Реферат

Утверд.

Н. контр.

Провер.

Разраб.

*Пацей Н.В.*

*Ковалевская Н.И*.

*Русакович А.Д.*

*Наркевич А.С.*

Ф.И.О.

Подпись

Дата

1

*Реферат*

Лист

1

*ДП 00.00.ПЗ*

Лит.

Листов

*БГТУ 74417033, 2019*

Пояснительная записка содержит 76 страниц, 36 рисунков, 22 таблицы,   
13 источников литературы, 4 приложения.

.NET FRAMEWORK, WPF, ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ, ПРОЕКТИРОВАНИЕ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА, УПРАВЛЕНИЕ ПРОФЕССИОНАЛЬНОЙ ИНФОРМАЦИЕЙ

Целью данного дипломного проектирования является разработка программного средства управления профессиональной информацией о сотрудниках IT компании.

В первом разделе дипломного проекта приведён аналитический обзор литературы и постановка задачи по теме дипломного проекта.

Во втором разделе описан процесс проектирования программного средства.

В третьем разделе описан процесс разработки и программной реализации программного средства.

В четвертом разделе описано тестирование программного средства.

В пятом разделе описана методика установки и использования программного средства.

В шестом разделе приведено технико-экономическое обоснование проекта.

Графическая часть включает:

* логическую схему базы данных;
* диаграмму вариантов использования;
* экранную копию интерфейса приложения;
* блок–схему алгоритма нахождения процента соответствия требованиям;
* диаграмму последовательности;
* диаграмму классов.

**Содержание**

Утверд.

Н. контр.

Провер.

Разраб.

*Пацей Н.В.*

*Ковалевская Н.И*.

*Русакович А.Д.*

*Наркевич А.С.*

Ф.И.О.

Подпись

Дата

2

*Содержание*

Лист

1

*ДП 00.00.ПЗ*

Лит.

Листов

*БГТУ 74417033, 2019*

[Введение 7](#_Toc10154392)

[1 Аналитический обзор литературы и постановка задачи 9](#_Toc10154393)

[1.1 Обзор предметной области 9](#_Toc10154395)

[1.2 Обзор аналогов 9](#_Toc10154396)

[1.3 Постановка задачи 12](#_Toc10154397)

[1.4 Выводы по разделу 12](#_Toc10154398)

[2 Проектирование программного средства 14](#_Toc10154399)

[2.1 Основные задачи разрабатываемого программного средства 14](#_Toc10154401)

[2.2 Структура разрабатываемого программного средства 15](#_Toc10154402)

[2.3 Информационное обеспечение 15](#_Toc10154403)

[2.4 Использованные для разработки технологии 20](#_Toc10154404)

[2.5 Выводы по разделу 22](#_Toc10154405)

[3 Разработка программного средства 23](#_Toc10154406)

[3.1 Структура программного средства 23](#_Toc10154408)

[3.3 Разработка программного кода 24](#_Toc10154409)

[3.4 Выводы по разделу 36](#_Toc10154410)

[4 Тестирование программного средства 37](#_Toc10154412)

[4.1 Негативное тестирование страниц входа и регистрации 37](#_Toc10154413)

[4.2 Негативное тестирования страницы профессиональной информации 38](#_Toc10154414)

[4.3 Негативное тестирование функционала для роли «Менеджер» 41](#_Toc10154415)

[4.4 Выводы по разделу 43](#_Toc10154416)

[5 Методика установки и использования 44](#_Toc10154418)

[5.1 Методика развертывания и требования приложения 44](#_Toc10154419)

[5.2 Руководство пользователя 45](#_Toc10154420)

[5.2.1 Экраны входа и регистрации в приложении 45](#_Toc10154421)

[5.2.2 Руководство пользователя для роли «Администратор» 46](#_Toc10154422)

[5.2.3 Руководство пользователя для роли «Менеджер» 48](#_Toc10154423)

[5.2.4 Руководство пользователя для роли «Сотрудник» 52](#_Toc10154424)

[5.3 Выводы по разделу 56](#_Toc10154425)

[6 Технико-экономическое обоснование проекта 57](#_Toc10154426)

[6.1 Общая характеристика разрабатываемого программного средства 57](#_Toc10154428)

[6.2 Исходные данные и маркетинговый анализ 57](#_Toc10154429)

[6.3 Методика обоснования цены 58](#_Toc10154430)

[6.3.1 Объём программного средства 58](#_Toc10154431)

[6.3.2 Основная заработная плата 59](#_Toc10154432)

[6.3.3 Дополнительная заработная плата 60](#_Toc10154433)

[6.3.4 Отчисления в Фонд социальной защиты населения 60](#_Toc10154434)

[6.3.5 Расходы на материалы 61](#_Toc10154435)

[6.3.6 Расходы на оплату машинного времени 61](#_Toc10154436)

[6.3.7 Прочие прямые затраты 61](#_Toc10154437)

[6.3.8 Накладные расходы 62](#_Toc10154438)

[6.3.9 Сумма расходов на разработку программного средства 62](#_Toc10154439)

[6.3.10 Расходы на сопровождение и адаптацию 62](#_Toc10154440)

[6.3.11 Полная себестоимость 63](#_Toc10154441)

[6.3.12 Определение цены, оценка эффективности 63](#_Toc10154442)

[6.4 Выводы по разделу 64](#_Toc10154443)

[Заключение](file:///C:\Users\Alexey\Desktop\Диплом\Документы\Пояснительная%20записка.docx#_Toc10154444) 65

[Список использованных источников](file:///C:\Users\Alexey\Desktop\Диплом\Документы\Пояснительная%20записка.docx#_Toc10154446) 66

[ПРИЛОЖЕНИЕ А 67](#_Toc10154448)

[ПРИЛОЖЕНИЕ Б 68](#_Toc10154449)

[ПРИЛОЖЕНИЕ В 70](#_Toc10154450)

[ПРИЛОЖЕНИЕ Г 71](#_Toc10154452)

### 

### Введение

Утверд.

Н. контр.

Провер.

Разраб.

*Пацей Н.В.*

*Ковалевская Н.И*.

*Русакович А.Д.*

*Наркевич А.С.*

Ф.И.О.

Подпись

Дата

1

*Введение*

Лист

1

*ДП 00.00.ПЗ*

Лит.

Листов

*БГТУ 74417033, 2019*

Лавина неупорядоченной документации способна легко погрести под собой и маленькую фирму, и транснациональную корпорацию. Продуманная система правления информацией – это ключевая составляющая стратегии развития компании. Труд сотрудников – важный экономический ресурс, которым, как и другими ресурсами, следует распоряжаться с максимальной эффективностью. Это означает, что рутинные задачи, действия, процессы выполнять необходимо максимально четко и оперативно. При необходимости найти подходящего кандидата внутри компании и при отсутствии системы, агрегирующей данные и позволяющей быстро и легко найти нужную информацию данная задача будет являться трудновыполнимой и заставит потратить на себя немало человеко-часов. Автоматизированные системы кадровых операций, давно внедренные в практику, заметно упрощают жизнь руководителя предприятия и HR-службы. Здесь имеется пространство для внедрения информационных технологий, а именно программного средства управления профессиональной информацией о сотрудниках IT компании.

Современный уровень развития информационных технологий позволяет не просто автоматизировать процессы управления информацией о сотрудниках, а строить для управления этими процессами кроссплатформенные решения и даже целые информационные экосистемы, включающие веб-сайты, настольные и мобильные приложения, которые, в свою очередь, могут взаимодействовать с помощью веб-сервисов и сервисов облачных вычислений. Для предприятий, содержащих большое количество сотрудников, внедрение таких решений теперь является скорее производственной необходимостью, нежели трендом, и напрямую влияет на их конкурентоспособность на рынке.

Темой данного дипломного проекта является реализация программного средства управления профессиональной информацией о сотрудниках IT компании, легковесного решения, использующего современные технологии разработки и отвечающего как общим требованиям рынка, так и специфике данного типа предприятия. Конечной целью данного дипломного проекта является готовый программный продукт, производительный, отказоустойчивый, обладающий современным графическим интерфейсом и, что является наиболее важным для заказчика, предоставляющий использующему его предприятию конкурентные преимущества на рынке.

### 1 Аналитический обзор литературы и постановка задачи

Утверд.

Н. контр.

Провер.

Разраб.

*Пацей Н.В.*

*Ковалевская Н.И*.

*Русакович А.Д.*

*Наркевич А.С.*

Ф.И.О.

Подпись

Дата

7

*Аналитический обзор*

*литературы и постановка задачи*

Лист

1

*ДП 01.00.ПЗ*

Лит.

Листов

*БГТУ 74417033, 2019*

### 1.1 Обзор предметной области

Главной задачей приложения является упрощение работы с профессиональными данными сотрудниками компании, представленными на рисунке 1.1, что позволяет сократить до минимума количество времени необходимое на поиск, изменение, сравнение и подбор необходимого кадра на необходимую позицию.

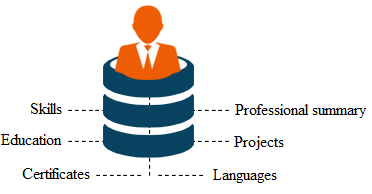


Рисунок 1.1 – Профессиональная информация сотрудника

Профессиональная информация сотрудника включает в себя следующие разделы:

* Professional summary (краткая информация сотрудника о его профессиональной деятельности);
* Skills (информация о его навыках и уровнях владения данными навыками);
* Education (информация о местах получения высшего образования);
* Projects (информация об участии в проектах компании);
* Certificates (информация о полученных сертификатах);
* Languages (информация об иностранных языках и их уровне, которыми обладает сотрудник).

Поиск кандидата на нужную позицию на проекте обычно содержит следующие этапы:

* поиск кандидатов по подходящим техническим навыкам;
* прохождение тестов с помощью которых можно установить уровень знаний и личностных качеств кандидата.
* собеседование, в ходе которого можно узнать необходимые детали;
* прием кандидата в команду проекта.

Некоторые этапы при поиске кандидата могут быть пропущены на усмотрение менеджера.

### 1.2 Обзор аналогов

В данный момент на рынке существует большое количество автоматизированных систем управления профессиональной информацией, предоставляющих вышеперечисленный функционал. Одни коммерческие системы предлагают универсальное решение, легко адаптируемое к любой предметной области, другие – узкоспециализированный продукт, учитывающий все нюансы конкретной предметной области. Большинство таких решений реализовано на основе самых современных технологий, использует облачные вычисления, предоставляет кроссплатформенные клиентские приложения или целые системы, включающие в себя сайты, настольные системы и мобильные приложения.

Наиболее популярными решениями являются:

* HR.my;
* Factorial;
* WebHR.

Рассмотрим HR.my [1]. Пример страницы системы HR.my представлен на рисунке 1.3.

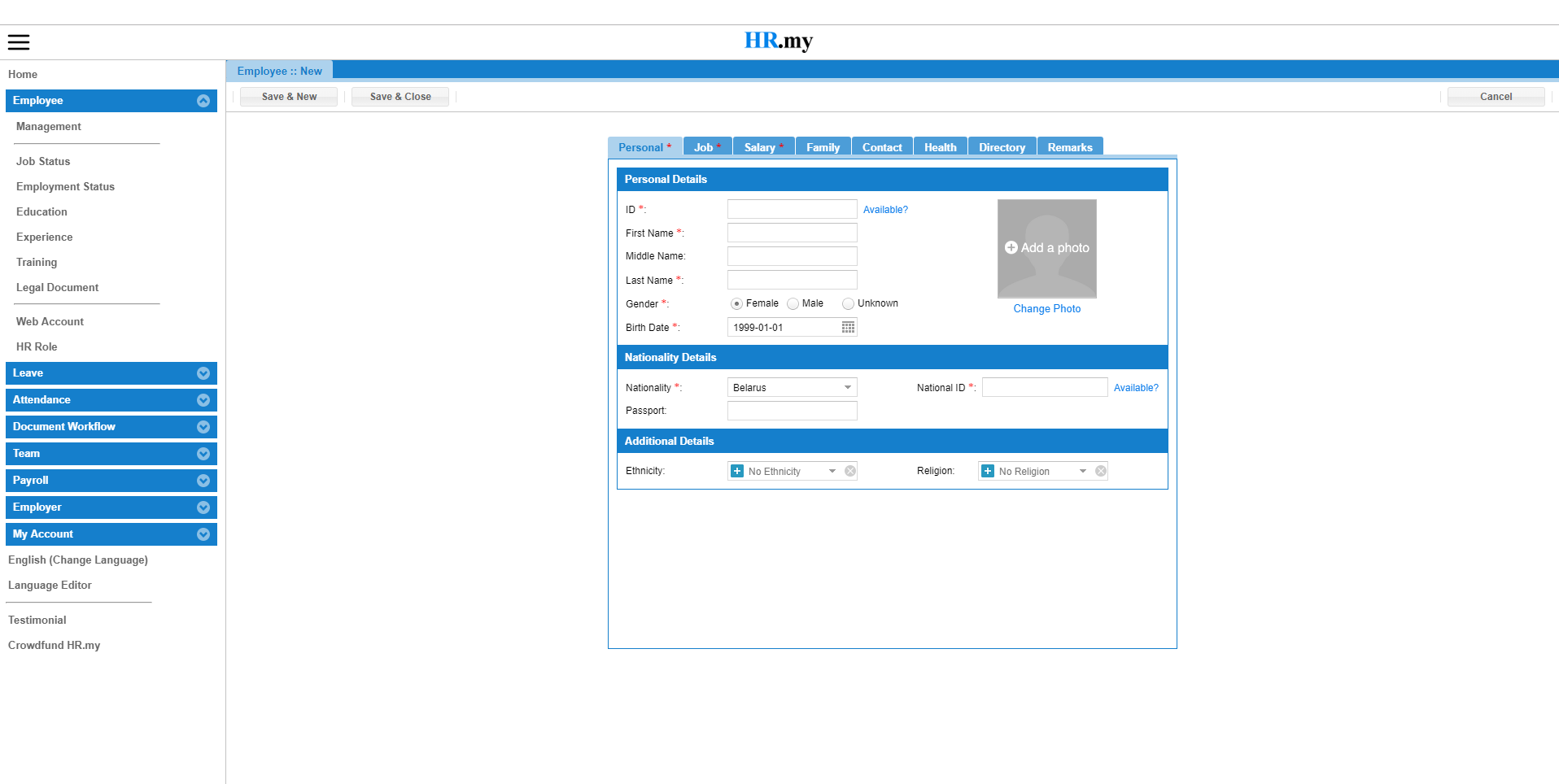


Рисунок 1.3 – Снимок экрана HR.my

Кроме возможности управления профессиональной информацией о сотрудниках данная система поддерживает управление отпусками сотрудников, имеет возможность следить за посещаемостью, хранить файлы сотрудника, поддерживает управление командой и расчёт заработной платы сотрудников. Данная система хоть и обладает большим функционалом, но является слишком громоздкой. Интерфейс данной системы не является интуитивно понятным. Основным недостатком данного решения является отсутствие поддержки поиска сотрудников по профессиональным критериям.

Перейдем к обзору системы Factorial [2]. Пример страницы системы Factorial представлен на рисунке 1.4.

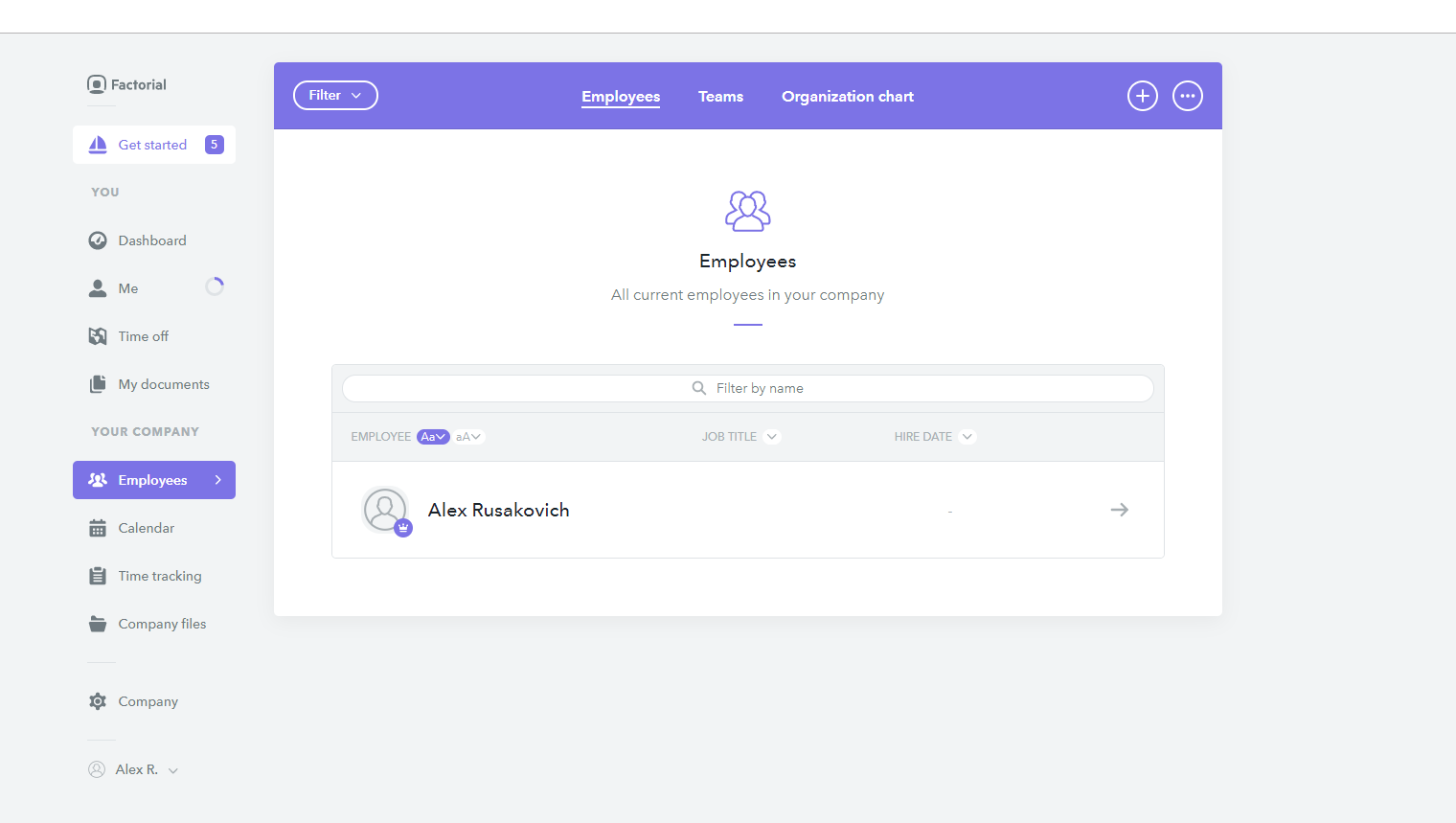


Рисунок 1.4 – Снимок экрана Factorial

Данная система поддерживает управления отпусками сотрудников, имеет возможность хранить необходимые документы в электронном виде, как самого сотрудника, так и компании. Так же данное решение позволяет создавать команды, отслеживать профессиональные связи между сотрудниками, имеет возможность управления ролями сотрудников. Заметным плюсом данной системы является интуитивный и красивый интерфейс.

Но данная система, как и вышеупомянутая, не позволяет осуществлять подбор сотрудников по профессиональным навыкам в случае необходимости заполнения позиций на проекте.

Так же данная система позволяет регистрировать события, указывая его дату, время, место его проведения и описание. Имеет возможность добавлять новости в новостную ленту компании, возможность отслеживать текущие задачи.

Factorial предоставляет инструментарий для управления информацией о компании, управление офисами компаний.

Так же данное решение позволяет управлять текущими настройками аккаунта пользователя. Имеет возможность управления уведомлениями приложения и кастомизацией страницы.

Несмотря на широкий функционал, данная система является легковесной, в связи с чем может не подойти большим компаниям.

Перейдем к обзору системы WebHR [3]. Пример страницы данной системы представлен на рисунке 1.5.

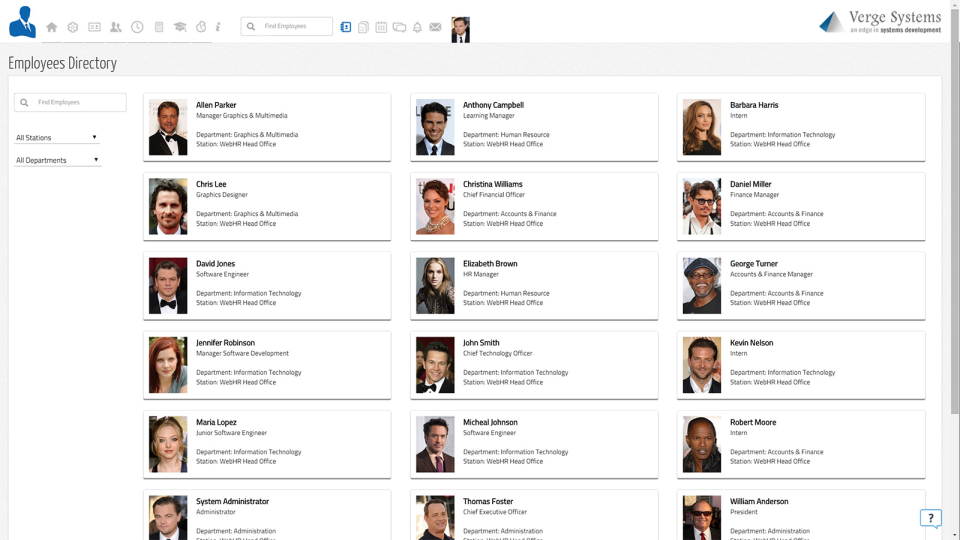


Рисунок 1.5 – Снимок экрана WebHR

Данная система так же обладает возможностью управления посещаемостью сотрудников, платежными ведомостями, управления документами, обладает функционалом рекрутинга. Так же данное решение позволяет отслеживать поставленные сотрудником цели, их прогресс. Поддерживает возможность отслеживать посещаемость по биометрическим данным сотрудника.

WebHR позволяет управлять тренингами и событиями в компании. Имеет возможность генерировать отчеты. Так же данная система имеет возможность резервного копирования данных, что может позволить быстро восстановить утраченную информацию. Клиентами данной системы являются такие компании как автопроизводители Nissan, Infinity, Ducati, Volvo, Renault, компания-производитель шин Bridgestone, компания гостиничного бизнеса Atlantic Hospitality и многие другие.

Основным недостатком данной системы является ее высокая стоимость.

Каждая из перечисленных выше систем имеет свои достоинства и недостатки, причем большинство из них продвигаются на рынке как универсальные, комплексные и легко масштабируемые решения.

Так же данные системы являются либо коммерческими (WebHR), либо бесплатными (HR.my и Factorial), но не имеют возможность добавление нового функционала, в отличие от Open Source решений.

Сравнительный анализ данных систем представлен в таблице 1.1

Таблица 1.1 – Сравнительный анализ систем-аналогов

|  |  |  |  |
| --- | --- | --- | --- |
| Функциональность | HR.my | Factorial | WebHR |
| Бесплатное количество сотрудников | Неограниченно | Неограниченно | 5 |
| Бесплатный объем хранилища данных | Неограниченно | Неограниченно | Неограниченно |
| Бесплатная поддержка | + (Email) | + (Email) | + (Email) |

Окончание таблицы 1.1

|  |  |  |  |
| --- | --- | --- | --- |
| Функциональность | HR.my | Factorial | WebHR |
| Плата за превышение бесплатного количества сотрудников | – | – | $1–7/user/month |
| Персонально отслеживание | + | + | + |
| Трекинг времени | + | – | + |
| Рекрутинг | – | – | + |
| Платежная ведомость | + | + | + |
| Open source | – | – | – |
| Поддержка мобильных приложений | – | – | + |

Некоторые из них имеют высокую стоимость, либо же предлагают избыточный для поставленной задачи функционал и предъявляют достаточно высокие требования к системе, в связи с чем было принято решение разработать собственное программное средство учитывая достоинства и недостатки рассмотренных решений.

### 1.3 Постановка задачи

Целью данного дипломного проекта является разработка программного средства управления профессиональной информацией о сотрудниках IT компании, то есть легковесного специализированного решения, отвечающего как общим требованиям рынка к системам такого рода, так и требованиям заявленной предметной области, и при этом не содержащего лишнего для предприятия функционала, снизив тем самым минимальные системные требования.

В задачи данного программного средства входит:

– авторизация и регистрация сотрудников предприятия, разделение прав доступа, а именно для администраторов, менеджеров и сотрудников;

– защита от несанкционированного доступа к данным предприятия, к персональным данным сотрудников, опирающаяся на систему авторизации и разделения прав доступа;

– удобное управление профессиональной информацией о сотрудниках, включающее операции чтения, добавления, удаления, редактирования и поиска;

– возможность подбора наиболее подходящего кандидата на проект;

* генерация поискового отчета по каждому кандидату;

– удобный и современный графический интерфейс;

– соответствие минимальным техническим и системным требованиям.

### 1.4 Выводы по разделу

В данном разделе был произведен обзор предметной области. Был предоставлен теоретический и графический материалы об основных аспектах профессиональной информации сотрудника. Были перечислены основные этапы отбора необходимого кандидата на проект.

Также в данном разделе был произведен обзор наиболее популярных систем–аналогов с целью выявления преимуществ и недостатков. Были рассмотрены возможности каждой системы. Так же приведен сравнительный анализ данных систем по функциональности.

В данном разделе так же была описана постановка задачи, в ходе которой была определена цель дипломного проекта, а именно разработка программного средства управления профессиональной информацией о сотрудниках IT компании.

### 2 Проектирование программного средства

Утверд.

Н. контр.

Провер.

Разраб.

*Пацей Н.В.*

*Ковалевская Н.И*.

*Русакович А.Д.*

*Наркевич А.С.*

Ф.И.О.

Подпись

Дата

9

*Проектирование программного средства*

Лист

1

*ДП 02.00.ПЗ*

Лит.

Листов

*БГТУ 74417033, 2019*

### 2.1 Основные задачи разрабатываемого программного средства

Программное средство управления профессиональной информацией о сотрудниках IT компании предоставляет пользователю удобный графический интерфейс. Для хранения данных о пользователях, их ролях, навыках, местах учебы, сертификатах, языках и другой вспомогательной информации система использует базу данных.

Сотрудник вводит данные своей учетной записи для авторизации в системе и при условии успешного входа приступает к работе. Программное средство поддерживает несколько ролей: администратор, менеджер, сотрудник. Оно позволяет с помощью панели навигации, расположенной в левой части интерфейса, переключаться между окнами, которые соответствую роли текущего пользователя.

Основные задачи приложения для роли «Администратор» можно выразить с помощью диаграммы вариантов использования. Диаграмма вариантов использования программного средства для роли «Администратор» представлена на рисунке 2.1.

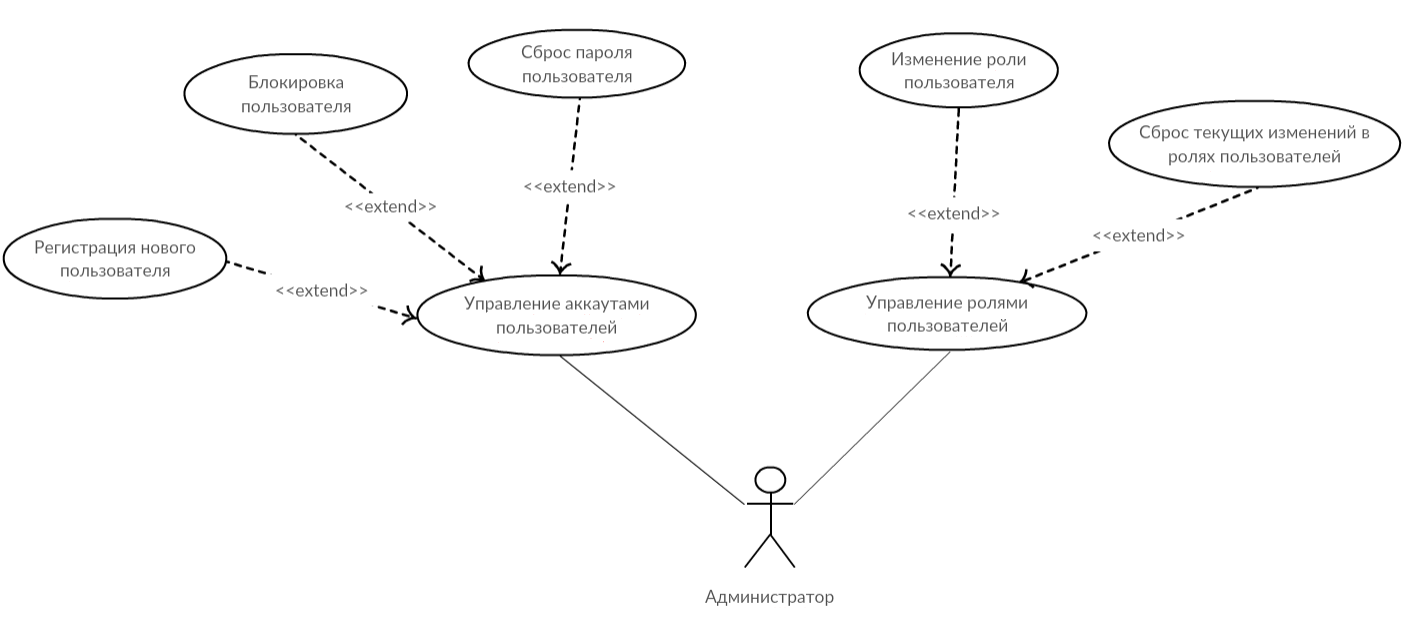


Рисунок 2.1 – Диаграмма вариантов использования для роли «Администратор»

Исходя из диаграммы вариантов использования можно заметить, что основными задачами приложения для роли «Администратор является предоставление функционала для управления ролями и аккаунтами пользователей.

### 2.2 Структура разрабатываемого программного средства

В структуре разрабатываемого программного средства можно условно выделить следующие основные блоки:

* база данных, которые хранит информацию о пользователях, их ролях, их профессиональной информации и других вспомогательных данных;
* модель данных, которая представляет собой логику работы с данными и описание фундаментальных данных, необходимых для работы приложения;
* модель представления, которая является посредником между слоем модели данных и слоем пользовательского интерфейса;
* графический пользовательский интерфейс, который обеспечивает взаимодействие пользователя с системой.

Схематично структура программного средства представлена на рисунке 2.2.



Рисунок 2.2 – Общая структура программного средства

Данная структура программного средства является типовой структурой приложения, использующего шаблон проектирования Model-View-ViewModel.

### 2.3 Информационное обеспечение

Информационное обеспечение – это данные разной формы и источников, поступающие на вход какой-либо информационной системы либо производимые данной информационной системой.

Основным хранилищем данных программного средства является база данных. База данных программного средства предназначена для хранения данных о пользователях, информации о технических навыках, образовании, сертификатах, иностранных языках, проектах. Помимо этого, в база данных имеет таблицы с предопределенными данными, такими как роли сотрудников, уровни владения иностранными языками и техническими навыками. Хранение медиафайлов в базе данных не предусмотрено.

База данных строится с учетом следующих особенностей задачи:

– она должна содержать информацию о пользователю, такую как имя, фамилия, дата рождения, роль, должность, информацию о логине и пароле, соли, использованной для хеширования пароля, информацию о текущем статусе аккаунта пользователя;

– во избежание дублирования данных перечисления, такие как уровни владения навыками и иностранными языками, роли и должности сотрудника, технические навыки, иностранные языки, следует вынести в отдельные таблицы;

– на одном проекте могут принимать участие несколько сотрудников, и таким же образом один сотрудник может быть занят несколькими проектами, что подразумевает связь «многие-ко-многим» и так называемую «развязочную» таблицу;

– в целях безопасности пароль пользователя следует подвергнуть хешированию, а не хранить в исходном виде.

Сущность «Пользователь» является ключевой в системе. Каждая строка в таблице содержит уникальный идентификатор, имя, фамилию, дату рождения, роль, должность, информацию о логине и пароле, соли, использованной для хеширования пароля, информацию о текущем статусе аккаунта пользователя. Описание атрибутов сущности «Пользователь» представлено в таблице 2.1.

Таблица 2.1 – Описание сущности «Пользователь»

|  |  |  |
| --- | --- | --- |
| № | Имя атрибута | Описание атрибута |
| 1 | Id | Уникальный идентификатор пользователя |
| 2 | Login | Логин пользователя |
| 3 | Password | Хешированный пароль пользователя |
| 4 | Salt | Соль, использованная для хеширования пароля |
| 5 | DefaultPassword | Первичный пароль пользователя |
| 6 | Name | Имя пользователя |
| 7 | Surname | Фамилия пользователя |
| 8 | Bithday | Дата рождения пользователя |
| 9 | IsInitial | Значение, указывающее статус аккаунта пользователя |
| 10 | IsActive | Значение, указывающее статус активности аккаунта |
| 11 | PositionId | Ссылка на должность пользователя |

Сущность «Навыки пользователя» содержит информацию о профессиональных навыках сотрудника, а так же уровнях владения данными навыками. Описание атрибутов сущности «Клиент» представлено в таблице 2.2.

Таблица 2.2 – Описание сущности «Навыки пользователя»

|  |  |  |
| --- | --- | --- |
| № | Имя атрибута | Описание атрибута |
| 1 | Id | Уникальный идентификатор записи в таблице |
| 2 | UserId | Ссылка на пользователя, владеющего навыком |
| 3 | SkillId | Ссылка на навык, которым владеет пользователь |
| 4 | SkillLevelId | Ссылка на уровень владения навыком |

Сущность «Проект» содержит информацию о проектах компании. Каждая строка таблицы содержит информацию об имени проекта, даты начала и окончании проекта, а так же описание данного проекта. Описание атрибутов сущности «Проект» представлено в таблице 2.3.

Таблица 2.3 – Описание сущности «Проект»

|  |  |  |
| --- | --- | --- |
| № | Имя атрибута | Описание атрибута |
| 1 | Id | Уникальный идентификатор записи в таблице |
| 2 | Name | Краткое название проекта |
| 3 | StartDate | Дата начала проекта |
| 4 | EndDate | Дата окончания проекта |
| 5 | Description | Текстовое описание проекта |

Сущность «Проекты пользователя» содержит информацию об участии пользователя на проектах компании. Каждая строка таблицы содержит ссылку на проект и позицию сотрудника на данном проекте. Описание атрибутов сущности «Проекты пользователя» представлено в таблице 2.4.

Таблица 2.4 – Описание сущности «Проекты пользователя»

|  |  |  |
| --- | --- | --- |
| № | Имя атрибута | Описание атрибута |
| 1 | Id | Уникальный идентификатор записи в таблице |
| 2 | UserId | Ссылка на пользователя |
| 3 | ProjectId | Ссылка на проект |
| 4 | PositionId | Ссылка на позицию пользователя на данном проекте |
| 5 | Comment | Комментарий пользователя о данном проекте |

Сущность «Образование» содержит информацию о местах получения высшего образования пользователем. Каждая строка таблицы содержит информацию о времени и месте получения образования. Описание атрибутов сущности «Образование» представлено в таблице 2.5.

Таблица 2.5 – Описание сущности «Образование»

|  |  |  |
| --- | --- | --- |
| № | Имя атрибута | Описание атрибута |
| 1 | Id | Уникальный идентификатор записи в таблице |
| 2 | UserId | Ссылка на пользователя |
| 3 | StartDate | Дата начала получения образования |
| 4 | EndDate | Дата окончания получения образования |
| 5 | University | Название университета |
| 6 | Faculty | Название факультета |
| 7 | Speciality | Название специальности |

Сущность «Сертификат» содержит информацию о сертификате, полученным пользователем и дате его получения. Описание атрибутов сущности «Сертификат» представлено в таблице 2.6.

Таблица 2.6 – Описание сущности «Сертификат»

|  |  |  |
| --- | --- | --- |
| № | Имя атрибута | Описание атрибута |
| 1 | Id | Уникальный идентификатор записи в таблице |
| 2 | UserId | Ссылка на пользователя |
| 3 | Date | Дата получения сертификата |
| 4 | Name | Название, краткое описание сертификата |

Сущность «Язык» содержит информацию об иностранных языках. Описание атрибутов сущности «Язык» представлено в таблице 2.7.

Таблица 2.7 – Описание сущности «Язык»

|  |  |  |
| --- | --- | --- |
| № | Имя атрибута | Описание атрибута |
| 1 | Id | Уникальный идентификатор записи в таблице |
| 2 | Name | Название иностранного языка |

Сущность «Языки пользователя» содержит информацию об иностранных языках, которыми владеет пользователь. Описание атрибутов сущности «Языки пользователя» представлено в таблице 2.8.

Таблица 2.8 – Описание сущности «Языки пользователя»

|  |  |  |
| --- | --- | --- |
| № | Имя атрибута | Описание атрибута |
| 1 | Id | Уникальный идентификатор записи в таблице |
| 2 | UserId | Ссылка на пользователя |
| 3 | LanguageId | Ссылка на иностранный язык |
| 4 | LanguageLevelId | Ссылка на уровень владения иностранным языком |

Сущность «Роль» содержит информацию о существующих ролях в приложении, которыми может владеть пользователь. Описание атрибутов сущности «Роль» представлено в таблице 2.9.

Таблица 2.9 – Описание сущности «Роль»

|  |  |  |
| --- | --- | --- |
| № | Имя атрибута | Описание атрибута |
| 1 | Id | Уникальный идентификатор записи в таблице |
| 2 | Name | Название роли |

Сущность «Роль пользователя» содержит информацию о ролях, которыми владеет пользователь. В данный момент в приложении существует возможность владения только одной ролью, но с учетом возможности владения пользователем несколькими ролями была создана отдельная таблица. Описание атрибутов сущности «Роль пользователя» представлено в таблице 2.10.

Таблица 2.10 – Описание сущности «Роль пользователя»

|  |  |  |
| --- | --- | --- |
| № | Имя атрибута | Описание атрибута |
| 1 | Id | Уникальный идентификатор записи в таблице |
| 2 | UserId | Ссылка на пользователя |
| 3 | RoleId | Ссылка на роль пользователя |

Сущность «Профессиональное описание» содержит краткую профессиональную информацию о пользователе. Описание атрибутов сущности «Профессиональное описание» представлено в таблице 2.11.

Таблица 2.11 – Описание сущности «Профессиональное описание»

|  |  |  |
| --- | --- | --- |
| № | Имя атрибута | Описание атрибута |
| 1 | Id | Уникальный идентификатор записи в таблице |
| 2 | UserId | Ссылка на пользователя |
| 3 | ProfessionalSummary | Краткое профессиональное описание пользователя |

Сущность «Уровень владения иностранным языком» содержит список уровней владения иностранными языками. Описание атрибутов сущности «Уровень владения иностранным языком» представлено в таблице 2.12.

Таблица 2.12 – Описание сущности «Уровень владения иностранным языком»

|  |  |  |
| --- | --- | --- |
| № | Имя атрибута | Описание атрибута |
| 1 | Id | Уникальный идентификатор записи в таблице |
| 2 | Name | Название уровня владения иностранным языком |

Сущность «Запрос» содержит информацию о поисковых запросах сохранённых пользователем, имеющим роль «Менеджер». Описание атрибутов сущности «Запрос» представлено в таблице 2.13.

Таблица 2.13 – Описание сущности «Запрос»

|  |  |  |
| --- | --- | --- |
| № | Имя атрибута | Описание атрибута |
| 1 | Id | Уникальный идентификатор записи в таблице |
| 2 | UserId | Ссылка на пользователя |
| 3 | QueryString | Строка содержащая информацию о поисковом запросе в спец. формате |
| 4 | Description | Название, краткое описание запроса |

Сущность «Профессиональный навык» содержит список профессиональных навыков, которыми может владеть пользователь. Описание атрибутов сущности «Профессиональный навык» представлено в таблице 2.14.

Таблица 2.14 – Описание сущности «Профессиональный навык»

|  |  |  |
| --- | --- | --- |
| № | Имя атрибута | Описание атрибута |
| 1 | Id | Уникальный идентификатор записи в таблице |
| 2 | Name | Имя профессионального навыка |

Сущность «Уровень владения профессиональным навыком» содержит список уровней владения профессиональными навыками. Описание атрибутов сущности «Уровень владения профессиональным навыком» представлено в таблице 2.15.

Таблица 2.15 – Описание сущности «Уровень владения иностранным языком»

|  |  |  |
| --- | --- | --- |
| № | Имя атрибута | Описание атрибута |
| 1 | Id | Уникальный идентификатор записи в таблице |
| 2 | Name | Название уровня владения профессиональным навыком |

Сущность «Должность» содержит список должностей сотрудников. Описание атрибутов сущности «Должность» представлено в таблице 2.16.

Таблица 2.16 – Описание сущности «Уровень владения иностранным языком»

|  |  |  |
| --- | --- | --- |
| № | Имя атрибута | Описание атрибута |
| 1 | Id | Уникальный идентификатор записи в таблице |
| 2 | Name | Название должности |

После описания атрибутов определяются связи между сущностями. В зависимости от условий, предлагаемых предметной областью, связи между объектами могут быть следующими:

– один к одному;

– один ко многим;

– многие ко многим.

Например, сущности «Пользователь» и «Проект» связаны отношением «многие-ко-многим», так как один сотрудник может иметь несколько проектов в системе, в то же время в каждом из проектов могут быть задействованы несколько сотрудников.

Графическое представление спроектированной модели данных, где отображены названия полей сущностей и типы этих полей, а также отмечены поля, являющиеся первичными либо внешними ключами, представлено в ДП 01.00.ГЧ.

### 2.4 Использованные для разработки технологии

Данный программный продукт разработан в рамках программной платформы Microsoft .NET Framework и совместимых с ней свободно распространяемых библиотек. По требованию заказчика основным языком программирования выбран объектно–ориентированный язык общего назначения C#[4].

В приложении использованы:

* программная платформа Windows Presentation Foundation;
* библиотека для создания настольных приложений Prism*;*
* контейнер инверсии зависимостей Unity;
* библиотека графического интерфейса Material Design Toolkit XAML;
* AutoMapper;
* библиотека PropertyChanged.Fod*y.*

Windows Presentation Framework (WPF) – это мощная среда для создания приложений Windows. WPF является полноценной частью платформы .NET, состоящая по большей части из управляемого кода и использующая средства устаревшей библиотеки User32 минимальным образом [5].

В основе WPF лежит векторная система визуализации, не зависящая от разрешения устройства вывода и созданная с учётом возможностей современного графического оборудования. WPF предоставляет средства для создания визуального интерфейса, включая язык разметки XAML, элементы управления, привязку данных, макеты, двухмерную и трёхмерную графику, анимацию, стили, шаблоны, документы, текст и мультимедиа. Графической технологией, лежащей в основе WPF, является DirectX, в отличие от Windows Forms, где используется GDI/GDI+, поэтому производительность WPF выше, чем у GDI+, за счёт использования аппаратного ускорения графики через DirectX.

Prism – это библиотека для создания слабосвязанных, поддерживаемых и тестируемых приложений XAML в WPF, Universal Windows Platform и Xamarin Forms. Prism предоставляет реализацию набора шаблонов проектирования, которые полезны при написании хорошо структурированных и поддерживаемых приложений, включая Model–View–ViewModel, внедрение зависимостей, команды и другие. Основная функциональность Prism содержится в портативной библиотеке классов, ориентированной на все перечисленные платформы. Расширения для каждой конкретной платформы предоставляются отдельными пакетами для поддержания модульности [6].

Material Design Toolkit XAML – библиотека с открытым исходным кодом, которая содержит набор стилей и элементов управления для на языке разметки XAML, разработанных в соответствии с набором рекомендаций Material Design от Google [7].

Material Design основан на четырех основных принципах:

– тактильные поверхности;

– полиграфический дизайн;

– осмысленная анимация;

– адаптивный дизайн.

В качестве примеров приложений, выполненных с учетом рекомендаций Google Material Design, можно отметить следующие приложения:

– Google Play;

– Google+;

– Google Chrome;

– Google Calendar;

– Gmail.

Для взаимодействия с базой данных используется:

– объектно-ориентированная технология доступа к данным Entity Framework;

– система управления базами данных Microsoft SQL Server.

Entity Framework (EF) представляет собой специальную объектно-ориентированную технологию на базе платформы .NET для работы с данными. Если традиционные средства ADO.NET позволяют создавать подключения, команды и прочие объекты для взаимодействия с базами данных, то EF предоставляет более высокий уровень абстракции, позволяющий устраняться от работы с самой базой данных и работать с данными независимо от типа хранилища. Если на физическом уровне мы оперируем таблицами, индексами, первичными и внешними ключами, но на концептуальном уровне, предлагаемом EF, мы работаем исключительно с объектами [8].

Microsoft SQL Server – система управления реляционными базами данных, входящая в состав платформы .NET. Основным используемым языком запросов является Transact-SQL, реализованный в соответствии со стандартом ANSI/ISO по структурированному языку запросов SQL с расширениями. Данная система управления базами данных используется для работы с базами данных размером от персональных до баз масштаба предприятия и успешно конкурирует с другими решениями в этом сегменте рынка [9].

Automapper позволяет проецировать одну модель на другую, что позволяет сократить объемы кода и упростить программу [10]. Данная библиотека может вам понадобиться в следующих случаях:

– преобразование иерархической модели объектов в плоскую;

– преобразование иерархической модели объектов в плоскую;

– получение проекций объектов (из доменного объекта генерируем View- Model);

– получение доменного объекта из DTO;

– преобразование доменных объектов в сериализуемые объекты для передачи через какую-либо коммуникационную среду.

PropertyChanged.Fody – это плагин, который добавляет оповещение (вызов) PropertyChanged события для всех сеттеров свойств объектов, унаследованных от INotifyPropertyChanged или помеченных атрибутов. Данный плагин для каждого отдельного публичного свойства класса, который реализует интерфейс INotifyPropertyChanged автоматически генерирует приватное свойство, таким образом сокращая объем написанного разработчиком кода.

### 2.5 Выводы по разделу

В данном разделе были рассмотрены основные задачи разрабатываемого программного средства. Так же были представлены основные роли в приложении, а для роли «Администратор» была рассмотрена диаграмма вариантов использования.

Так же была представлена структура программного средства и отношения между его структурными блоками.

В данном разделе была рассмотрена структура каждой таблицы базы данных, использующейся в приложении. Каждому полю таблицы было дано описание.

Так же были описаны используемые технологии, задействованные в программном средстве.

### 3 Разработка программного средства

Утверд.

Н. контр.

Провер.

Разраб.

*Пацей Н.В.*

*Ковалевская Н.И*.

*Русакович А.Д.*

*Наркевич А.С.*

Ф.И.О.

Подпись

Дата

14

*Разработка*

*программного средства*

Лист

1

*ДП 03.00.ПЗ*

Лит.

Листов

*БГТУ 74417033, 2019*

### 3.1 Структура программного средства

Программное средство разработано на основе архитектурного шаблона Model-View-ViewModel с четким разделением обязанностей между слоями [11]. При реализации приложения была использована интегрированная среда разработки Microsoft Visual Studio 2017 Community. Решение состоит из одного проекта содержащих 13 папок. Далее подключим к проекту все необходимые вспомогательные инструменты и библиотеки с помощью менеджера пакетов NuGet. В соответствии с требованиями по реализации все эти инструменты являются бесплатными. Список подключенных инструментов и библиотек для проекта представлен в таблице 3.1.

Таблица 3.1 – Список подключенных библиотек

|  |  |
| --- | --- |
| Подключенные библиотеки | Описание и использование |
| Entity Framework | Библиотека объектно-ориентированного доступа к данным, в проекте используется для доступа к базе данных и построения слоя хранилища данных |
| Prism.Core | Библиотека основных классов платформы для построения настольных приложений Prism, в проекте служит для построения базовых классов сущностей |
| Unity | Контейнер инверсии зависимостей, в проекте служит для динамического разрешения зависимостей между интерфейсами и их реализациями |
| Prism.Core | Библиотека основных классов реализующей шаблон Model-View*-*ViewModel платформы Prism, необходима для функционирования остальных модулей Prism |
| Prism.Wpf | Модуль Prism, предоставляющий полный функционал Prism для приложений Windows Presentation Foundation |
| Prism. Unity | Библиотека расширений Prism, служащая для поддержки интеграции с контейнером Unity |
| Material Design XAML | Библиотека, предназначенная для построения графического интерфейса на языке разметки XAML в соответствии с рекомендациями Google Material Design и содержащая набор готовых настраиваемых элементов управления |
| AutoMapper | Библиотека, предназначенная для маппинга свойств одной сущности в другую по определенным пользователем правилам |

Окончание таблицы 3.1

|  |  |
| --- | --- |
| Подключенные библиотеки | Описание и использование |
| PropertyChanged.Fody | Библиотека, которая добавляет к каждому свойству класса, реализующего интерфейс INotifyPropertyChanged*,* нотификацию представления при изменении значения данного свойства. |

Для удобства реализации и сопровождения файлы проектов распределялись по папкам в зависимости от типа и назначения конкретного файла. Организация групп классов в папки позволяет избежать конфликтов между именами классов, ведь принадлежность к пакету позволяет гарантировать однозначность имен [12]. Таким образом, файлы экранов должны находиться в папке Views, а файлы их моделей – в папке ViewModels. Структура проекта описана в таблице 3.2.

Таблица 3.2 – Структура проекта

|  |  |
| --- | --- |
| Название папки | Описание папки |
| AttachedProperties | Содержит файлы классов прикрепляемых свойств, которые позволяют добавлять дополнительное описание или данные к элементам управления |
| Behaviors | Содержит классы поведений, реализующие интерфейс Behavior<T>, которые позволяют добавить некоторое дополнительное поведение элементу управления |
| Converters | Содержит все необходимые конвертеры для взаимодействия представлений с моделями представлений |
| Database | Содержит классы сущностей базы данных, а так же dbcontext с помощью которого осуществляется доступ к базе данных |
| Database.Scripts | Содержит необходимые скрипты для создания базы данных, заполнения необходимой первоначальной информацией |
| Events | Содержит классы событий, использующихся классом EventAggregator для взаимодействия между слабосвязанными компонентами |
| Helpers | Содержит классы, содержащие вспомогательную функциональность, такую как реализацию навигации, класс-хелпер для хеширования пароля, класс констант и т.д. |
| Images | Содержит графические файлы, использующиеся в приложении |
| Libraries | Содержит dll файлы сторонней библиотеки Syncfusion |
| Styles | Содержит файлы стилей приложения |
| Models | Содержит классы сущностей базы данных, использующихся в бизнес логике приложения |
| ViewModels | Классы моделей представлений, необходимые для управления графическим интерфейсом приложения из кода |
| Views | Содержит файлы разметки на языке XAML, описывающие экраны графического интерфейса положения |

Организация групп классов в папке так же является необходимостью при использовании библиотеки Prism*,* так как DataContext страницы приложения устанавливается в соответствии с соглашением о наименовании.

### 3.3 Разработка программного кода

В первую очередь на основе разработанной модели данных была создана база данных. База данных построена в среде Microsoft SQL Server с использованием языка запросов Transact-SQL. Опишем создание нескольких ключевых таблиц. Для удобства манипуляцией объектами базы данных использовалась SQL Server Management Studio.

Таблица «Пользователь» является ключевой и содержит основную информацию об аккаунте пользователя в системе и, как следствие, содержит большое количество ссылок на другие таблицы, заданных внешними ключами. Код создания таблицы «Пользователь» на языке Transact-SQLпредставлен в листинге 3.1.

|  |
| --- |
| CREATE TABLE [dbo].[User](  [Id] [int] IDENTITY(1,1) NOT NULL,  [Name] [nvarchar](50) NOT NULL,  [Surname] [nvarchar](50) NOT NULL,  [Birthday] [date] NOT NULL,  [PositionId] [int] NOT NULL,  [Login] [nvarchar](20) NOT NULL,  [Password] [nvarchar](256) NOT NULL,  [Salt] [nvarchar](56) NOT NULL,  [IsInitial] [bit] NOT NULL,  [DefaultPassword] [nvarchar](32) NULL,  [IsActive] [bit] NOT NULL,  CONSTRAINT [PK\_User] PRIMARY KEY CLUSTERED  ([Id] ASC)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]) ON [PRIMARY] |

Листинг 3.1 – Создание таблицы «Пользователь»

Часть кода добавления внешних ключей к таблице «Пользователь» представлена в листинге 3.2.

|  |
| --- |
| ALTER TABLE [dbo].[Education] WITH CHECK ADD CONSTRAINT [FK\_Education\_User] FOREIGN KEY([UserId])REFERENCES [dbo].[User] ([Id])  ALTER TABLE [dbo].[Language] WITH CHECK ADD CONSTRAINT [FK\_Language\_User] FOREIGN KEY([UserId])REFERENCES [dbo].[User] ([Id])  ALTER TABLE [dbo].[ProfessionalSummary] WITH CHECK ADD CONSTRAINT [FK\_ProfessionalSummary\_User] FOREIGN KEY([UserId])  REFERENCES [dbo].[User] ([Id])  ALTER TABLE [dbo].[Query] WITH CHECK ADD CONSTRAINT [FK\_Query\_User] FOREIGN KEY([UserId])REFERENCES [dbo].[User] ([Id])  ALTER TABLE [dbo].[Sertificate] WITH CHECK ADD CONSTRAINT [FK\_Sertificate\_User] FOREIGN KEY([UserId])REFERENCES [dbo].[User] ([Id]) |

Листинг 3.2 – Добавление внешних ключей к таблице «Пользователь»

Таблица «Навыки сотрудника» также играет важную роль в системе, определяет технические навыки сотрудников и позволяет производить поиск по ним. Код создания таблицы «Навыки сотрудника» на языке Transact–SQL представлен в листинге 3.3.

|  |
| --- |
| CREATE TABLE [dbo].[UserSkills](  [Id] [int] IDENTITY(1,1) NOT NULL,  [UserId] [int] NOT NULL,  [SkillId] [int] NOT NULL,  [SkillLevelId] [int] NOT NULL,  CONSTRAINT [PK\_UserSkills] PRIMARY KEY CLUSTERED  ([Id] ASC)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]) ON [PRIMARY] |

Листинг 3.3 – Создание таблицы «Навыки сотрудника»

Таблица «Проекты пользователей» несет мало смысла в контексте предметной области, но технически является связующим звеном между пользователями и проектами, реализуя связь «многие–ко–многим». Причина ее существования лишь в том, что язык запросов SQL не реализует вышеупомянутой связи «многие–ко–многим». Код создания таблицы «Проекты пользователей» на языке Transact–SQLпредставлен в листинге 3.4.

|  |
| --- |
| CREATE TABLE [dbo].[UserProjects](  [Id] [int] IDENTITY(1,1) NOT NULL,  [ProjectId] [int] NOT NULL,  [UserId] [int] NOT NULL,  [PositionId] [int] NOT NULL,  [Comment] [nvarchar](200) NULL,  CONSTRAINT [PK\_UserProjects] PRIMARY KEY CLUSTERED  ([Id] ASC)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]) ON [PRIMARY] |

Листинг 3.4 – Создание таблицы «Проекты пользователей»

В качестве типичного примера таблицы-перечисления, служащей для возможности избежать дублирования данных, приведем таблицу «Роли». Код создания данной таблицы на языке Transact-SQLпредставлен в листинге 3.5.

|  |
| --- |
| CREATE TABLE [dbo].[Role](  [Id] [int] IDENTITY(1,1) NOT NULL,  [Name] [nvarchar](50) NOT NULL,  CONSTRAINT [PK\_Role] PRIMARY KEY CLUSTERED  ([Id] ASC)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]) ON [PRIMARY] |

Листинг 3.5 – Создание таблицы «Роли»

После построения оставшихся таблиц базы данных перейдем к реализации классов осуществляющих логику навигации в приложении. Изначально был создан интерфейс содержащий поведение навигации в приложении. Код данного интерфейса приведен в листинге 3.6.

|  |
| --- |
| public interface INavigationService  {  void NavigateTo(string viewName);  void NavigateToWithParameters(  string viewName,  string regionName = null,  NavigationParameters navigationParameters = null);  } |

Листинг 3.6 – Создание таблицы «Роли»

Данный интерфейс реализует класс NavigationService, код которого приведен в листинге 3.7

|  |
| --- |
| public class NavigationService : INavigationService  {  private readonly IRegionManager \_regionManager;  public NavigationService(IRegionManager regionManager)  {  regionManager = regionManager;  }  public void NavigateTo(string viewName)  {  \_regionManager.NavigateTo(viewName);  }  public void NavigateToWithParameters(  string viewName,  string regionName = null,  NavigationParameters navigationParameters = null)  {  \_regionManager.NavigateToWithParameters(viewName,  regionName, navigationParameters);  }  } |

Листинг 3.7 – Код класса NavigationService

В данном классе переменная \_regionManager является реализацией интерфейса IRegionManager, который использует класс NavigationHelper, содержащий методы расширения.

Для создания метода расширения необходимо создать статический класс, содержащий статические методы, первым параметром которого будет являться ссылка на тип объекта к которому данное расширение будет применяться. Код класса NavigationHelper приведен в листинге 3.8.

|  |
| --- |
| public static class NavigationHelper  {  public static void NavigateTo(this IRegionManager  regionManager, string pageName)  {  NavigateToWithParameters(regionManager, pageName);  }  public static void NavigateToWithParameters(  this IRegionManager regionManager, string pageName,  string regionName = null,  NavigationParameters navigationParameters = null)  {  if (regionManager == null || pageName == null)  return;  regionManager.RequestNavigate(regionName ??  Constants.MainRegion, pageName, navigationParameters);  }  } |

Листинг 3.8 – Код класса NavigationHelper

Вернемся к коду проекта, отвечающего за графический интерфейс приложения. Для управления процессом запуска приложения, а именно стартового проекта UI, создан класс Bootstrapper. Часть кода данного класса представлен в листинге 3.9. Полный код данного класса вынесен в приложение А.

|  |
| --- |
| public class Bootstrapper : UnityBootstrapper  {  protected override void InitializeShell()  {  base.InitializeShell();  App.Current.MainWindow = (Window)Shell;  App.Current.MainWindow.Show();  }  protected override void ConfigureContainer()  {  base.ConfigureContainer();  Container.RegisterTypeForNavigation<LoginView>  (Constants.LoginView);  Container.RegisterTypeForNavigation<RegisterView>  (Constants.RegisterView); //…  }  } |

Листинг 3.9 – Код класса Bootstrapper

Данный класс наследуется от класса UnityBootstrapper и выполняет первоначальную настройку приложения перед запуском: разрешает зависимости классов, инициализирует начальный экран приложения, настраивает механизм навигации между экранами. Переопределенные и вспомогательные методы вызываются при настройке конфигурации ядра контейнера Unity на этапе запуска приложения.

Метод InitializeShell() предназначен для инициализации представления, которое должно появляться при запуске приложение, то есть его начальный экран. Метод ConfigureContainer() служит для регистрации экранов приложения в системе навигации, а так же регистрации реализаций интерфейсов использующихся для инъекции в конструкторы классов приложения.

После этого перейдем к реализации экранов приложения. С  
технической точки зрения, в соответствии с паттерном Model-View-ViewModel каждый экран приложения состоит из двух частей: представления  
и модели представления. Представление задает расположение элементов управления на экране и представляет собой XAML-разметку, тогда как  
модель представления управляет этим экраном и представляет собой класс на языке C#. Библиотека Prism поддерживает соглашение наименований, по которому представление с некоторым названием и окончанием View и класс с аналогичным названием и окончанием ViewModel автоматически связываются друг с другом.

Для удобства был создан базовый класс модели представления BaseViewModel, содержащий ссылки на все необходимые интерфейсы, в том числе интерфейс IRegionManager, поддерживающий навигацию между экранами, и интерфейс IEventAggregator, предоставляющий глобальную систему событий приложения. Часть кода данного класса приведен приведена в листинге 3.10. Полный код класса BaseViewModelвынесен в приложение Б.

|  |
| --- |
| public class BaseViewModel : INotifyPropertyChanged, IDataErrorInfo  {  protected readonly IRegionManager regionManager;  protected readonly IEventAggregator eventAggregator;  public BaseViewModel (IRegionManager regionManager,  IEventAggregator eventAggregator, IRepository repo)  this.regionManager = regionManager;  this.eventAggregator = eventAggregator;  }  //…  } |

Листинг 3.10 – Часть кода класса BaseViewModel

В качестве простого примера опишем код реализации корневого экрана, который является контейнером для всех остальных. Данный код позволяет определить регионы в приложении, в которые в дальнейшем будут подставляться соответствующие данному региону страницы. Библиотека Prism позволяет определять любое количество регионов, а так же предоставляет класс для управления данными регионами.

Фрагмент кода представления на языке разметки XAML представлен в листинге 3.11. Полный код модели представления вынесен в приложение В.

|  |
| --- |
| <Controls:MetroWindow x:Class="ITManager.Views.ShellView"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"  WindowStartupLocation="CenterScreen"  WindowState="{Binding WindowState}"  Title="ITManager" Height="600" Width="800">  <Grid>  <Grid.ColumnDefinitions>  <ColumnDefinition Width="Auto" MaxWidth="250"/>  <ColumnDefinition Width="\*"/>  </Grid.ColumnDefinitions>  <ContentControl Grid.Column="0" Visibility="{BindingMenuVisibility, UpdateSourceTrigger=PropertyChanged, Mode=TwoWay}"  prism:RegionManager.RegionName="{x:Static c:Constants.MenuRegion}"/>  <ContentControl Grid.Column="1"  prism:RegionManager. RegionName= "{x:Static c:Constants.MainRegion}"/>  </Grid>  </Controls:MetroWindow> |

Листинг 3.11 – Код представления ShellView

Пример разделения приложения на регионы представлен на рисунке 3.1.

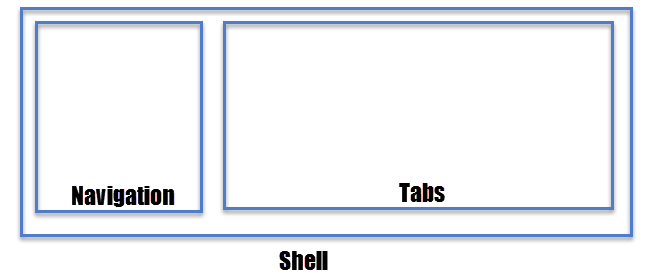


Рисунок 3.1 – Разделение приложения на регионы

Остальные экраны создаются схожим образом, поэтому описывать их реализацию не имеет смысла. В качестве типовой модели представления приведем модель представления, созданную для экрана личной страницы, содержащую профессиональную информацию сотрудника, поддерживающую добавление, удаление, редактирование информации содержащийся в секциях данной страницы. Данная страница так же поддерживает валидацию редактируемых секций. Так же в приложении существует возможность просмотра страниц профессиональной информации других сотрудников, поэтому на данной странице происходит проверка роли пользователя, а так же принадлежность текущего пользователя к открытой странице. Если пользователь проходит данную проверку, то у него появляется возможность редактировать секции на странице.

Код данной модели представления вынесен в приложение Г. Данная страница реализована с использованием контейнера компоновки Grid на языке XAML. Данный элемент управления позволяет определить для каждой секции, содержащую профессиональную информацию сотрудника, свою собственную строку. Каждой отдельной строки можно задать свойства такие как высота, максимальная высота, минимальная высота и т. д. Каждая строка содержит в себе элемент управления StackPanel, позволяющий расположить элементы один под другим в случае вертикальной ориентации, и один за другим в случае горизонтальной.

В процессе реализации некоторых экранов программного средства привязать свойства элемента управления представления напрямую к модели представления не являлось возможным. Механизм привязки позволяет вмешиваться в процесс передачи значения из представления в модель с помощью определения конвертера значений. Реализовать конвертер значений можно наследуясь от интерфейса IValueConverter. В нашем случае потребовалось реализовать конвертер для свойства, управляющего видимостью элемента, чтобы привязать его к свойству булева типа модели представления. Код конвертера BoolToVisibilityConverter представлен в листинге 3.12.

|  |
| --- |
| public class BoolToVisibilityConverter : IValueConverter  {  public object Convert(object value,  Type targetType,  object parameter,  System.Globalization.CultureInfo culture)  {  if (value is bool boolean && boolean)  return Visibility.Visible;  return Visibility.Collapsed;  }  public object ConvertBack(object value,  Type targetType,  object parameter,  System.Globalization.CultureInfo culture)  {  if (value is Visibility visibility && visibility ==  Visibility.Visible)  return true;  return false;  }  } |

Листинг 3.12 – Код класса BoolToVisibilityConverter

Так же немаловажное значение имеет использование DataTrigger на персональной странице профессиональной информации сотрудника. Данный триггер позволяет изменять свойства исходного элемента управления в зависимости от некого свойства, к которому мы привязываемся. Таким образом, с использованием данного триггера мы реализовываем логику редактирования секций. Пример части кода данного триггера представлен в листинге 3.13.

|  |
| --- |
| <DataTrigger Binding="{Binding IsChecked, ElementName=SkillsEditToggleButton, UpdateSourceTrigger=PropertyChanged}" Value="False">  <Setter Property="ItemTemplate">  //..  </Setter>  </DataTrigger> |

Листинг 3.13 – Код использования DataTrigger

Таким образом, мы привязываемся к свойству IsChecked элемента с именем SkillsEditToggleButtonи если значение данного свойства будет равно False, то мы присвоим другое значение свойству ItemTemplate исходного элемента управления. Поэтому при нажатии на элемент управления ToggleButtonсоответствующей секции будет изменяться контент данной секции. Пример кода элемента управления ToggleButtonсекции Skillsпредставлен в листинге 3.14.

|  |
| --- |
| <ToggleButton  Content="✎"  IsChecked="{Binding IsSkillsChecked, Mode=TwoWay}"  Visibility="{Binding CanUserChangeData, Converter={StaticResource VisibleIfTrueConverter}}"  Style="{StaticResource MetroCircleToggleButtonStyle}"  Width="40"  Height="40"/> |

Листинг 3.14 – Код использования ToggleButton

У данного элемента управления ToggleButtonсвойство Visibilityпривязано к свойству CanUserChangeData, которое определяет будет ли отображаться данный элемент управления или нет. Значение данного свойства зависит от того хватает ли у вас прав для изменения информации на данной страницы или нет. Есть прав у вас недостаточно, то данный элемент управления не будет отображаться, в противном случае – данный элемент управления будет присутствовать у каждой секции.

В левой части содержится панель навигации, которая позволяет перемещаться по страницам в приложении, а в правой части находится страница контента, содержащую в себе текущую выбранную страницу.

При нажатии на кнопку на панели навигации приложение происходит выполнение прикрепленной команды NavigateToкоторая выполняет смену страниц в приложении. Данная команда является переменной типа Command, которой присваивается делегат типа DelegateCommand, определенный в используемой библиотеке Prism. Параметром делегата является метод, выполняющий навигацию в приложении. Код реализации данного метода представлен в листинге 3.15.

|  |
| --- |
| private void NavigateToView(string viewName)  {  if (viewName == Constants.LoginView)    \_eventAggregator.GetEvent<CloseMenuEvent>().Publish(true);  var navigationParameters = new NavigationParameters();  navigationParameters.Add("IsLogout", true);  if(viewName == Constants.SearchView && ShellViewModel.CurrentUser.UserRoles.Any(r => r.RoleId == Constants.UserRole))  {  \_navigationService.NavigateToWithParameters(Constants.DefaultSearchView, navigationParameters:NavigationParameters);  return;  }  \_navigationService.NavigateToWithParameters(viewName,  navigationParameters: navigationParameters)  } |

Листинг 3.15 – Код реализации команды NavigateTo

Так же практически любое приложение не может обойтись без валидации. В данном программном средстве для валидации использовался интерфейс IDataErrorInfo, который позволяет прописать правила для каждого свойства и выполнять валидацию каждого свойства в соответствии с ними.

Для того что бы использовать валидацию в приложении необходимо наследоваться от интерфейса IDataErrorInfo и реализовать его. Пример части кода реализации метода валидации свойств страницы управления аккаунтами пользователей представлен в листинге 3.16.

|  |
| --- |
| public override string Validate(string propertyName)  {  switch (propertyName)  {  case nameof(Login):  if(Login.IsNullOrWhiteSpace())  return string.Format(Constants.FieldMustBeFilledMessageFormat,  nameof(Login));  else if(!Login.IsLengthBetween(3, 20))  return string.Format(Constants.LengthErrorMessageFormat,  nameof(Login), 3, 20);  break;  }  } |

Листинг 3.16 – Код реализации интерфейса IDataErrorInfo

Для остальных страниц валидация осуществляется аналогичным образом. Лишь имена свойств и правила валидации могут отличаться, поэтому рассматривать их не имеет смысла.

Немаловажным является применение стилей в приложении. Все пользовательские стили в приложении, которые использовались в разных местах приложения были вынесены в отдельный файл ресурсов LightStyles.xaml. Для того что бы мы могли в любом месте использовать данные стили мы должны подключить данный файл ресурсов к нашему приложению в файле App.xaml. Пример части кода файла App.xamlпредставлен в листинге 3.17.

|  |
| --- |
| <Application.Resources>  <ResourceDictionary>  <ResourceDictionary.MergedDictionaries>  //…  <ResourceDictionary Source="Styles/LightStyles.xaml"/>  </ResourceDictionary.MergedDictionaries>  </ResourceDictionary>  </Application.Resources> |

Листинг 3.17 – Подключение словаря ресурсов

Важным при написании приложения является создания класса или классов констант, которые будут содержать использующиеся много раз константные значения. При написании кода очень легко совершить грамматическую или иную ошибку, поэтому для удобства данные значения записывают в свойства и выносят в отдельный класс. Пример части кода класса констант представлен в листинге 3.18.

|  |
| --- |
| public static class Constants  {  // …  public const string DateMustBeCorrectMessage = "Date must be  correct.";  public const string FieldMustBeFilledMessageFormat = "'{0}'  field is required.";  public const string LengthErrorMessageFormat=  "The length of '{0}' must be from {1} to {2}.";  public const string FieldMustBeLessThan= "Field '{0}' must be  less than {1}.";  public const string FieldMustBeGreaterThan=  "Field '{0}' must be greater than {1}.";  } |

Листинг 3.18 – Код класса констант

При написании программного средства так же использовались классы поведений. Поведения служат для инкапсуляции частей функциональности пользовательского интерфейса, чтобы их можно было применять к элементам без написания соответствующего кода вручную. По-другому поведение можно рассматривать как службу, предлагаемую для элементов. Эта служба обычно предусматривает прослушивание нескольких различных событий и выполнение множества связанных операций. Для создания такого класса необходимо наследоваться от класса типа Behavior<T>, где T – тип объекта к которому будет применяться данное поведение. В данном приложении был создан класс поведений PasswordBehavior наследованный от типа PasswordBehavior <PasswordBox>. Данное поведение было создано для возможности привязать свойство к контенту элемента управления PasswordBox. Пример части кода класса поведения представлен в листинге 3.19.

|  |
| --- |
| public class PasswordBehavior : Behavior<PasswordBox>  {  public static readonly DependencyProperty PasswordProperty =  DependencyProperty.Register("Password",typeof(string),typeof(PasswordBehavior),new PropertyMetadata(default(string)));  public string Password  {  get { return (string)GetValue(PasswordProperty); }  set { SetValue(PasswordProperty, value); }  }  protected override void OnAttached()  {  AssociatedObject.PasswordChanged +=  PasswordBox\_PasswordChanged;  }  protected override void OnDetaching()  {  AssociatedObject.PasswordChanged –=  PasswordBox\_PasswordChanged;  }  } |

Листинг 3.19 – Код класса поведения

Пример использования созданного класса поведения для элемента управления PasswordBox представлен в листинге 3.20.

|  |
| --- |
| <PasswordBox Controls:TextBoxHelper.Watermark="Password\*"  MaxLength="32" VerticalContentAlignment="Center"  Controls:TextBoxHelper.ClearTextButton="True">  <i:Interaction.Behaviors>  <Behaviors:PasswordBehavior Password="{Binding Password,  Mode=TwoWay, ValidatesOnDataErrors=True}" />  </i:Interaction.Behaviors>  </PasswordBox> |

Листинг 3.20 – Пример использования созданного поведения

Данные поведения можно создать для абсолютно любого элемента управления, тем самым расширить его функционал.

### 3.4 Выводы по разделу

В данном разделе была предоставлена и описана структура проекта. Так же были рассмотрены и описаны подключенные библиотеки.

Так же был рассмотрен код создания основных таблиц базы данных. Был рассмотрен код наиболее важных классов программного средства, а так же было описано их назначение.

В данном разделе так же приводился код создания сервиса навигации, использующегося в приложении. Была рассмотрена части кода создания класса, управляющего запуском приложения. Так же был рассмотрен код базового класса модели представления.

Была предоставлена часть кода создания представления корневого экрана, а так же рассмотрен рисунок, показывающий разбиение на регионы с помощью данного кода. Так же были рассмотрены части кода создания класса конвертера и триггера данных, код создания toggle button и метода валидации. Диаграмма классов представлена в ДП 02.00.ГЧ.

### 4 Тестирование программного средства

Утверд.

Н. контр.

Провер.

Разраб.

*Пацей Н.В.*

*Ковалевская Н.И*.

*Русакович А.Д.*

*Наркевич А.С.*

Ф.И.О.

Подпись

Дата

7

*Тестирование*

*программного средства*

Лист

1

*ДП 04.00.ПЗ*

Лит.

Листов

*БГТУ 74417033, 2019*

### 4.1 Негативное тестирование страниц входа и регистрации

Негативным называют тестирование, в рамках которого применяются сценарии, которые соответствуют внештатному поведению тестируемой системы. Это могут быть, например, исключительные ситуации или неверные данные. Работа нашего приложения начинается с экрана входа в приложение, поэтому начнем тестирование с данного этапа. На экране входа в приложение находятся поля ввода логина и пароля, кнопка входа в приложение и ссылка на страницу регистрации. Изначально попытаемся оставить пустой форму логина и попробуем войти в приложение. Результат выполнений данного теста можно увидеть на рисунке 4.1.

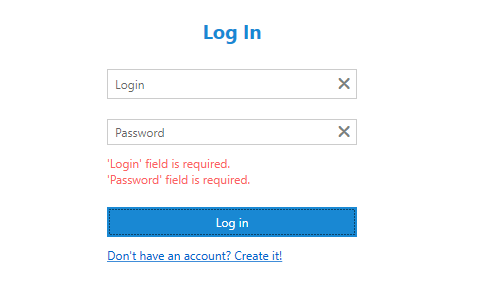


Рисунок 4.1 – Пример работы валидации формы входа

Приложение отработало корректно и отобразило сообщения правил валидации о необходимости заполнения обязательных полей. Для данной формы так же имеются правила валидации на длину поля и корректности логина или пароля.

После тестирования экрана входа перейдем на экран регистрации, предоставленный на рисунке 4.2. На данной форме регистрации существуют следующие поля для заполнения: логин, имя, фамилия, дата рождения, должность, пароль и подтверждение пароля. Так же на данной форме есть кнопка, по нажатию на которую происходит регистрация нового пользователя, и ссылка на форму входа в приложение с помощью существующего аккаунта. Поля, которые являются обязательными, помечены знаком звездочки. Введем невалидные данные в форму регистрации и попробуем зарегистрировать нового пользователя. Результат данного теста можно увидеть на рисунке 4.2.

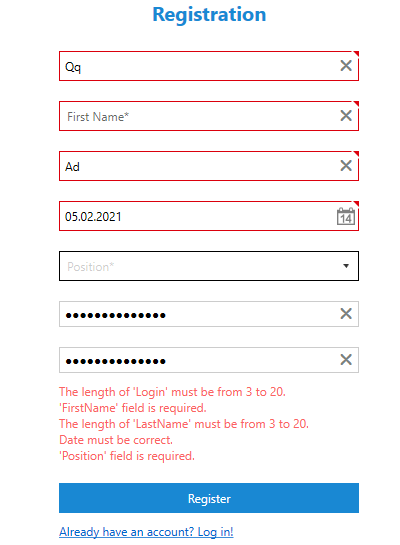


Рисунок 4.2 – Результат валидации формы регистрации

Данная форма так же не позволила ввести пользователю некорректные данные и отобразила сообщения правил валидации.

На данной форме применяется большее количество правил валидации в отличие от формы входа в приложения вследствие увеличения количества полей ввода и их разнообразия. На данной форме применяются как правила проверки заполнения полей, длины полей, так и проверки корректности введенного контента. На странице управления учетными записями пользователей при текущей роли «Администратор», в секции регистрации нового пользователя содержатся аналогичные поля и применяются аналогичные правила валидации, поэтому тестирование данной страницы не имеет смысла.

### 4.2 Негативное тестирования страницы профессиональной информации

Перейдем к тестированию страницы заполнения профессиональной информации. Секция Professional Summaryпозволяет ввести текстовую информацию с кратким описанием профессиональной деятельности. Сам элемент управления, использующийся для ввода данной информации, накладывает ограничения на длину введенного текста, поэтому ввести текстовое сообщение невалидной длинны является невозможным. Пример тестирования данной секции представлен на рисунке 4.3.

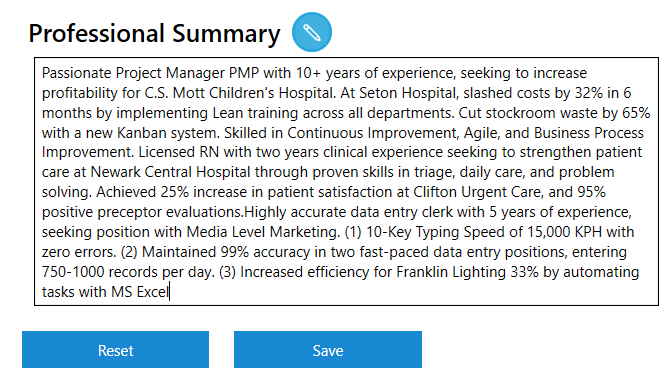


Рисунок 4.3 – Пример тестирования секции Professional Summary

Далее перейдем к тестированию секции Skills. Данная секция позволяет добавлять или удалять навыки пользователя с последующим их сохранением в базу данных. Добавленный навык требует обязательного заполнения двух полей: самого навыка и уровня владения данным навыком. В данной секции нет возможности выбрать невалидное значение из списка, содержащегося в элементе управления ComboBox, в связи с программной реализацией данного контрола. Таким образом, использование данного элемента управления позволяет избежать части ошибок при проверке данных на корректность. Кнопка Resetпозволяет сбросить текущие изменения касающиеся данной секции, а кнопка Saveнаоборот – сохраняет все изменения при условии их валидности в базу данных. Кнопка, содержащая в себе знак плюса, позволяет добавить новый элемент в данную секцию. Так же данная секция имеет возможность удаления существующих навыков путем нажатия на кнопку удаления, расположенную справа от каждого навыка. Пример тестирование секции Skills представлен на рисунке 4.4.

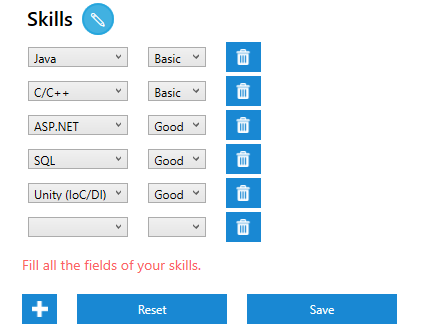


Рисунок 4.4 – Пример тестирования секции Skills

Далее перейдем к секции Projects. Данная секция позволяет добавлять проект, в котором он участвует, на страницу его профессиональной информации. В данной секции при добавлении нового элемента необходимо выбрать существующий проект и заполнить позицию сотрудника на данном проекте. Оба поля являются обязательными для заполнения. Пример тестирование секции Projects представлен на рисунке 4.5.

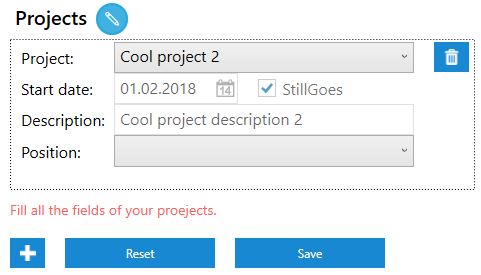


Рисунок 4.5 – Пример тестирования секции Projects

Перейдем к секции Education. Данная секция позволяет добавлять информацию о получении высшего образования пользователя. Все поля добавляемого элемента в данной секции являются обязательными для заполнения. Так же существует проверка на корректность даты в добавляемом элементе. Пример тестирования секции Educationпредставлен на рисунке 4.6.

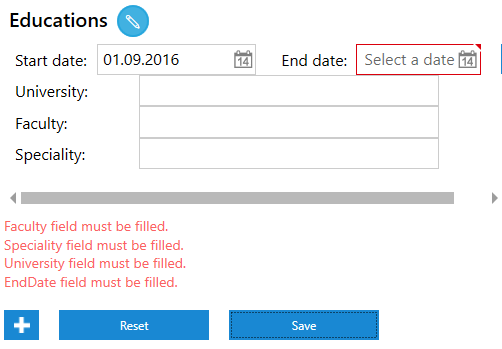


Рисунок 4.6 – Пример тестирования секции Education

Далее идет секция Certificates. Данная секция позволяет добавлять информацию о полученных сертификатах. Все поля добавляемого элемента в данной секции являются обязательными для заполнения. Длина вводимых данных ограничена самим элементом управления, поэтому ввести текстовое сообщение большой длины невозможно. Пример тестирования секции Certificates представлен на рисунке 4.7.

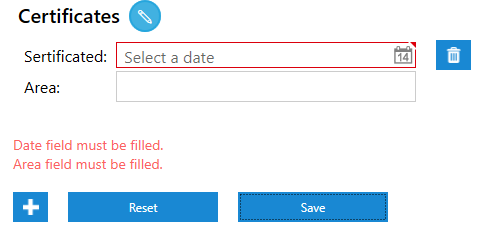


Рисунок 4.7 – Пример тестирования секции Certificates

И заключительной секцией данной страницы является секция Languages. Она позволяет добавлять информацию об иностранных языках, которыми владеет сотрудник. Пример тестирования секции Languages представлен на рисунке 4.8.

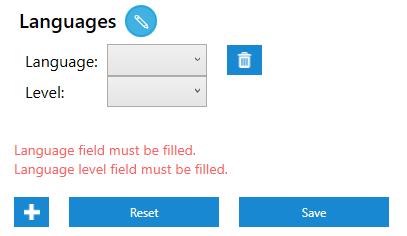


Рисунок 4.8 – Пример тестирования секции Languages

Все поля добавляемого элемента данной секции так же являются обязательными для заполнения.

### 4.3 Негативное тестирование функционала для роли «Менеджер»

Перейдем к тестированию функционала относящегося к роли «Менеджер». Пользователю с ролью «Менеджер» доступны следующие страницы: персональная страница с профессиональной информацией, поиск кандидатов по профессиональным навыкам, страница сохраненных поисковых запросов, управление профессиональными навыками и управление проектами. Персональная страница с профессиональной информацией уже была протестирована выше. Страница, содержащая сохраненные поисковые запросы, не содержит никаких текстовых полей, в связи с этим выполнить негативное тестирование для данной страницы не представляется возможным. Перейдем к тестированию страницы поиска кандидатов. На данной странице содержится 3 элемента ComboBox для выбора необходимых профессиональных навыков, иностранных языков и проектов, кнопка поиска, текстовое поля для сохранения поискового запроса и кнопка для инициации его сохранения. Под элементами управления для выбора навыков и иностранных языков находятся панели, позволяющие произвести расширенный поиск по выбранным элементам. Пример тестирование сохранения невалидного поискового запроса представлен на рисунке 4.9.

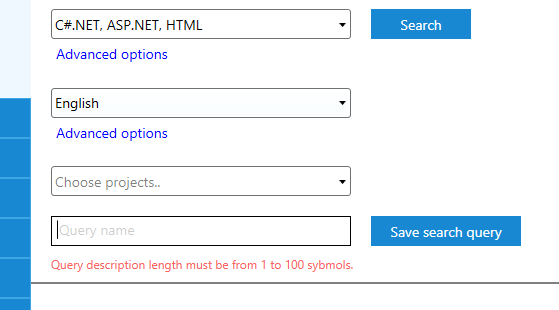


Рисунок 4.9 – Пример сохранения невалидного поискового запроса

Остальные элементы управления не позволяют выбирать невалидные значения, поэтому негативное тестирование для них выполнить не возможно.

Далее перейдем к тестированию страницы управления профессиональными навыками. На данной странице находится список существующих профессиональных навыков, текстовое поле для добавления нового навыка, кнопка добавления нового навыка, кнопка сброса и сохранения текущих изменений навыков. Пример негативного тестирования для добавления нового навыка представлен на рисунке 4.10.



Рисунок 4.10 – Пример тестирования добавления нового навыка

Так же на данной странице имеется возможность изменения имени уже существующих навыков. Нужно помнить, что при изменении имени профессионального навыка его длина должна быть от 2-х до 50-ти символов. Пример тестирования изменения имени существующего навыка представлен на рисунке 4.11.

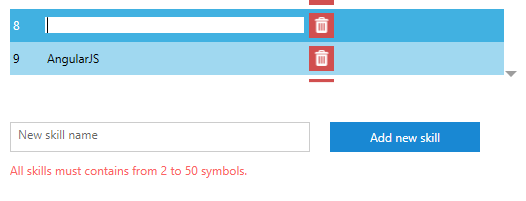


Рисунок 4.11 – Пример тестирования изменения имени навыка

Перейдем к тестированию страницы управления проектами. На данной странице содержится список существующих проектов, панель добавления новых проектов, кнопка сброса и сохранения текущих изменений по проектам. Пример тестирования добавления нового проекта с невалидными данными представлен на рисунке 4.12.

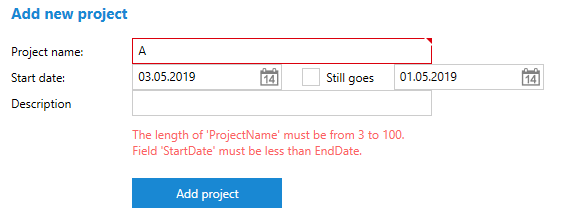


Рисунок 4.12 – Добавление нового проекта с невалидными данными

Так же при добавлении нового проекта начальная дата проекта должна быть меньше, чем дата его окончания.

### 4.4 Выводы по разделу

В данном разделе были рассмотрены примеры негативного тестирования страницы входа и регистрации нового сотрудника.

Так же были приведены примеры негативного тестирования страниц относящихся к пользователям с ролью «Администратор», «Менеджер» и «Сотрудник».

В ходе работы тестирования были протестированы все поля подразумевающие ввод данных, вследствие чего были минимизирована вероятность появления исключительных ситуаций в программном средстве.

### 5 Методика установки и использования

Утверд.

Н. контр.

Провер.

Разраб.

*Пацей Н.В.*

*Ковалевская Н.И*.

*Русакович А.Д.*

*Наркевич А.С.*

Ф.И.О.

Подпись

Дата

13

*Методика установки*

*и использования*

Лист

1

*ДП 05.00.ПЗ*

Лит.

Листов

*БГТУ 74417033, 2019*

### 5.1 Методика развертывания и требования приложения

Для корректной работы разрабатываемого программного средства компьютер должен соответствовать следующим требованиям к аппаратной части:

– тактовая частота центрального процессора не менее 800 МГц;

– объем оперативной памяти не менее 512 Мбайт;

– тактовая частота графического процессора не менее 600 МГц;

– наличие 500 Мбайт свободного места на жестком диске.

На данном компьютере должна быть установлена операционная система Windows версии 7 или выше, а также платформа .NET Framework версии 4.5 и выше. Для развертывания приложения на клиентской машине используется загрузочный пакет, который автоматически установит приложения на данном компьютере. Так же после установки приложения на клиентской машине должна быть сконфигурирована строка подключения к базе данных, которая находится в файле конфигураций App.config. Данная строка конфигурируется разработчиками программного средства, при предоставлении удаленного доступа к необходимой машине.

В данном приложении для сохранения информации используется реляционная база данных MS SQL. Для развертывания базы данных используется скрипты, поставляемые вместе с установочным модулем приложения. База данных не обязательно должна располагаться на клиентской машине. Она может быть расположена как в облаке, так и на серверной машине, использующейся в организации. В таком случае будет необходимо сконфигурировать строку подключения к базе данных в зависимости от местонахождения базы данных в сети.

Развертывание базы данных требует наличие на машине установленного

Microsoft SQL Server 2012*.* Так же для удобства управления базой данных рекомендуется установка Microsoft SQL Management Studio*.*

SQL Server 2012 Express поддерживает на операционные системы Windows версии 7 или выше.

Для установки SQL Server 2012 Express минимум необходимо:

* процессор с частотой не ниже 1 ГГц;
* размер оперативной памяти не менее 512 МБ;
* свободное место на жестком диске 4,2 ГБ.

Microsoft SQL Management Studio поддерживает следующие платформы:

* Windows 10 (64-bit);
* Windows 8.1 (64-bit);
* Windows Server 2016;
* Windows Server 2012 (64-bit).

### 5.2 Руководство пользователя

### 5.2.1 Экраны входа и регистрации в приложении

Взаимодействие зарегистрированного пользователя начинается с  
экрана авторизации, представленного на рисунке 5.1.

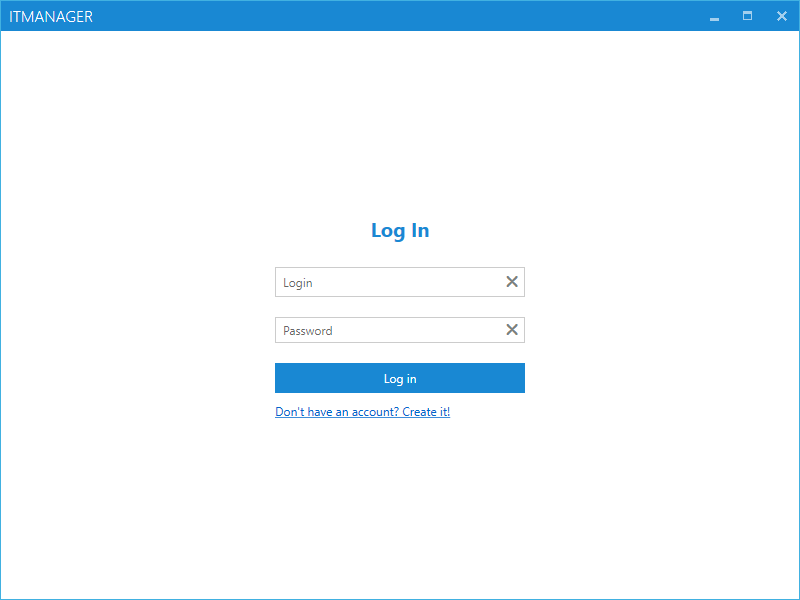


Рисунок 5.1 – Экран авторизации

На данном экране пользователь видит форму для ввода данных учетной записи, а именно логина и пароля, и кнопку «Log In». Прежде чем проверить наличие пользователя в базе данных и соответствие введенного пароля введенному логину производится валидация данных полей. После успешной валидации входных данных, и в случае наличия введенных данных в системе пользователь перенаправляется на стартовый экран, соответствующий его роли. Для роли «Администратор» данным экраном является управление ролями пользователей, для «Менеджера» – поиск кандидатов, а для роли «Сотрудник» – страница управления профессиональной информацией. В случае если пользователя не существует в системе, он может перейти по ссылке, ведущей на страницу регистрации и зарегистрироваться в ней.

На экране регистрации пользователь видит форму для ввода данных для новой учетной записи. Все поля на данной форме являются обязательными для заполнения. Так же все поля проходят валидацию прежде чем новый пользователь попадет в систему. Логин пользователя должен быть уникальным в системе.

После успешной регистрации новый пользователь получит роль «Сотрудник» и может войти в свою учетную запись используя форму входа. Чтобы получить роль «Менеджер» или «Администратор» необходимо обратиться к своему администратору.

Так же можете обратиться к своему администратору для создания учетной записи в случае временного отсутствия доступа к приложению. Он сможет создать необходимую учетную запись с первичным паролем и передать вам данные от нее.

Пример экрана регистрации представлен на рисунке 5.2.

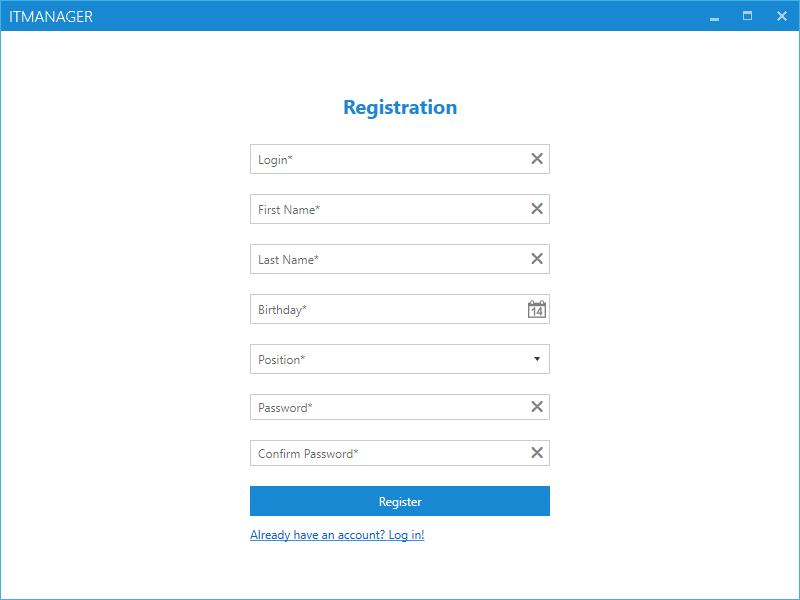


Рисунок 5.2 – Экран регистрации нового пользователя

После успешной регистрации пользователь будет автоматически перенаправлен на форму входа в приложение.

### 5.2.2 Руководство пользователя для роли «Администратор»

При авторизации пользователя, обладающим ролью «Администратор», приложение перенаправит его на страницу управления ролями пользователя. На данной странице находится список пользователей, которые имеют роль «Сотрудник» или «Менеджер». Имеется возможность сброса текущих изменений путем нажатия на кнопку «Reset», а так же возможность сохранения текущих изменений с помощью кнопки «Save». Checkbox в столбце «Is Changed» отображает наличие изменений у сотрудника, к которому он относится.

Пример страницы управления ролями представлен на рисунке 5.3.

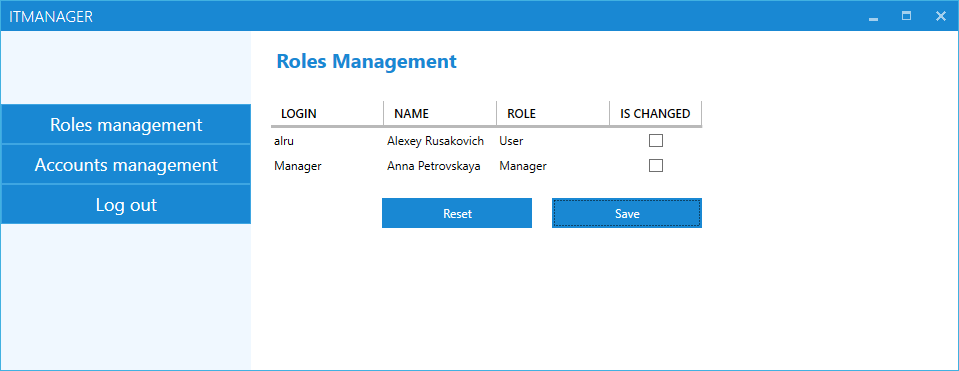


Рисунок 5.3 – Экран управления ролями пользователей

Так же у пользователя, имеющего роль «Администратор» имеется возможность управления учетными записями пользователя. На данной странице можно как заблокировать, так и разблокировать пользователя, имеется возможность скинуть текущий пароль, который станет первичным, а так же регистрации нового пользователя с первичным паролем. Пример экрана управления учетными записями пользователей представлен на рисунке 5.4.

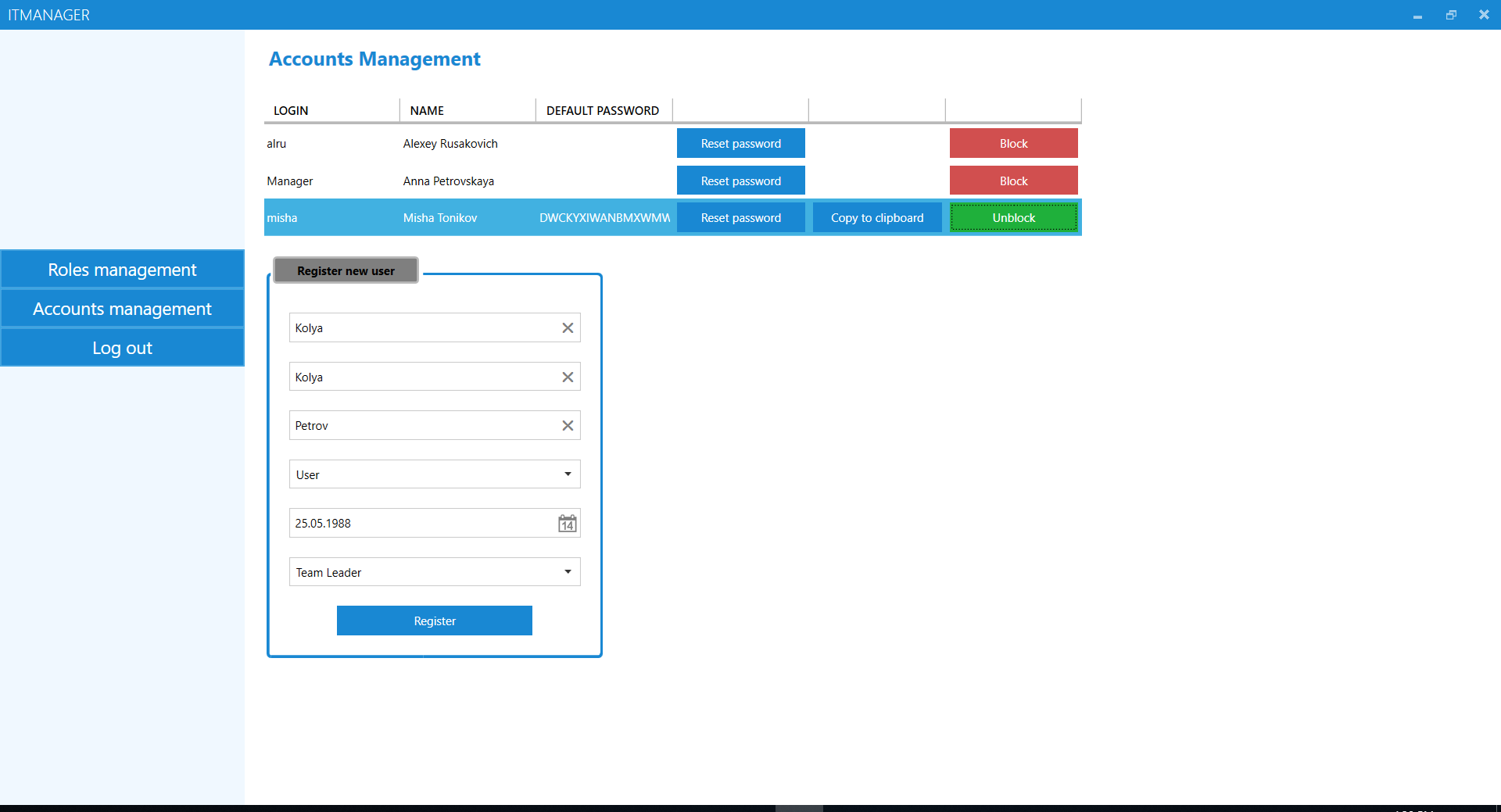


Рисунок 5.4 – Экран управления учетными записями пользователей

В столбце «Default Password» содержится временный пароль, в случае если соответствующая учетная запись была создана администратором, либо если же пароль был сброшен. Кнопка «Reset password» предназначена для сброса пароля. При нажатии на нее происходит автогенерация первичного пароля и появление кнопки «Copy to clipboard». Кнопка «Copy to clipboard» осуществляет сохранение данных учетной записи пользователя в буфер обмена в формате: «Login: <Login> Password: <Password>», которая в дальнейшем может быть отправлена сотруднику. В последнем столбце таблицы находится кнопка блокировки/разблокировки пользователя. При нажатии на кнопку блокировки пользователя, данная кнопка сменяется так же кнопкой которая позволяет разблокировать пользователя, и наоборот. Каждое текстовое поле формы регистрации содержит водяную подсказку отражающей какую информацию необходимо ввести в данное поле. Так же есть возможность удалить весь контент текстового поля по нажатию на знак крестика, находящегося в правой части элемента управления. В случае необходимости форма регистрации может быть скрыта, путем нажатии на Toggle Button с надписью «Register new user». Открытие данной формы осуществляется таким же образом.

В случае, если пользователь входит в учетную запись с использованием первичного пароля, то он обязан будет сменить его на новый используя форму смены первичного пароля.

Пример экрана смены первичного пароля представлен на рисунке 5.5.

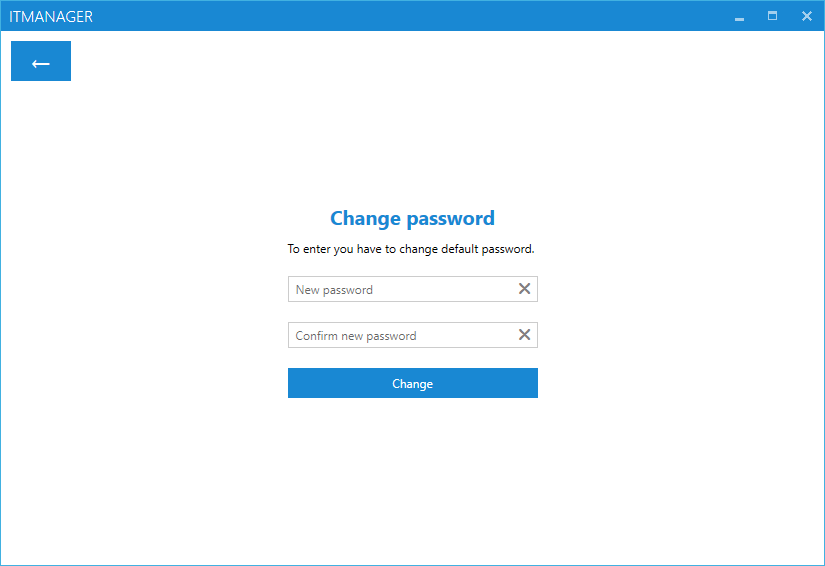


Рисунок 5.4 – Экран смены первичного пароля

Данный экран содержит в себе форму для смены пароля, а так же предусматривает возможность возврата на экран входа в приложение с помощью использования кнопки, расположенной в левом верхнем углу экрана. Данная форма так же применяет правила валидации на входные данные. Пароли, содержащиеся в текстовых полях должны совпадать, и иметь длину в диапазоне от 5 до 32 символов.

### 5.2.3 Руководство пользователя для роли «Менеджер»

При авторизации пользователя, обладающим ролью «Администратор», приложение перенаправит его на страницу поиска подходящих кандидатов. Пример экрана поиска кандидатов представлен на рисунке 5.5.

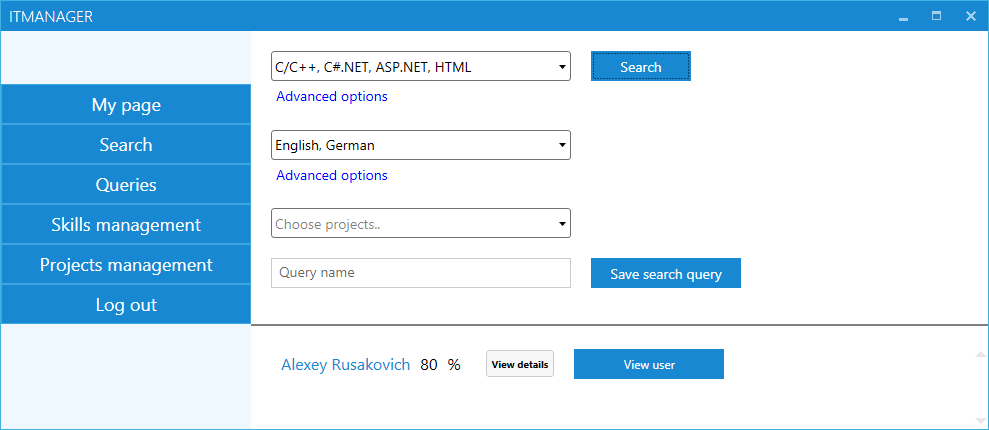


Рисунок 5.5 – Экран поиска кандидатов

На данном экране имеется возможность поиска кандидата по навыкам таких как иностранный язык и профессиональные навыки пользователя. У каждого профессионального навыка имеются расширенные опции, позволяющие отрегулировать необходимый уровень владения данным навыком. Данная секция расширенных опций может быть открыта путем нажатия на надпись «Advanced options». Так же есть возможность учесть принадлежность к какому–либо проекту компании, воспользовавшись элементом управления выпадающего списка с водяной надписью «Choose projects…». Если же необходимо использовать данный запрос в будущем, то можно воспользоваться формой его сохранения. Все запросы, сохраненные пользователем, имеющего роль «Менеджер» будут находиться на странице «Queries». При поиске для каждого кандидата рассчитывается процент, показывающий уровень соответствия искомым требованиям. Блок-схема алгоритма нахождения данного процента представлен в ДП 03.00.ГЧ. Так же существует возможность просмотреть статистику по кандидату путем нажатия на кнопку «View details». При нажатии на кнопку «View user» мы перейдем на персональную страницу данного пользователя.

Пример отчета поискового запроса кандидата по необходимым требованиям представлен на рисунке 5.6.

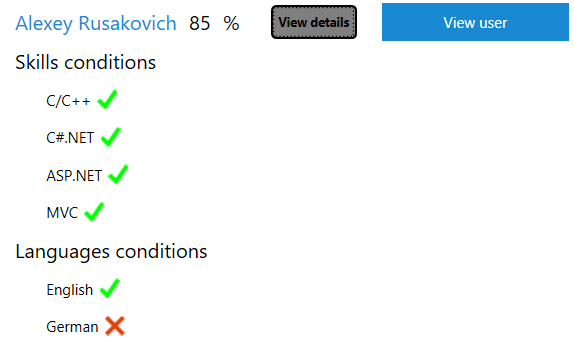


Рисунок 5.6 – Отчет поискового запроса по кандидату

У пользователя с ролью «Менеджер» так же есть возможность управления профессиональными навыками на странице «Skills management». На данной странице существует возможность добавления, удаления и редактирования профессиональных своих навыков.

На данной странице содержится список существующих профессиональных навыков в системе, а так же форма для их добавления. Каждая строка списка содержит в себе только для чтения уникальный идентификатор, предусматривающее изменение имя профессионального навыка, а так же кнопку удаления соответствующего навыка .

Пример страницы управления профессиональными навыками представлен на рисунке 5.7.

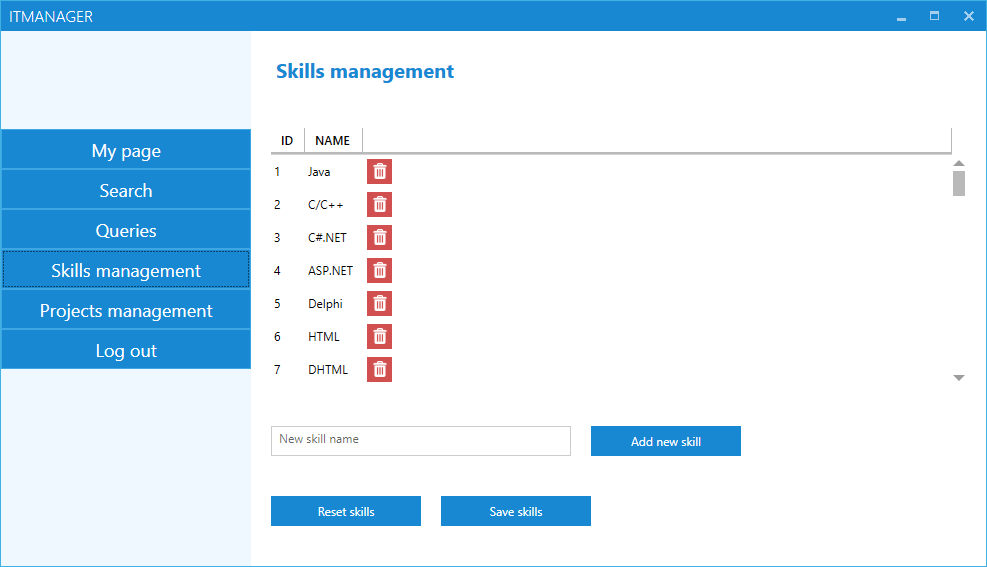


Рисунок 5.7 – Экран управления профессиональными навыками

Форма добавления нового профессионального навыка состоит из текстового поля, для ввода имени навыка, а так же кнопки для добавления этого навыка в систему. Причем его имя должно быть уникальным, в противном случае будет отображено сообщение правила валидации о уже существующем навыке.

На данной странице существует возможность сбросить все текущие изменения касающихся списка навыков путем нажатия на кнопку «Reset skills». По нажатию на кнопку «Save skills» все изменения, которые успешно прошли валидацию, будут сохранены в базе данных.

Перейдем к обзору экрана сохраненных поисковых запросов. Пример данного экрана представлен на рисунке 5.8.



Рисунок 5.8 – Экран сохраненных поисковых запросов

На данном экране содержится список сохраненных поисковых запросов. Каждый элемент списка содержит описание поискового запроса, кнопку «Execute query» и кнопку «Remove query». По нажатию на кнопку выполнения сохраненного поискового запроса менеджер будет перенаправлен на страницу поиска, а данный поисковой запрос будет применен. По нажатию на кнопку «Remove query» происходит удаление сохраненного поискового запроса.

Так же для данной роли существует возможность управления проектами. Для перехода на данную страницу необходимо нажать на кнопку «Projects management». Пример данной страницы представлен на рисунке 5.9.

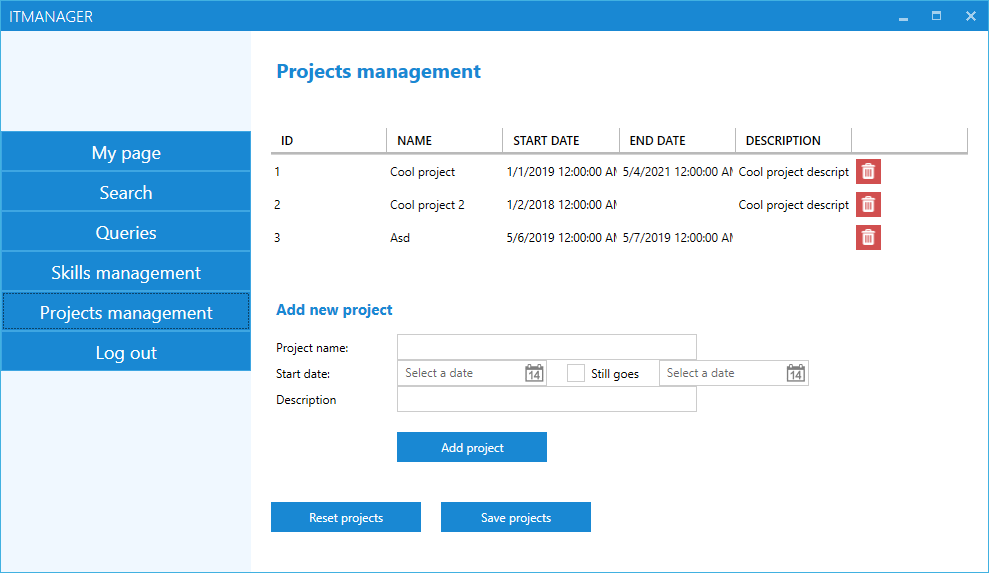


Рисунок 5.9 – Экран управления проектами

На данной странице находится список существующих проектов. Каждая запись списка содержит неизменяемый идентификатор проекта, неизменяемое имя проекта, так же неизменяемую дату начала проекта, разрешающую изменение поле даты конца проекта, изменяемое поле описание проекта и кнопку, позволяющую удалить соответствующий проект.

Так же на данной странице находится форма добавления нового проекта. На данной форме обязательным полем для заполнения является имя проекта и дата начала проекта. В случае если Checkbox «Still goes» не помечен галочкой, то поле окончание проекта так же является обязательным для заполнения. Поле описания проекта не является обязательным.

Checkbox с надписью «Still goes» позволяет указать является ли данный проект в активной стадии, либо же он уже был окочен. В случае если Checkbox был помечен галочкой, то поле для ввода даты окончания проекта исчезнет.

При нажатии на кнопку добавления нового проекта для введенной информации применяются правила валидации и если все входные данные корректны, то текущий проект будет добавлен в список проектов, после чего его можно будет сохранить в базу данных путем нажатия на кнопку «Save projects».

Кнопка «Reset projects» позволяет скинуть все текущие изменения по проектам.

### 5.2.4 Руководство пользователя для роли «Сотрудник»

При авторизации пользователя, обладающим ролью «Сотрудник», приложение перенаправит его на страницу управления профессиональной информацией. Пример экрана данной страницы представлен на рисунке 5.10.

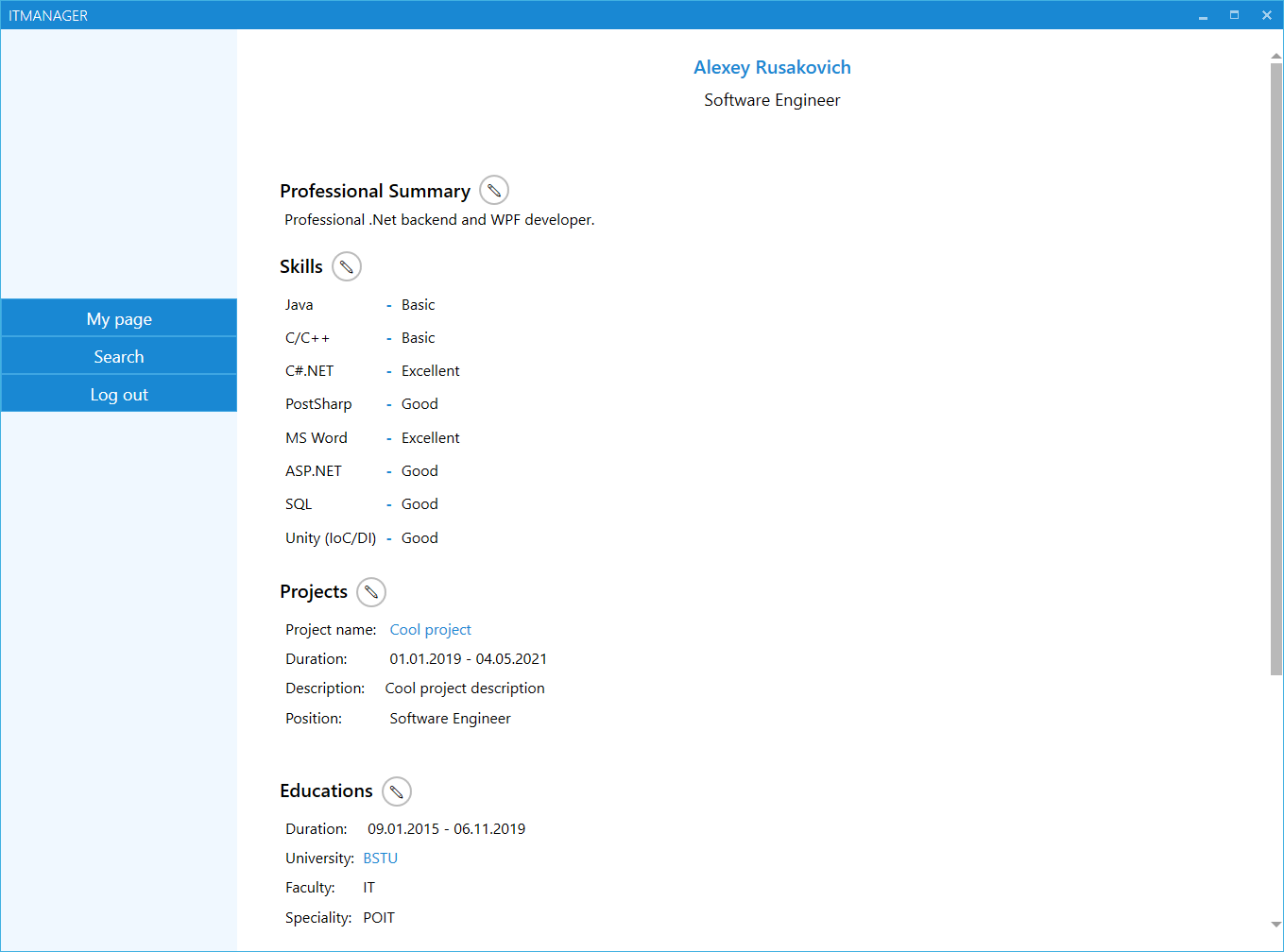


Рисунок 5.10 – Экран управления профессиональными навыками

Данный экран содержит следующие секции: краткое текстовое описание профессиональной деятельности, профессиональные навыки, участие в проектах компании, информация о высшем образовании, полученные сертификаты, владение иностранными языками. Перейдем к обзору секции Professional summary. Пример данной секции в режиме редактирования представлен на рисунке 5.11.

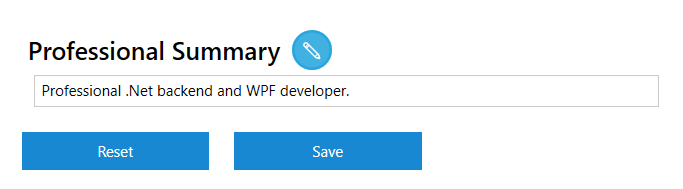


Рисунок 5.11 – Пример секции Professional summary в режиме редактирования

Данная секция содержит текстовое поле, поддерживающий ввод текста длиной до 1000 символов. Обычно в данной секции приводится краткое описание профессиональной деятельности сотрудника. Так же данная секция позволяет сбросить сделанные изменения с помощью кнопки «Reset» или сохранить их с помощью кнопки «Save». Сброс изменений и их сохранение в остальных секциях осуществляется аналогичным образом.

Перейдем к обзору секции Skills. Пример данной секции в режиме редактирования представлен на рисунке 5.12.

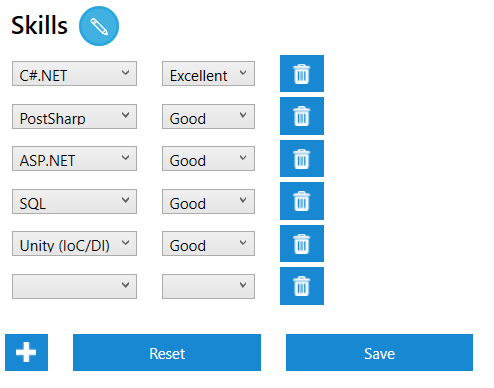


Рисунок 5.12 – Пример секции Skills в режиме редактирования

В режиме редактирования каждый элемент данной секции представляет собой ComboBox, содержащий список навыков, ComboBox, содержащий уровни владения навыками, а так же кнопку для возможности удаления соответствующего навыка. Все поля данной секции являются обязательными для заполнения.

При нажатии на кнопку, содержащую знак плюса происходит добавление нового элемента в данную секцию. Добавление элементов и их удаление в остальных секциях происходит аналогичным образом.

Перед возможностью сохранить изменения в данной секции происходит валидация входных данных. В случае успешной валидации данные сохраняются в базу данных, в обратном же случае отображаются сообщения правил валидации. Принцип логики сохранения данных аналогичен для последующих секций.

Перейдем к обзору секции Certificates. Пример секции в режиме редактирования представлен на рисунке 5.13.

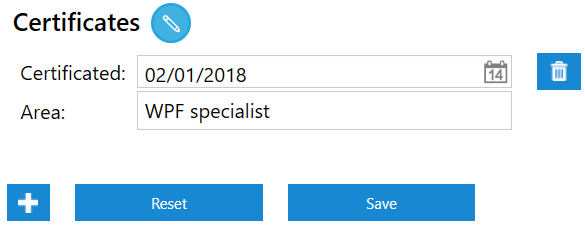


Рисунок 5.13 – Пример секции Certificates в режиме редактирования

Данная секция содержит в себе 2 текстовых поля обязательных для заполнение. Первое поле содержит дату получения сертификата, второе же содержит краткое описание полученного сертификата. Поле описание сертификата позволяет ввести текст длиной не более 100 символов.

Перейдем к обзору секции Projects. Пример данной секции в режиме редактирования представлен на рисунке 5.14.

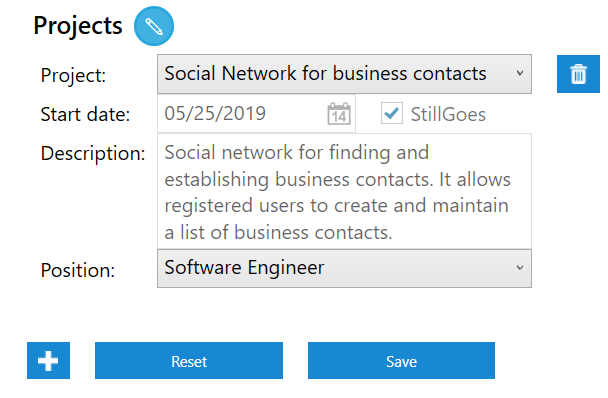


Рисунок 5.14 – Пример секции Projects в режиме редактирования

Данная секция содержит в себе информацию об имени проекта, дате начала и дате окончания, если проект был завершен. Так же данная секция содержит описание проекта и позицию сотрудника на данном проекте. В данной секции в поле сотрудник может выбрать один из проектов, находящихся в системе. Все данные, кроме позиции сотрудника на данном проекте подтянутся в текстовые поля и эти данные не будут доступны для редактирования. Пользователю же остается только выбрать позицию на данном проекте и сохранить текущие изменения. Все поля данной секции являются обязательными.

Перейдем к обзору секции Languages. Пример данной секции в режиме редактирования представлен на рисунке 5.15.

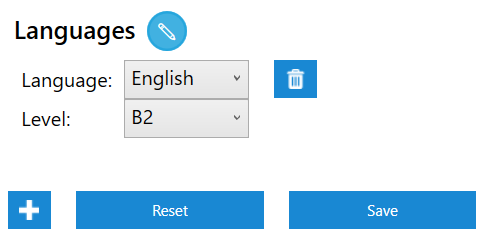


Рисунок 5.15 – Пример секции Languages в режиме редактирования

Данная секция содержит 2 поля для ввода: название иностранного языка и уровень владения данным языком. Список уровней владения иностранным языком содержит значения из общеевропейской шкалы языковой компетенции (The Common European Framework of Reference for Languages: Learning, Teaching, Assessment, CEFR или CEF), а именно: A1, A2, B1, B2, C1, C2. Все поля данной секции являются обязательными для заполнения. Так же нелья добавлять одинаковые языки.

Перейдем к обзору секции Educations. Пример данной секции в режиме редактирования представлен на рисунке 5.16.

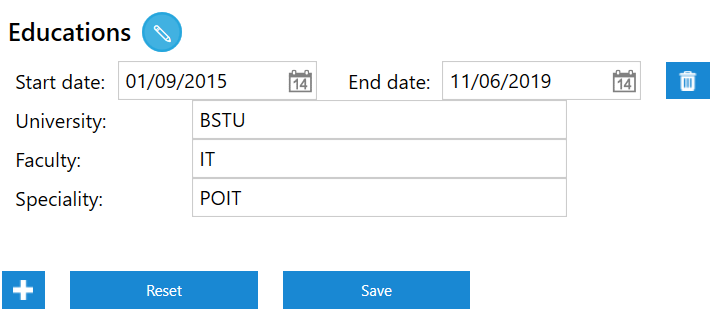


Рисунок 5.16 – Пример секции Educations в режиме редактирования

В данной секции заполняется информация о полученном или получаемом высшем образовании. Данная секция содержит следующие поля: дата начала обучения, дата конца обучения, название университета, название факультета, название специальности. Все поля данной секции являются обязательными. Поля университета, факультета и специальности разрешают ввод теста длиной до 100 символов.

Так же пользователь имеет возможность поиска других сотрудников на странице «Search». Пример страницы поиска представлен на рисунке 5.17.

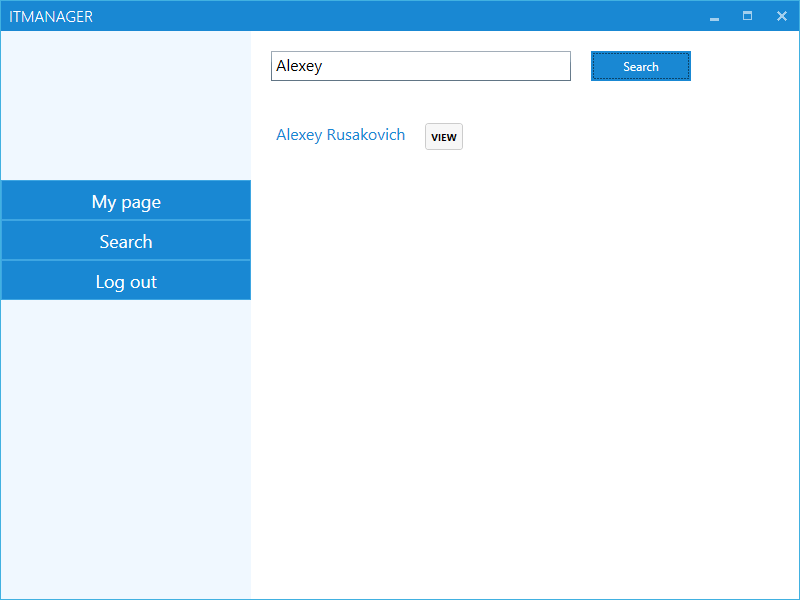


Рисунок 5.17 – Пример экрана поиска сотрудников

Поиск на данной странице осуществляется по имени и фамилии сотрудника. Справа от каждого найденного сотрудника находится кнопка для перехода на страницу профессиональной информации, относящуюся к данному сотруднику. Так же при переходе на страницу профессиональной информации проверяются права доступа. В случае их наличия разрешается редактирование находящихся на странице секций. В обратном случае все секции будут находиться в режиме просмотра.

### 5.3 Выводы по разделу

В данном разделе была описана методика развертывания и требования программного средства. Были приведены системные требования к нему, а так же необходимые действия по его конфигурации на клиентской машине.

Так же было предоставлено руководство пользователя для трех типов пользователей, предусмотренных в приложении для правильного использования программного средства. Были описаны назначения элементов управления содержащихся на страницах и их ограничения.

Был описан функционал для каждой роли и предоставлен графический материал, содержащий страницы или ее элементы.

### 6 Технико-экономическое обоснование проекта

Утверд.

Н. контр.

Провер.

Разраб.

*Пацей Н.В.*

*Ковалевская Н.И*.

*Русакович А.Д.*

*Наркевич А.С.*

Ф.И.О.

Подпись

Дата

1

*Технико-экономическое обоснование проекта*

Лист

1

*ДП 06.00.ПЗ*

Лит.

Листов

*БГТУ 74417033, 2019*

### 6.1 Общая характеристика разрабатываемого программного средства

При выполнении данного дипломного проекта использовалась технология WPF, совместно с фреймворком Prism. Для хранения данных данное программное средство использует базу данных. Доступ к данным осуществляется с помощью объектно-ориентированной технологии доступа к данным Entity Framework. Разработанное программное средство представляет средство управления профессиональной информацией о сотрудниках IT компании.

Данный раздел служит для определения затрат, произведенных на всех стадиях разработки программного средства.

### 6.2 Исходные данные и маркетинговый анализ

Исходные данные для расчёта приведены в таблице 6.1.

Таблица 6.1 – Исходные данные для расчёта

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование показателя | Единица измерения | Условные обозначения | Норматив |
| Численность разработчиков | чел. | Чр | 1 |
| Норматив дополнительной заработной платы | % | Ндз | 15 |
| Ставка отчислений в Фонд социальной защиты населения | % | Нфсзн | 34 |
| Ставка отчислений в БРУСП «Белгосстрах» | % | Нбгс | 0,4 |
| Цена одного машино-часа | руб. | Смч | 0,06 |
| Норматив прочих затрат | % | Нпз | 15 |
| Норматив накладных расходов | % | Нобп*,* обх | 150 |
| Норматив расходов на сопровождение и адаптацию | % | Нрса | 10 |
| Ставка НДС | % | Нндс | 20 |

Стоимость разработки зависит как от размера и сложности программного средства, так и от сферы бизнеса, на который оно ориентировано. В ходе проведения маркетингового анализа, была выявлена стоимость разработки программного средства управления профессиональной информацией о сотрудниках IT компании. Средняя цена разработки составляет 16000 рублей.

*Евлаш А.И.*

Консульт.

### 6.3 Методика обоснования цены

В современных рыночных экономических условиях программное средство (ПС) выступает преимущественно в виде продукции организаций, представляющей собой функционально завершенные и имеющие товарный вид, реализуемые покупателям по рыночным отпускным ценам. Все завершенные разработки являются научно-технической продукцией.

Широкое применение вычислительных технологий требует постоянного обновления и совершенствования ПС. Выбор эффективных проектов ПС связан с их экономической оценкой и расчетом экономического эффекта, который может определяться как у разработчика, так и у пользователя.

У разработчика экономический эффект выступает в виде чистой прибыли от реализации ПС, остающейся в распоряжении организации, а у пользователя – в виде экономии трудовых, материальных и финансовых ресурсов, получаемой за счет:

* снижения трудоемкости расчетов и алгоритмизации программирования и отладки программ;
* сокращения расходов на оплату машинного времени и других ресурсов на отладку программных средств;
* снижения расходов на материалы;
* оптимизации программных средств;
* ускорения ввода в эксплуатацию новых систем;
* улучшения показателей основной деятельности в результате использования передовых программных средств.

Стоимостная оценка программных средств у разработчиков предполагает определение затрат, что включает следующие статьи:

* заработная плата исполнителей – основная и дополнительная;
* отчисления в фонд социальной защиты населения;
* отчисления по обязательному страхованию от несчастных случаев на производстве и профессиональных заболеваний;
* расходы на материалы и комплектующие;
* расходы на спецоборудование;
* расходы на оплату машинного времени;
* прочие прямые затраты;
* накладные расходы.

На основании затрат рассчитывается себестоимость и отпускная цена конечного программного средства.

### 6.3.1 Объём программного средства

Для общей оценки объёма программного средства, функции приложения оцениваются с помощью специальной классификационной таблицы, представленной в приложении *Е*, в которой определяется объём каждой функции. Общий объем программного средства *Vo*, вычисляется как сумма объемов *Vi* каждой из n его функций (формула 6.1).

|  |  |  |
| --- | --- | --- |
|  |  | (6.1) |

В ходе рассмотрения классификационной таблицы были выбраны функции, присутствующие в итоговом программном средстве. В таблице 6.2 представлены функции в условных машино-командах.

Таблица 6.2 – Содержание и объем функций в программном средстве

|  |  |  |
| --- | --- | --- |
| № функции | Содержание функции | Объем, условных машино-команд |
| 101 | Организация ввода информации | 570 |
| 102 | Контроль, предварительная обработка | 240 |
| 111 | Управление вводом/выводом | 2 700 |
| 202 | Взаимодействие между компонентами системы | 600 |
| 401 | Взаимодействие с базой данных | 720 |
| 402 | Вспомогательные методы | 440 |
| 506 | Обработка ошибочных и сбойных ситуаций | 460 |
| 707 | Графический вывод результатов | 510 |

Исходя из данных таблицы 6.2, можно рассчитать объем программного средства, разработанного в процессе дипломного проектирования:

*Vo* = 570 + 240 + 2700 + 600 + 720 + 440 + 460 + 510 = 6 240 условных машино-команд

Уточненный объем программного средства *V*o*/* равен произведению объема программного средства *V*o на коэффициент изменения скорости обработки информации *Кск* (формула 6.2).

|  |  |  |
| --- | --- | --- |
|  | *V*o*/ = V*o*·*Кск | (6.2) |

*V*o*/* = 6240 ⋅ 0,6 = 3744 условных машино-команд

Исходя из вычисленного объема программного средства, можно определить уточненный объем программного средства.

### 6.3.2 Основная заработная плата

Для определения величины основной заработной платы, было проведено исследование величин заработных плат для специалистов в сфере разработки настольных приложений на базе фреймворка WPF. Источником данных служили открытые веб-порталы, различные форумы, официальная отчетность, а также общий средний уровень заработка в сфере информационных технологий. Итогом изучения и анализы полученных данных, стала информация о том, что средняя месячная заработная плата для позиции software engineer составляет 1500 рублей.

Проект разрабатывался одним человеком на протяжении двух месяцев. Основная заработная плата рассчитывается по формуле (6.3):

|  |  |
| --- | --- |
|  | (6.3) |

где Соз – основная заработная плата, руб.;

Траз – время раработки, месяцев;

Краз – количество разработчиков, человек;

Сзп – средняя месячная заработная плата.

руб.

Данная заработная плата была взята как медиана за 2019 год для позиции software engineer при опыте работы от 1 до 3-х лет.

### 6.3.3 Дополнительная заработная плата

Законодательство о труде предусматривает наличие выплат, которые определяются по нормативу в процентах к основной заработной плате по формуле (6.4):

|  |  |
| --- | --- |
|  | (6.4) |

где Соз– основная заработная плата, руб.;

Ндз – норматив дополнительной заработной платы, %.

Исходя из основной заработной платы, а также норматива дополнительной заработной платы, можно рассчитать сумму дополнительной заработной платы.

### 6.3.4 Отчисления в Фонд социальной защиты населения

Отчисления в Фонд социальной защиты населения (ФСЗН) определяются в соответствии с действующими законодательными актами по нормативу в процентном отношении к фонду основной и дополнительной зарплаты исполнителей и вычисляются по формуле (6.5):

|  |  |
| --- | --- |
|  | (6.5) |

где Соз – основная заработная плата, руб.;

Сдз – дополнительная заработная плата на конкретное ПС, руб.;

Нфсзн– норматив отчислений в Фонд социальной защиты населения, %.

руб.

Отчисления в БРУСП «Белгосстрах» вычисляются по формуле (6.6):

|  |  |
| --- | --- |
|  | (6.6) |

руб.

Данные платежи впоследствии направляются на финансирование страховых социальных выплат, к которым относятся:

* пенсии (трудовые и профессиональные) и многочисленные пособия (на погребение, по уходу за ребенком и пр.);
* покрытие расходов на оздоровление и санитарно-курортное лечение детей и взрослых;
* и т.д.

### 6.3.5 Расходы на материалы

Сумма расходов на материалы СМ определяется как произведение нормы расхода материалов в расчете на сто строк исходного кода НМ на уточненный объем программного средства *V*o*/* (формула 6.7).

|  |  |  |
| --- | --- | --- |
|  |  | (6.7) |

СМ = 0,460 ⋅ 3744 / 100 = 17,22 руб.

Учитывая, что норма расхода материалов в расчете на сто строк исходного кода равен 0,460 руб., можно определить сумму расходов на материалы.

### 6.3.6 Расходы на оплату машинного времени

Сумма расходов на оплату машинного времени Смв определяется как произведение стоимости одного машино-часа Смч на уточненный объем программного средства *V*o*/* и на норматив расхода машинного времени на отладку ста строк исходного кода НМВ (формула 6.8).

|  |  |  |
| --- | --- | --- |
|  |  | (6.8) |

Смв = 0,06 ⋅ 3744 ⋅ 15 / 100 = 33,7 руб.

Учитывая, что норматив машинного времени на отладку ста строк исходного кода равен 15 (по данным, приведенным в приложении 2 таблице П 2.11 «Оценка значений среднего машинного времени на отладку 100 строк исходного кода без применения ПС» методического пособия), можно определить сумму расходов на оплату машинного времени.

### 6.3.7 Прочие прямые затраты

Сумма прочих затрат Спз определяется как произведение основной заработной платы исполнителей на конкретное программное средство Соз на норматив прочих затрат в целом по организации Нпз (формула 6.9).

|  |  |  |
| --- | --- | --- |
|  |  | (6.9) |

Спз = 3000 ⋅ 15 / 100 = 450 руб.

К прочим прямым затратам относятся все расходы, связанные с приобретением и необходимые для приведения актива в состояние, пригодное к использованию.

### 6.3.8 Накладные расходы

Сумма накладныхрасходовСобп,обх – произведение основной заработной платы исполнителей на конкретное программное средство Соз на норматив накладных расходов в целом по организации Нобп,обх (формула 6.10).

|  |  |  |
| --- | --- | --- |
|  |  | (6.10) |

Собп,обх = 3000 ⋅ 150 / 100 = 4500 руб.

Все данные необходимые для вычисления есть, поэтому можно определить сумму накладных расходов.

### 6.3.9 Сумма расходов на разработку программного средства

Сумма расходов на разработку программного средства вычисляется по формуле (6.11):

|  |  |  |
| --- | --- | --- |
|  | Ср = Соз + Сдз + Сфсзн + Сбгс + См + Смв + Спз + Собп,обх | (6.11) |

Ср = 3000 + 450 + 1173 + 13,8 + 17,22 + 33,7 + 450 + 4500 = 9637 ,72 руб.

Ср определяется как сумма основной и дополнительной заработных плат исполнителей на конкретное программное средство, отчислений на социальные нужды, расходов на материалы, расходов на оплату машинного времени, суммы прочих затрат и суммы накладных расходов.

### 6.3.10 Расходы на сопровождение и адаптацию

Сумма расходов на сопровождение и адаптацию рассчитывается по формуле (6.12):

|  |  |
| --- | --- |
|  | (6.12) |

Срса = 9637 ,72 ⋅ 10 / 100 = 963,77 руб.

Срса определяется как произведение суммы расходов на разработки на норматив расходов на сопровождение и адаптацию Нрса.

### 6.3.11 Полная себестоимость

Полная себестоимость вычисляется по формуле (6.13):

|  |  |  |
| --- | --- | --- |
|  |  | (6.13) |

Сп = 9637 ,72 + 963,77 = 10601,49 руб.

Сп определяется как сумма двух элементов: суммы расходов на разработку Ср и суммы расходов на сопровождение и адаптацию программного средства Срса.

### 6.3.12 Определение цены, оценка эффективности

При расчете прибыли будем опираться на среднюю рыночную цену, по которой можно разработать данное программное обеспечение. Средняя рыночная цена составляет 16000 руб.

Прибыль рассчитывается по формуле (6.14):

|  |  |  |
| --- | --- | --- |
|  | , | (6.14) |

где Ппс – прибыль от реализации программного средства, руб.;

Цр – средняя рыночная цена продукта, руб.;

Сп – полная себестоимость программного средства, руб.;

Ппс = (16000/1,2) – 10601,49 = 2731.84 руб.

Уровень рентабельности разработанного программного средства определяется по формуле (6.15):

|  |  |
| --- | --- |
| , | (6.15) |

где Урент – уровень рентабельности программного средства, %

Сп – полная себестоимость программного средства, руб.;

Ппс – прибыль от реализации программного средства, руб.

Урент = (2731.84/ 10601,49) ⋅ 100 = 25,76%

Уровень рентабельности – один из лучших механизмов определения финансовой, инвестиционной и операционной эффективности производства.

### 6.4 Выводы по разделу

В таблице 6.4 представлены результаты расчетов для основных показателей данного раздела.

Таблица 6.4 – Результаты расчетов

|  |  |
| --- | --- |
| Наименование показателя | Значение |
| Время разработки, мес. | 2 |
| Количество программистов, чел. | 1 |
| Зарплата с отчислениями, руб. | 4636,8 |
| Расходы на материалы, оплату машинного времени, прочие, руб. | 500,92 |
| Накладные расходы, руб. | 4500 |
| Себестоимость разработки программного средства, руб. | 9637 ,72 |
| Расходы на сопровождение и адаптацию, руб. | 963,77 |
| Полная себестоимость, руб. | 10601,49 |
| Цена аналога, руб. | 16000 |
| Прибыль от реализации, руб. | 2731.84 |
| Рентабельность разработки, % | 25,76 |

С целью повышения производительности труда за счет снижения времени на документооборот в IT компаниях проводят ряд организационно-технических мероприятий: применение новых технологий, замена морально и физически устаревшего оборудования на более технологичное и высокопроизводительное, применение спецприспособлений, обучение персонала. Предложенные мероприятия позволяют повысить конкурентоспособность данного вида приложения за счет снижения себестоимости и сокращения времени на работу с документами.

Необходимость разработки программного средства обусловлена повышенным спросом на данного вида продукт.

Разработка программного средства, осуществляемая одним программистом в течении двух месяцев, при заданных условиях обойдется компании в 10601,49 руб. Реализации данного программного средства по среднерыночной цене в 16000 руб., принесет прибыль компании в сумме 2731.84 руб. Уровень рентабельности составит 25,76%.

Результаты проведенных расчетов показывают, что проектируемое программное средство для управления информацией о сотрудниках IT компании является рентабельным и экономически обоснованным, а отпускная цена является приемлемой для потребителя, обеспечит конкурентоспособность на внутреннем рынке.

### Заключение

Утверд.

Н. контр.

Провер.

Разраб.

*Пацей Н.В.*

*Ковалевская Н.И*.

*Русакович А.Д.*

*Наркевич А.С.*

Ф.И.О.

Подпись

Дата

1

### *Заключение*

Лист

1

*ДП 00.00.ПЗ*

Лит.

Листов

*БГТУ 74417033, 2019*

В ходе выполнения дипломного проекта было реализовано программное средство управления профессиональной информацией о сотрудниках IT компании. Принял участие в 70-й научно-технической конференции учащихся, студентов и магистрантов БГТУ. Дипломный проект выполнен и оформлен в соответствии с требованиями действующих нормативных документов [13].

В процессе выполнения цели дипломного проекта, была спроектирована база данных. Была описана структура программного средства и принцип взаимодействия между его компонентами.

Было проведено негативное тестирования основных экранов программного средства, произведен обзор основных ошибок.

Был разработан интуитивно понятный интерфейс и руководство пользователя, которое описывает основные сценарии работы с системой. Это позволит пользователю легко ориентироваться в программном средстве и работать с ним.

Данная система является законченным программным продуктом, который реализует все перечисленные возможности в полной форме. Программное средство использует архитектуру, которая в дальнейшем позволит быстро и безопасно расширить его функционал.

Программное средство соответствует цели дипломного проекта, реализует все поставленные перед ней задачи.

В качестве перспектив развития программного средства можно предложить:

* реализацию механизма взаимодействия между сотрудником и менеджером с помощью чата;
* построение системы сбора статистики и генерации отчетной документации в соответствии с требованиями предприятия по оформлению данного типа отчетности;
* добавление возможности генерации резюме на основе профессиональной информации сотрудника;
* добавление пользовательских настроек приложения, позволяющих изменить его внешний вид и сценарии работы в соответствии с пожеланиями конкретного пользователя;
* построение экосистемы приложений, включающее веб-сайт и мобильные приложения для разных платформ, использующие разработанный веб-сервис как разделяемую серверную часть.

### Список использованных источников

Утверд.

Н. контр.

Провер.

Разраб.

*Пацей Н.В.*

*Ковалевская Н.И*.

*Русакович А.Д.*

*Наркевич А.С.*

Ф.И.О.

Подпись

Дата

1

### *Список использованных источников*

Лист

1

*ДП 00.00.ПЗ*

Лит.

Листов

*БГТУ 74417033, 2019*

1. HR Online Leave Management System [Electronic resource] / Online HR & Payroll – Mode of access: https://hr.my/. *–* Dateof access:01.03.2019.

2. Human Resources Software [Electronic resource] / Human Resources management – Mode of access: https://factorialhr.com/. *–* Dateof access:01.03.2019.

3. Online HR Software [Electronic resource] / Cloud Based Social HR Software – Mode of access: https://webhr.co/. *–* Dateof access:01.03.2019.

4. Стиллмен, Э. Изучаем C# / Э. Стиллмен, Дж. Грин. – 3-е изд. – СПб.: Питер, 2014. – 816 с.

5. Мак-Дональд, М. WPF 4: Windows Presentation Foundation в .NET 4.0 с примерами на C# 2010 для профессионалов / М. Мак-Дональд. – Москва: ООО «И. Д. Вильямс», 2011. – 1024 с.

6. Prism Library [Electronic resource] / Prism Library Documentation – Mode of access: <https://prismlibrary.github.io/>. *–* Dateof access:15.03.2019.

7. Material Design [Electronic resource] / Material Design Documentation – Mode of access: <https://material.io/>. *–* Dateof access:16.03.2019.

8. Фримен, А. LINQ: язык интегрированных запросов в C#2010 для профессионалов / А. Фримен, Дж. С. Раттц-мл.– Москва.: ООО «И. Д. Вильямс», 2011. – 656 с.

9. Сеть разработчиков Microsoft [Электронный ресурс] / Документация – Режим доступа: <https://msdn.microsoft.com/>. – Дата доступа 17.03.2019.

10. Рихтер, Дж. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C#. / Дж. Рихтер. – 4–е изд. – СПб.: Питер, 2013. – 896 с.

11. Гамма, З. Приемы объектно-ориентированного проектирования. Паттерны проектирования / З. Гамма. – СПб.: Питер, 2001. – 358 с.

12. Макконнелл, С. Совершенный код. Мастер–класс / С. Макконнелл – Москва.: Издательство «Русская редакция», 2010. – 896 с.

13. СТП БГТУ 001–2019, Стандарт организации. Проекты (работы) дипломные. Требования и порядок подготовки, представления к защите и защиты. – Минск: БГТУ, 2019. – 72 с.

### ПРИЛОЖЕНИЕ А

**Листинг кода файла Bootstrapper.cs**

public class Bootstrapper: UnityBootstrapper {

protected override DependencyObject CreateShell() {

return Container.Resolve < ShellView > ();

}

protected override void InitializeShell() {

base.InitializeShell();

App.Current.MainWindow = (Window) Shell;

App.Current.MainWindow.Show();

}

protected override void ConfigureContainer() {

base.ConfigureContainer();

Container.RegisterType < IEventAggregator, EventAggregator > (new ContainerControlledLifetimeManager());

Container.RegisterType < INavigationService, NavigationService > (new ContainerControlledLifetimeManager());

Container.RegisterTypeForNavigation < ShellView > (Constants.ShellView);

Container.RegisterTypeForNavigation < LoginView > (Constants.LoginView);

Container.RegisterTypeForNavigation < RegisterView > (Constants.RegisterView);

Container.RegisterTypeForNavigation < RolesManagementView > (Constants.RolesManagementView);

Container.RegisterTypeForNavigation < AccountsManagementView > (Constants.AccountsManagementView);

Container.RegisterTypeForNavigation < ChangePasswordView > (Constants.ChangePasswordView);

Container.RegisterTypeForNavigation < MyPersonalPageView > (Constants.MyPersonalPageView);

Container.RegisterTypeForNavigation < QueriesView > (Constants.QueriesView);

Container.RegisterTypeForNavigation < SearchView > (Constants.SearchView);

Container.RegisterTypeForNavigation < SkillsManagementView > (Constants.SkillsManagementView);

Container.RegisterTypeForNavigation < ProjectsManagementView > (Constants.ProjectsManagementView);

Container.RegisterTypeForNavigation < DefaultSearchView > (Constants.DefaultSearchView);

}

### ПРИЛОЖЕНИЕ Б

**Листинг кода файла BaseViewModel.cs**

public class BaseViewModel: INotifyPropertyChanged, IDataErrorInfo {

public Visibility ErrorsVisibility => string.IsNullOrWhiteSpace(Errors) ? Visibility.Collapsed : Visibility.Visible;

public string Errors {

get;

set;

}

public bool DoValidation {

get;

set;

} = false;

public string Title {

get;

private set;

}

public bool IsBusy {

get;

set;

}

public ICommand LoadedCommand {

get;

set;

}

public event PropertyChangedEventHandler PropertyChanged;

public BaseViewModel(string \_title) {

Title = \_title;

LoadedCommand = new DelegateCommand(LoadedMethod);

}

private void LoadedMethod() {

DoValidation = true;

}

protected void Set < T > (ref T target, T value, [CallerMemberName] string propertyName = "") {

target = value;

RaisePropertyChanged(propertyName);

}

protected void RaisePropertyChanged([CallerMemberName] String propertyName = "") {

PropertyChanged ? .Invoke(this, new PropertyChangedEventArgs(propertyName));

}

# region Data validation

public virtual string Validate(string propertyName) {

return null;

}

protected bool IsValid(string[] validatingStrings, out string Error) {

Error = null;

foreach(var str in validatingStrings) {

var error = Validate(str);

if (!string.IsNullOrWhiteSpace(error))

Error += $ "{error}\r\n";

}

return Error is null;

}

string IDataErrorInfo.Error {

get {

return null;

}

}

public string this[string columnName] {

get {

string error = null;

return error = Validate(columnName);

}

}

# endregion

}

### ПРИЛОЖЕНИЕ В

**Листинг кода файла ShellViewModel.cs**

public class ShellViewModel: BaseViewModel {

private static User currentUser {

get;

set;

}

public static User CurrentUser {

get {

return currentUser;

}

set {

currentUser = value;

CurrentUserId = currentUser == null ? -1 : currentUser.Id;

}

}

public static int CurrentUserId {

get;

set;

}

public Visibility MenuVisibility {

get;

set;

}

private readonly IEventAggregator \_eventAggregator;

private readonly IRegionManager \_regionManager;

public ShellViewModel(IEventAggregator eventAggregator, IRegionManager regionManager, INavigationService navigationService): base("Shell") {

\_eventAggregator = eventAggregator;

\_regionManager = regionManager;

\_regionManager.RegisterViewWithRegion(Helpers.Constants.MainRegion, typeof(LoginView));

\_eventAggregator.GetEvent < CloseMenuEvent > ().Subscribe(MenuStateChanged);

MapperInitializer.Initialize();

}

private void MenuStateChanged(bool value) {

MenuVisibility = value ? Visibility.Collapsed : Visibility.Visible;

RaisePropertyChanged(nameof(MenuVisibility));

}

}

### ПРИЛОЖЕНИЕ Г

**Листинг кода файла MyPersonalPageViewModel.cs**

public class MyPersonalPageViewModel: BaseViewModel, INavigationAware {

// Current user, taken from database

public User User {

get;

set;

}

#

region Data validation properties

public string SkillsErrors {

get;

set;

}

public Visibility SkillsErrorsVisibility => string.IsNullOrWhiteSpace(SkillsErrors) ? Visibility.Collapsed : Visibility.Visible;

public string ProjectsErrors {

get;

set;

}

public Visibility ProjectsErrorsVisibility => string.IsNullOrWhiteSpace(ProjectsErrors) ? Visibility.Collapsed : Visibility.Visible;

public string EducationErrors {

get;

set;

}

public Visibility EducationErrorsVisibility => string.IsNullOrWhiteSpace(EducationErrors) ? Visibility.Collapsed : Visibility.Visible;

public string CertificatesErrors {

get;

set;

}

public Visibility CertificatesErrorsVisibility => string.IsNullOrWhiteSpace(CertificatesErrors) ? Visibility.Collapsed : Visibility.Visible;

public string LanguagesErrors {

get;

set;

}

public Visibility LanguagesErrorsVisibility => string.IsNullOrWhiteSpace(LanguagesErrors) ? Visibility.Collapsed : Visibility.Visible;

#

endregion

# region Commands

public ICommand SaveProfessionalSummary {

get;

set;

}

public ICommand ResetProfessionalSummary {

get;

set;

}

public ICommand SaveSkills {

get;

set;

}

public ICommand ResetSkills {

get;

set;

}

public ICommand AddSkill {

get;

set;

}

public ICommand RemoveSkill {

get;

set;

}

public ICommand SaveProjects {

get;

set;

}

public ICommand ResetProjects {

get;

set;

}

public ICommand AddProject {

get;

set;

}

public ICommand RemoveProject {

get;

set;

}

public ICommand SaveEducations {

get;

set;

}

public ICommand ResetEducations {

get;

set;

}

public ICommand AddEducation {

get;

set;

}

public ICommand RemoveEducation {

get;

set;

}

public ICommand SaveSertificates {

get;

set;

}

public ICommand ResetSertificates {

get;

set;

}

public ICommand AddSertificate {

get;

set;

}

public ICommand RemoveSertificate {

get;

set;

}

public ICommand SaveLanguages {

get;

set;

}

public ICommand ResetLanguages {

get;

set;

}

public ICommand AddLanguage {

get;

set;

}

public ICommand RemoveLanguage {

get;

set;

}

#

endregion

# region Constants collections

public IList < Project > ProjectsList {

get;

set;

}

public IList < LanguagesList > LanguagesList {

get;

set;

}

public IList < LanguageLevel > LanguageLevels {

get;

set;

}

public IList < SkillLevel > SkillLevels {

get;

set;

}

public IList < Position > Positions {

get;

set;

}

public IList < ProfessionalSkill > ProfessionalSkills {

get;

set;

}

# endregion

# region User properties

public bool CanUserChangeData {

get;

set;

}

public string UserName {

get;

set;

}

public Models.UserPageModel.Position Position {

get;

set;

}

public Models.UserPageModel.ProfessionalSummary ProfessionalSummary {

get;

set;

}

public ObservableCollection < Models.UserPageModel.UserSkill > Skills {

get;

set;

}

public ObservableCollection < Models.UserPageModel.Project > Projects {

get;

set;

}

public ObservableCollection < Models.UserPageModel.Education > Educations {

get;

set;

}

public ObservableCollection < Models.UserPageModel.Sertificate > Sertificates {

get;

set;

}

public ObservableCollection < Models.UserPageModel.Language > Languages {

get;

set;

}

# endregion

# region Toggle buttons properties

public bool IsProfessionalSummaryChecked {

get;

set;

}

public bool IsSkillsChecked {

get;

set;

}

public bool IsProjectsChecked {

get;

set;

}

public bool IsEducationsChecked {

get;

set;

}

public bool IsSertificatesChecked {

get;

set;

}

public bool IsLanguagesChecked {

get;

set;

}

#

endregion

}