования.

Рисунок 3.2 – Рейтинг языков программирования

Однако основной причиной выбора этого языка программирования является то, что на этом языке построена система построения клиентских приложений WPF. И именно в WPF реализуется используемый в курсовой работе архитектурный паттерн MVVM (Model-View-ViewModel).

# 3.3 Система для построения клиентских приложений Windows WPF

Windows Presentation Foundation (WPF) - это платформа пользовательского интерфейса для создания клиентских приложений для настольных систем. Платформа разработки WPF поддерживает широкий набор компонентов для разработки приложений, включая модель приложения, ресурсы, элементы управления, графику, макет, привязки данных, документы и безопасность. Она является частью платформы .NET Framework. WPF использует расширяемый язык разметки для приложений (XAML), чтобы предоставить декларативную модель для программирования приложений [4].

WPF развивается вместе с фреймворком .NET и имеет те же версии. Первая версия WPF 3.0 вышла вместе с .NET 3.0 и операционной системой Windows Vista в 2006 году. С тех пор платформа последовательно развивается. Последняя версия WPF 4.6 вышла параллельно с .NET 4.6 в июле 2015 года, ознаменовав девятилетие данной платформы.

Почему не Window Forms? На самом деле ответ довольно прост. WPF обладает колоссальным набором инструментов для работы с формами и графикой в отличие от WinForms. Также, WPF предоставляет некоторые функции для реализации ООП паттернов, что позволяет более эффективно разрабатывать приложения, используя именно эту систему, а не его предшественника – Windows Forms. И, как было сказано выше, именно в WPF реализуется архитектурный паттерн MVVM, используемый при разработке приложения. Преимущества WPF:

1. Для создания логики приложений используются стандартные языки .NET: C# и Visual Basic;
2. Обладает возможностью декларативного определения графического интерфейса с помощью специального языка разметки XAML, основанном на xml и предоставляет альтернативу программному созданию графики и элементов управления, а также возможность комбинировать XAML и C#/VB.NET;
3. Независимость от разрешения экрана: приложения разработанные в WPF легко масштабируются под экраны с разными разрешениями, так-как все элементы измеряются в независимых от устройства единицах;
4. Возможность создания трёхмерных моделей, привязка данных, стили, шаблоны, темы;
5. Богатые возможности по созданию различных приложений: поддержка мультимедиа, двухмерной и трёхмерной графики, создание анимации, возможность создания собственных элементов управления;
6. Аппаратное ускорение графики: WPF работает напрямую с DirectX, а потому повышается производительность и графика становится более плавной.

# 4 РАЗРАБОТКА ПРИЛОЖЕНИЯ АВТОМАТИЧЕСКАЯ ГЕНЕРАЦИЯ ХАРАКТЕРИСТИК

4.1 Входные и выходные данные приложения

Входные данные приложения: все данные о пользователе, изменяемые настройки, создание рабочего стола и его редактирование, создание ярлыка и его редактирование.

Выходные данные: вывод данных о пользователе на экран, отображение картинок рабочих столов и ярлыков, применение настроек. Запись в базу данных информации о пользователе, его рабочих стола, ярлыках и их настройках.

4.2 Проектирование структуры приложения

Структурно приложение состоит из следующих классов:

1. IGenerate.cs
2. Generate.cs
3. ICheckValidity.cs
4. CheckValidity.cs
5. MainWindow.xaml.cs
6. Characteristics\_Maker.xaml.cs
7. Settings.xaml.cs

Схема взаимодействия классов приведена на рисунке 4.1:

Рисунок 4.1 Диаграмма классов

* 1. Описание объектов и их взаимодействия для ООП

Запись введённой информации для генерации характеристики в docx-файл

Первым делом создаём переменную filename, где будет храниться путь и имя файла. Далее, создаём сам документ, используя заранее скаченную и подключенную библиотеку DocX. После форматируем вид заголовка файла: шрифт, размер текста, позицию и цвет. После форматируем остальной текст, а потом формируем всё в единую строку, открываем файл для записи и сохраняем изменения. Открываем файл для проверки.

private Boolean WriteToDocFile()

{

String filename = @"test.docx";

var doc = DocX.Create(filename);

//Formatting Title

Formatting titleFormat = new Formatting();

//Specify font family

titleFormat.FontFamily = new Font("Times New Roman");

//Specify font size

titleFormat.Size = 14D;

titleFormat.Position = 40;

titleFormat.FontColor = System.Drawing.Color.Black;

Paragraph paragraphTitle = doc.InsertParagraph(types\_[2], false, titleFormat);

Paragraph paragraphTitle2 = doc.InsertParagraph(date, false, titleFormat);

paragraphTitle.Alignment = Alignment.center;

paragraphTitle2.Alignment = Alignment.left;

Paragraph paragraphTitle3 = doc.InsertParagraph(header, false, titleFormat);

paragraphTitle3.Alignment = Alignment.center;

//Formatting Text Paragraph

Formatting textParagraphFormat = new Formatting();

//font family

textParagraphFormat.FontFamily = new Font("Times New Roman");

//font size

textParagraphFormat.Size = 14D;

//Spaces between characters

textParagraphFormat.Spacing = 1.5;

textParagraphFormat.Position = 10;

textParagraphFormat.FontFamily = new Font("Times New Roman");

var tmp = first + worker\_name\_ + ", " + birth\_date\_ + second+", "+third+education\_+"."

+ Environment.NewLine+family\_position\_+ Environment.NewLine + four+ accept\_date\_+five+new\_position\_ + Environment.NewLine +

six+seven+eight+nine+accur\_+attantive\_+fame\_+respons\_+ Environment.NewLine + ten + Environment.NewLine + Environment.NewLine +

types\_[4]+"\t"+types\_[3];

//Insert text

doc.InsertParagraph(tmp, false, textParagraphFormat);

doc.Save();

Process.Start("WINWORD.EXE", filename);

return true;

}

Полный код класса «Generate.cs» можно посмотреть в Листинге Б.3.

# 5 ТЕСТИРОВАНИЕ ПРОГРАММНОГО ПРОДУКТА

5.1 Аппаратные и программные средства создания и эксплуатации интернет-приложения

Для работы с приложением необходимо иметь вычислительную систему следующей базовой конфигурации:

1) процессор: 1.0 ГГц;

2) оперативная память: 512 Мб;

3) свободное место на жёстком диске: 200ы Мб;

4) видеоадаптер: 512 Мб;

5) разрешение экрана 1920х1080 пикселей;

6) наличие основных устройств ввода-вывода (мышь, клавиатура, монитор).

Необходимое программное обеспечение:

1. операционная система Windows не ниже версии 7;
2. установленный .NET Framework не ниже версии 4.7.2.

Для работы с кодом программы вычислительную систему следующей базовой конфигурации:

1) процессор: 2.0 ГГц;

2) оперативная память: 2 Гб;

3) свободное место на жёстком диске: 2 Гб;

4) видеоадаптер: 512 Мб;

5) разрешение экрана 1920х1080 пикселей;

6) наличие основных устройств ввода-вывода (мышь, клавиатура, монитор).

Необходимое программное обеспечение:

1. операционная система Windows не ниже версии 7;
2. установленный .NET Framework не ниже версии 4.7.2;
3. Visual Studio не ниже версии 2020 года;
4. Установленная система для создания пользовательских приложений WPF;

5.2 Описание контрольных примеров

После запуска приложения отображается окно главного меню (рисунок А.1)

Для создания характеристики для работника необходимо нажать на кнопку «Сгенерировать характеристику», открывается окно «Генерации отчёта» с 4-мя вкладками: «Вид», «Общая информация», «Стаж» и «Личностная характеристика» (рисунок А.2, рисунок А.4, рисунок А.6 и рисунок А.8).

При попытке перейти на любую последующую вкладку без полного и корректного заполнения необходимых полей, будет выдаваться ошибка (сообщение) с просьбой заполнить необходимые поля, которые в этот момент будут подсвечены красным цветом (рисунок А.3, рисунок А.5 и русунок А.7).

Исключение является последняя вкладка (рисунок А.8), где обязательных полей нет. Там же и находится кнопка «Генерация». При нажатии на эту клавишу начинается генерация отчёта на основе встроенных шаблонов. По умолчанию, документ записывается в файл test.docx и хранится в корне каталога проекта, после генерации файл автоматически открывается, чтобы его можно было отредактировать (если требуется) и удостовериться в правильности указанных данных (рисунок А.9).

Нажав на кнопку «Назад» пользователя возвращает на главное меню (рисунок А.1).

На главном меню (рисунок А.1) нажав на кнопку «Настройки» открывается форма «Настроек» (рисунок А.10).

В форме настроек (рисунок А.10) есть возможность изменить шаблон для личностных характеристик. При наведении указателя мышки на область текстового поля в всплывающей подсказке отображается текущее значение шаблона.

Кнопка «Сохранить» этой же формы изменяет шаблоны для личностных характеристик.

При попытке изменить значения шаблонов на пустые будет выводиться ошибка (сообщение) о необходимости заполнения всех строк (рисунок А.11).

# ЗАКЛЮЧЕНИЕ

В результате работы было разработано приложение, которое, благодаря своему функционалу может стать незаменимым помощником для работников отдела кадров при написании характеристик на работников и позволяет пользователям легко и просто создавать готовые документы, используя встроеные шаблоны.

Во время разработки приложения была изучена предметная область написания характеристик. Была изучена документация языка программирования C#, фреймворка .NET Framework, системы для построения клиентских приложений WPF, изучена интегрированная среда разработки Visual Studio. Была изучена документация по объектно-ориентированному программированию. Все поставленные задачи были выполнены.

При разработке приложения, были подробно изучены методы работы с языком программирование С#, системой клиентских приложений WPF.

В будущем, планируется расширение функционала приложения и увеличение кросс-платформенности.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Как правильно написать характеристику сотруднику с места работы

URL: http://pilotbiz.ru/xarakteristik/#1 (Дата обращения: 28.03.2020)

1. Персональный блог… Самые популярные инструменты и средства разработки //

URL: https://igorosa.com/most-popular-tools-and-tools-development/ (дата обращения: 02.03.2020).

1. Пример составления характеристики с места работы (образец)

URL: https://www.papajurist.ru/job/primer-harakteristiki/ (Дата обращения: 28.03.2020)

1. Приступая к работе (WPF) // Microsoft – официальная домашняя cтраница

URL: https://docs.microsoft.com/ru-ru/dotnet/framework/wpf/getting-started/ (дата обращения: 04.03.2020).

1. Работа с Visual Studio. // Professor Web

URL: https://professorweb.ru/my/programs/visual-studio/level1/ (дата обращения: 20.03.2020).

1. Характеристика сотрудника с места работы: как ее написать + примеры

URL: https://hurma.work/ru/blog/harakteristika-sotrudnika-s-mesta-raboty-kak-ee-napisat-primery/ (Дата обращения: 28.03.2020)

1. Язык программирования С# // Язык программирования С#-Критическая оценка

URL: http://www.uni-vologda.ac.ru/~c3c/articles/CScritics.htm (дата обращения: 17.03.2020).

# ПРИЛОЖЕНИЕ Б Фрагменты листинга

Листинг Б.1 – Файл «MainWindow.xaml.css»

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

private void Exit\_Click(object sender, RoutedEventArgs e)

{

this.Close();

}

private void Setting\_Click(object sender, RoutedEventArgs e)

{

this.Hide();

Settings settings = new Settings(this);

settings.Show();

}

private void Submit\_Click(object sender, RoutedEventArgs e)

{

this.Hide();

Characteristics\_Maker characteristics = new Characteristics\_Maker(this);

characteristics.Show();

}

}

Листинг Б.2 – Скрипт «Characteristics\_Maker.xaml.css»

public partial class Characteristics\_Maker : Window

{

private MainWindow main\_ = new MainWindow();

private CheckValidity check = new CheckValidity();

private IGenerate gen;

public Characteristics\_Maker()

{

InitializeComponent();

}

public Characteristics\_Maker(MainWindow main\_)

{

InitializeComponent();

DataContext = this;

this.main\_ = main\_;

}

private void Back\_Click(object sender, RoutedEventArgs e)

{

main\_.Show();

this.Close();

}

private void Next\_ToGeneral(object sender, RoutedEventArgs e)

{

if (type.SelectedValue != null && way.SelectedValue != null && company.Text.Trim() != ""

&& giver.Text.Trim() != "" && giver\_position.Text.Trim() != "")

{

type.BorderBrush = System.Windows.Media.Brushes.Green;

way.BorderBrush = System.Windows.Media.Brushes.Green;

company.BorderBrush = System.Windows.Media.Brushes.Green;

giver.BorderBrush = System.Windows.Media.Brushes.Green;

giver\_position.BorderBrush = System.Windows.Media.Brushes.Green;

Characteristics.SelectedIndex++;

}

else

{

type.BorderBrush = System.Windows.Media.Brushes.Red;

type.ToolTip = "Поле обязательно для заполнения.";

way.BorderBrush = System.Windows.Media.Brushes.Red;

way.ToolTip = "Поле обязательно для заполнения.";

company.BorderBrush = System.Windows.Media.Brushes.Red;

company.ToolTip = "Поле обязательно для заполнения.";

giver.BorderBrush = System.Windows.Media.Brushes.Red;

giver.ToolTip = "Поле обязательно для заполнения.";

giver\_position.BorderBrush = System.Windows.Media.Brushes.Red;

giver\_position.ToolTip = "Поле обязательно для заполнения.";

MessageBox.Show("Заполните все поля!");

}

}

private void Generate\_Click(object sender, RoutedEventArgs e)

{

try

{

gen = new Generate(combineTypeCharacteristicsIntoList(), worker\_name.Text, family\_state.Text,

birth\_date.SelectedDate.Value, education.Text, war\_position.Text, job\_timing.Text,

job\_accept.SelectedDate.Value, promotion.Text, special\_echivements.Text,

Convert.ToBoolean(courses\_check.IsChecked), additional\_curses.Text,

Convert.ToBoolean(fines.IsChecked), combinePersonalCharacteristicsIntoList()

);

if (gen.GenerateCharacteristic())

{

MessageBoxResult result = MessageBox.Show("Характеристика сгенерирована! Хотите создать ещё одну?", "Генератор характеристик", MessageBoxButton.YesNo);

switch (result)

{

case MessageBoxResult.Yes:

ClearAllItems();

Characteristics.SelectedIndex = 0;

break;

case MessageBoxResult.No:

Characteristics.Items.Clear();

main\_.Show();

this.Close();

break;

}

}

} catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

}

private void Next\_ToJob(object sender, RoutedEventArgs e)

{

if (worker\_name.Text.Trim() != "" && family\_state.SelectedValue != null &&

birth\_date.SelectedDate != null && education.Text.Trim() != "")

{

if (check.checkDateValidation(birth\_date.SelectedDate.Value))

{

birth\_date.BorderBrush = System.Windows.Media.Brushes.Green;

family\_state.BorderBrush = System.Windows.Media.Brushes.Green;

worker\_name.BorderBrush = System.Windows.Media.Brushes.Green;

education.BorderBrush = System.Windows.Media.Brushes.Green;

Characteristics.SelectedIndex++;

}

else

{

birth\_date.BorderBrush = System.Windows.Media.Brushes.Red;

birth\_date.ToolTip = "Не корректная дата";

}

}

else

{

birth\_date.BorderBrush = System.Windows.Media.Brushes.Red;

birth\_date.ToolTip = "Поле обязательно для заполнения.";

family\_state.BorderBrush = System.Windows.Media.Brushes.Red;

family\_state.ToolTip = "Поле обязательно для заполнения.";

worker\_name.BorderBrush = System.Windows.Media.Brushes.Red;

worker\_name.ToolTip = "Поле обязательно для заполнения.";

education.BorderBrush = System.Windows.Media.Brushes.Red;

education.ToolTip = "Поле обязательно для заполнения.";

MessageBox.Show("Заполните все поля!");

}

}

private void Next\_ToPersonal(object sender, RoutedEventArgs e)

{

if (job\_timing.Text.Trim() != "" && job\_accept.SelectedDate != null )

{

if(courses\_check.IsChecked.Value && additional\_curses.Text!="")

{

if(check.checkDateValidation(job\_accept.SelectedDate.Value) &&

check.checkDigitValidation(job\_timing.Text))

{

job\_timing.BorderBrush = System.Windows.Media.Brushes.Green;

job\_accept.BorderBrush = System.Windows.Media.Brushes.Green;

additional\_curses.BorderBrush = System.Windows.Media.Brushes.Red;

courses\_check.BorderBrush = System.Windows.Media.Brushes.Red;

Characteristics.SelectedIndex++;

}

else

{

job\_timing.BorderBrush = System.Windows.Media.Brushes.Red;

job\_timing.ToolTip = "Не корректные данные";

job\_accept.BorderBrush = System.Windows.Media.Brushes.Red;

job\_accept.ToolTip = "Не корректные данные";

}

} else if (additional\_curses.Text == "" && !additional\_curses.IsEnabled)

{

if (check.checkDateValidation(job\_accept.SelectedDate.Value) &&

check.checkDigitValidation(job\_timing.Text))

{

job\_timing.BorderBrush = System.Windows.Media.Brushes.Green;

job\_accept.BorderBrush = System.Windows.Media.Brushes.Green;

Characteristics.SelectedIndex++;

}

else

{

job\_timing.BorderBrush = System.Windows.Media.Brushes.Red;

job\_timing.ToolTip = "Не корректные данные";

job\_accept.BorderBrush = System.Windows.Media.Brushes.Red;

job\_accept.ToolTip = "Не корректные данные";

}

}

else

{

additional\_curses.BorderBrush = System.Windows.Media.Brushes.Red;

courses\_check.BorderBrush = System.Windows.Media.Brushes.Red;

}

}

else

{

job\_timing.BorderBrush = System.Windows.Media.Brushes.Red;

job\_timing.ToolTip = "Поле обязательно для заполнения.";

job\_accept.BorderBrush = System.Windows.Media.Brushes.Red;

job\_accept.ToolTip = "Поле обязательно для заполнения.";

MessageBox.Show("Заполните все поля!");

}

}

private List<Boolean> combinePersonalCharacteristicsIntoList()

{

List<Boolean> characteristics = new List<Boolean>()

{

char\_1.IsChecked.Value,

char\_2.IsChecked.Value,

char\_5.IsChecked.Value,

char\_6.IsChecked.Value,

};

return characteristics;

}

private List<String> combineTypeCharacteristicsIntoList()

{

List<String> tmp = new List<string>()

{

type.Text,

way.Text,

company.Text,

giver.Text,

giver\_position.Text,

};

return tmp;

}

private Boolean ClearAllItems()

{

type.Text = "";

way.Text = "";

company.Text = "";

giver.Text = "";

giver\_position.Text = "";

worker\_name.Text = "";

family\_state.SelectedItem = null;

birth\_date.SelectedDate = null;

education.Text = "";

war\_position.Text = "";

job\_timing.Text = "";

job\_accept.SelectedDate = null;

promotion.Text = "";

special\_echivements.Text = "";

additional\_curses.Text = "";

this.courses\_check.IsChecked = false;

this.fines.IsChecked = false;

this.char\_1.IsChecked = false;

this.char\_2.IsChecked = false;

this.char\_5.IsChecked = false;

this.char\_6.IsChecked = false;

return true;

}

}

Листинг Б.3 – Скрипт «Generate.css»

public class Generate : IGenerate

{

#region Private Fields

private String date = DateTime.Now.Date.ToString().Replace('/','.');

private String header = "ХАРАКТЕРИСТИКА";

private String first = "Харрактеристика дана ";

private String second = "года рождения ";

private String third = "образование ";

private String four = "Принят/та в ";

private String five = "на должность ";

private String six = "За время работы на предприятии проявил/ла себя, как работник ";

private String seven = " ответственный и дисциплинированный. ";

private String eight = "Характеризуется как ";

private String nine = "отзывчивый, честный и добрый человек ";

private String ten = "Дисциплинарным взысканиям не подвергался/лась. ";

private String accur\_ = " Во время работы проявил/ла себя, как человек предельно осторожный/ая и аккуратный/ая. ";

private String attantive\_ = " Показывал/ла рвение и усидчивость во время работы. ";

private String fame\_ = " Есть хорошие задатки лидера. ";

private String respons\_ = " Человек крайне ответственный и внимательный. ";

//type

private List<String> types\_; //5 issues

//basic info

private String worker\_name\_;

private String family\_position\_;

private DateTime birth\_date\_;

private String education\_;

private String war\_position\_;

//job

private String job\_timing\_;

private DateTime accept\_date\_;

private String new\_position\_;

private String prof\_skills\_;

private Boolean additional\_skills\_;

private String add\_skills\_;

private Boolean penalty\_;

//personal

private List<Boolean> qualities\_; //4 qualities

#endregion

#region Public Fields

public String AccuracyDesc

{

get { return accur\_; }

set

{

accur\_ = value;

}

}

public String AttantDesc

{

get { return attantive\_; }

set

{

attantive\_ = value;

}

}

public String FameDesc

{

get { return fame\_; }

set

{

fame\_ = value;

}

}

public String RespDesc

{

get { return respons\_; }

set

{

respons\_ = value;

}

}

#endregion

#region Constructors

public Generate() { }

public Generate(

List<String> types, String worker\_name, String family, DateTime birth, String education,

String war\_pos, String job\_timing, DateTime accept\_date, String new\_pos, String prof\_skills,

Boolean additional\_skills, String add\_skils, Boolean penalty, List<Boolean> qualities)

{

types\_ = types;

worker\_name\_ = worker\_name;

family\_position\_ = family;

birth\_date\_ = birth;

education\_ = education;

war\_position\_ = war\_pos;

job\_timing\_ = job\_timing;

accept\_date\_ = accept\_date;

new\_position\_ = new\_pos;

prof\_skills\_ = prof\_skills;

additional\_skills\_ = additional\_skills;

add\_skills\_ = add\_skils;

penalty\_ = penalty;

qualities\_ = qualities;

}

#endregion

#region Public Methods

public bool GenerateCharacteristic()

{

WriteToDocFile();

Check();

return true;

}

#endregion

#region Private Methods

private void Check()

{

if (!penalty\_)

{

ten = "Дисциплинарным взысканиям подвергался/лась. ";

}

if (!qualities\_[0])

{

accur\_ = "";

}

if (!qualities\_[1])

{

fame\_ = "";

}

if (!qualities\_[2])

{

attantive\_ = "";

}

if (!qualities\_[3])

{

respons\_ = "";

}

}

private Boolean WriteToDocFile()

{

String filename = @"C:\Users\oksa0\source\repos\test.docx";

var doc = DocX.Create(filename);

//Formatting Title

Formatting titleFormat = new Formatting();

//Specify font family

titleFormat.FontFamily = new Font("Times New Roman");

//Specify font size

titleFormat.Size = 14D;

titleFormat.Position = 40;

titleFormat.FontColor = System.Drawing.Color.Black;

Paragraph paragraphTitle = doc.InsertParagraph(types\_[2], false, titleFormat);

Paragraph paragraphTitle2 = doc.InsertParagraph(date, false, titleFormat);

paragraphTitle.Alignment = Alignment.center;

paragraphTitle2.Alignment = Alignment.left;

Paragraph paragraphTitle3 = doc.InsertParagraph(header, false, titleFormat);

paragraphTitle3.Alignment = Alignment.center;

//Formatting Text Paragraph

Formatting textParagraphFormat = new Formatting();

//font family

textParagraphFormat.FontFamily = new Font("Times New Roman");

//font size

textParagraphFormat.Size = 14D;

//Spaces between characters

textParagraphFormat.Spacing = 1.5;

textParagraphFormat.Position = 10;

textParagraphFormat.FontFamily = new Font("Times New Roman");

var tmp = first + worker\_name\_ + ", " + birth\_date\_ + second+", "+third+education\_+"."

+ Environment.NewLine+family\_position\_+ Environment.NewLine + four+ accept\_date\_+five+new\_position\_ + Environment.NewLine +

six+seven+eight+nine+accur\_+attantive\_+fame\_+respons\_+ Environment.NewLine + ten + Environment.NewLine + Environment.NewLine +

types\_[4]+"\t"+types\_[3];

//Insert text

doc.InsertParagraph(tmp, false, textParagraphFormat);

doc.Save();

Process.Start("WINWORD.EXE", filename);

return true;

}

#endregion

}