ГБПОУ МО «Ступинский техникум им. А.Т. Туманова»

Курсовая работа

по МДК 01.01 «Разработка программных модулей»

Тема: Разработка информационной системы для центра занятости населения.

Специальность: 09.02.07 «Информационные системы и программирование»

Выполнил(а):
Студент 4 курса группы ИСП-4

Паламарчук Анатолий Анатольевич

Проверил:

Жильцов Алексей Анатольевич

Ступино, 2023

Содержание

Введение

В настоящее время существует огромное количество программных продуктов, которые применяются во многих сферах бизнеса. Это специализированные профессиональные информационные системы, повышающие качество работы с клиентами и эффективность работы компании, автоматизирующие многие рутинные операции.

Основная цель профессиональной ориентации службы занятости - оказывать содействие гражданам, обращающимся в службу занятости, в получении подходящей работы в соответствии с их личными интересами, потребностями работодателей и рынка труда путем их профессионального информирования, консультирования.

Система профессиональной ориентации в государственной службе занятости предоставляет следующие услуги:

- информирование и консультирование граждан, обращающихся в службу занятости в целях выбора работы, режима труда;

- профессиональная ориентация безработных граждан.

Основная проблема в работе рекрутерских фирм - отсутствие взаимопонимания с клиентами. Заказывая работника, российские клиенты обычно сами не знают, чего именно они хотят. Нередки случаи, когда потенциальные работодатели просят фирму подобрать специалиста, но при этом название должности, функциональные обязанности будущего работника и предлагаемая ему заработная плата между собой никак не коррелируют. Во многом это связанно с тем, что в большинстве фирм обязанности сотрудников сформулированы плохо и без учета их квалификации.

Подав заявление в систему, трудоустраиваемый работник или работодатель становится ее клиентом и начинает обслуживаться на протяжении срока обслуживания заявки. Срок обслуживания заявки рассматривается несколько месяцев. Если за это время заявка не выполняются то она возвращается. Заявка представляет собой анкету.

Основным назначением системы является автоматизация ввода и хранения данных по трудоустраиваемым гражданам и работодателям. Система позволяет изменять, дополнять, вести поиск и просмотр информации о трудоустраиваемых гражданах и работодателях.

В данной курсовой работе рассматривается проектирование АИС для центра занятости населения. Приложение должно осуществлять:

– хранение и редактирование данных о вакансиях, резюме;

Объектом исследования является центр занятости населения пгт Михнево.

Предмет исследования – подбор вакансий для соискателей.

При разработке информационной системы нужно решить две задачи – разработка базы данных для хранения информации и разработка клиентского приложения для более автоматизированного и простого учета базы данных.

Целью развития и внедрения современной автоматизированной системы управления является автоматизация ввода и хранения данных по трудоустраиваемым гражданам и работодателям. Система позволяет изменять, дополнять, вести поиск и просмотр информации о трудоустраиваемых гражданах и работодателях.

Основная цель профессиональной ориентации службы занятости - оказывать содействие гражданам, обращающимся в службу занятости, в получении подходящей работы в соответствии с их личными интересами, потребностями работодателей и рынка труда путем их профессионального информирования, консультирования.

Система профессиональной ориентации в государственной службе занятости предоставляет следующие услуги:

- информирование и консультирование граждан, обращающихся в службу занятости в целях выбора работы, режима труда;

- профессиональная ориентация безработных граждан.

Цель работы – разработать информационную систему для центра занятости населения.

Следующими задачами для достижения цели является:

- Описать Аналитическую часть;
- Описать предметную область и функции решаемых задач;
- Описать входную и выходную информацию;
- Постановка задачи;
- Описать средства разработки ПО;
- Спроектировать и разработать базу данных;
- Спроектировать концептуальную модель БД;
- Построить логическую модель БД;
- Построить физическую модель БД;
- Реализовать проект в среде конкретной СУБД;
- Создать клиентскую части информационной системы;
- Выполнить тестирование информационной системы;

Для выполнения практической части работы были применены следующие средства и инструменты:

1. UML – язык для определения, визуализации, конструирования и документирования программных систем;
2. Среда разработки программного обеспечения Visual Studio 2019, язык программирования C#;
3. Среда разработки информационного обеспечения (базы данных) SQLManagment;
4. CASE-система для построения моделей системы – Erwin Data Modeler.
5. CASE-система для построения моделей бизнес-процессов – BPwin.

1. Аналитическая часть

1.1. Описание предметной области и функции решаемых задач

Автоматизированные информационные системы (АИС) на сегодняшний день все больше внедряются в жизнь любой организации, предприятия. Они помогают автоматизировать процесс на всех уровнях, успешно контролировать и улучшать выполнение работы государственных программ, постановлений. АИС – это программы самого разного направления, предназначенные для решения рабочих процессов, в любой сфере деятельности (финансы, оформление заявлений, постановка на учет и т.д.). При этом особым отличием этих программ является то, что они создаются для человека, владеющего ПК на уровне среднего пользователя.

Центр занятости населения (ЦЗН) в виде базы данных (БД) актуальных (свежих) предложений работы (объявлений вакансий) в пгт Михнево от прямых работодателей и соискателей работы (работников) в форме анкет резюме. Каждый работодатель может условно-бесплатно разместить подробную информацию о вакансии с целью подбора кадров, поиска персонала на вакантное место работы, а лицо, которое ищет работу, может условно-бесплатно составить и оставить свое подробное резюме, чтобы трудоустроиться в своем городе (регионе).

Вакансии нашего Электронного ЦЗН (Центра Занятости Населения) пгт Михнево помогут быстро найти подходящую работу соискателям. Поиск кадров с помощью нашего городского сайта о работе также решается в кратчайшие сроки. Поиск работы в пгт Михнево доступен в виде расширенного запроса по базе данных объявлений вакансий с учетом конкретного города. Поиск сотрудников в пгт Михнево доступен по банку данных актуальных резюме с учетом требуемого населенного пункта, а следовательно и прилегающей к нему области.

Отметим, что работа в виде объявлений с вакансиями в пгт Михнево размещается представителями прямых работодателей, специалистами

отделов кадров организаций, фирм, компаний в виде анкет объявлений с вакансиями таким образом, что предложения работы (вакансии) всегда актуальны, свежие. Наш условно-бесплатный мультирегиональный интернет-сайт Центра занятости населения решает в том числе задачи Центра занятости молодежи, а именно трудоустройство подростков, школьников, учащихся и выпускников профтехучилищ, лицеев, ВУЗов.

Гордостью нашего электронного мультирегионального Центра трудоустройства населения является обширный электронный банк данных высококвалифицированных специалистов в области "Продаж", "Маркетинга", "Менеджмента", "Финансов", "Юриспруденции", "IT-индустрии", "Строительства", "Торговли"

Автоматизированная информационная система для центра занятости населения должна обеспечивать выполнение функций:

- ввод, вывод, редактирование, хранение информации о резюме:
- ввод, вывод, редактирование, хранение информации о вакансиях;
- Авторизация пользователей
- Поиск и сортировка информации

Целью курсовой работы является повышение эффективности работы центра занятости населения путем автоматизации процесса.

К задачам курсовой работы относятся:

- изучение предметной области,
- концептуальное и логическое проектирование БД,
- разработка БД,
- проектирование и разработка информационной системы.

Для общего представления деятельности центра занятости населения использовался унифицированный язык моделирования UML (Unified Modeling Language). Он применяется для моделирования любых систем: от информационных масштаба предприятия до распределенных web-приложений и даже встроенных систем реального времени. Это очень выразительный язык,

позволяющий рассмотреть систему со всех точек зрения, имеющих отношение к ее разработке и последующему развертыванию.

В данной работе рассмотрим диаграммы UML: варианты использования, последовательность действий, классы, компоненты. Для этого применялась среда разработки StarUML™, которая превосходно настраивается в соответствии с требованиями пользователя и имеет высокую степень расширяемости, особенно в области своих функциональных возможностей.

Диаграммы вариантов использования описывают функциональное назначение системы или то, что система должна делать. Разработка диаграммы преследует следующие цели:

– определить общие границы и контекст моделируемой предметной области;

– сформулировать общие требования к функциональному поведению проектируемой системы;

– разработать исходную концептуальную модель системы для ее последующей детализации в форме логических и физических моделей;

– подготовить исходную документацию для взаимодействия разработчиков системы с ее заказчиками и пользователями. Диаграмма вариантов использования представлена на рисунке.

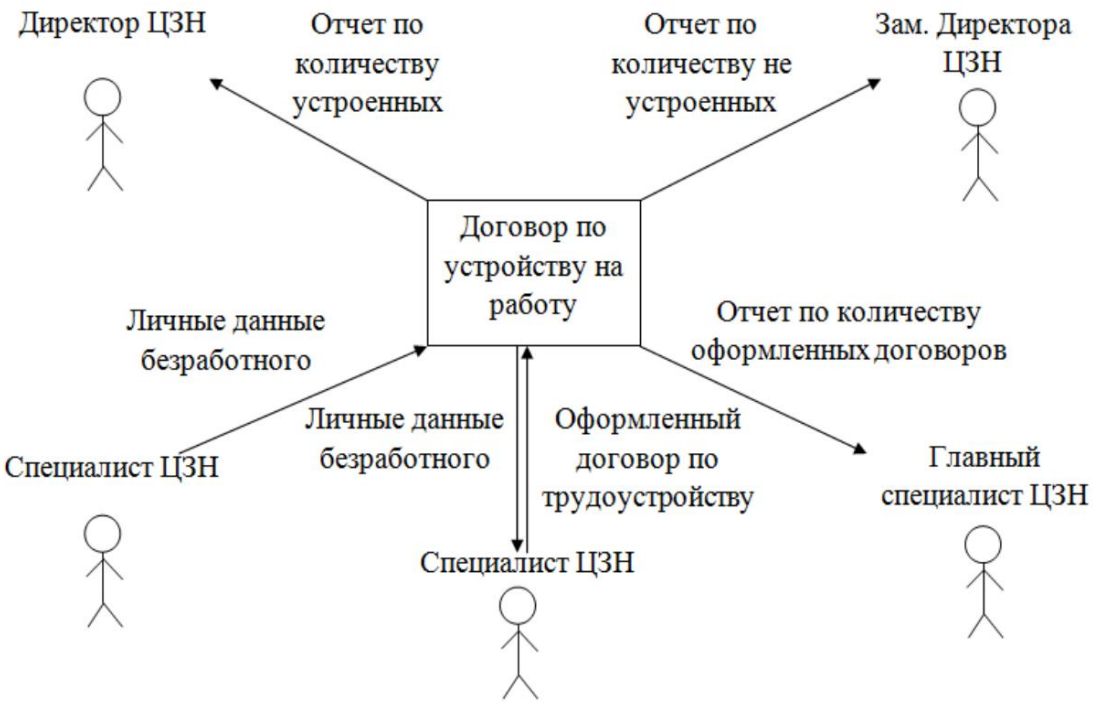

Рисунок 1 – Диаграмма вариантов использования

Специалисты отдела трудоустройства выполняют следующие функции:

– содействуют гражданам в поисках подходящей работы;

Специалисты отдела маркетинга выполняют следующие функции:

– содействуют работодателям в подборе необходимых работников;

– проводят информирование о положении на рынке труда;

1.2. Описание входной и выходной информации

Входной информацией системы является:

а) Данные о вакансиях;

б) Данные о соискателях;

в) Резюме людей;

г) Заявления работодателей;

д) Резюме людей;

Выходной информацией системы является:

а) Обработанные резюме.

б) Списки кандидатов;

Для проведения анализа и реорганизации бизнес - процессов предназначено CASE-средство верхнего уровня AllFusion Process Modeler (BPwin), поддерживающее методологии:

IDEF0 (функциональная модель);

DFD (DataFlow Diagram);

IDEF3 (Workflow Diagram).

Функциональная модель предназначена для описания существующих бизнес - процессов на предприятии (так называемая модель AS-1S) и идеального положения вещей - того, к чему нужно стремиться (модель TO-BE). Методология IDEF0 предписывает построение иерархической системы диаграмм - единичных описаний фрагментов системы.

Построение модели информационной системы начинается с описания функционирования предприятия (системы) в целом в виде контекстной диаграммы. На рисунке представлена контекстная диаграмма информационной системы «Службы занятости».

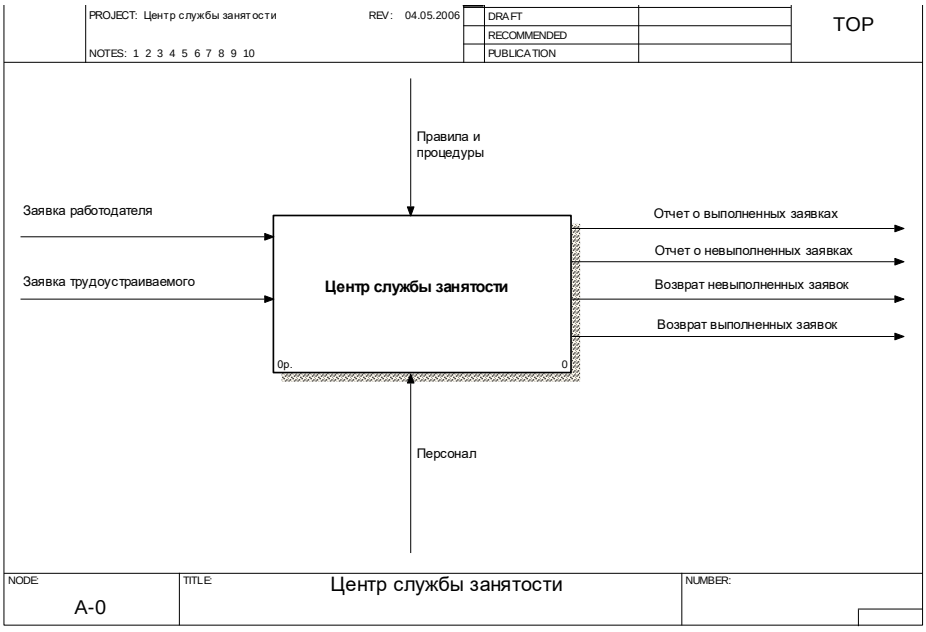

Рисунок 2 - Контекстная диаграмма «Службы занятости».

Взаимодействие системы с окружающей средой описывается в терминах входа (на рис.1 это «Заявка работодателя» и «Заявка трудоустраиваемого»),

управления («Правила и процедуры») и механизмов («Персонал» – это ресурсы, необходимые для процесса функционирования службы занятости).

«Правила и процедуры» – это правила, которыми управляется процесс функционирования службы занятости.

В оказании услуг принимает участие “Персонал” службы занятости.

После описания контекстной диаграммы проводится функциональная декомпозиция - система разбивается на подсистемы и каждая подсистема описывается отдельно (диаграммы декомпозиции). Затем каждая подсистема разбивается на более мелкие и так далее до достижения нужной степени подробности. В результате такого разбиения, каждый фрагмент системы изображается на отдельной диаграмме декомпозиции.

После дальнейшего разбиения диаграммы получаем три диаграммы декомпозиции, описывающие каждая одну из работ, представленных на диаграмме верхнего уровня.

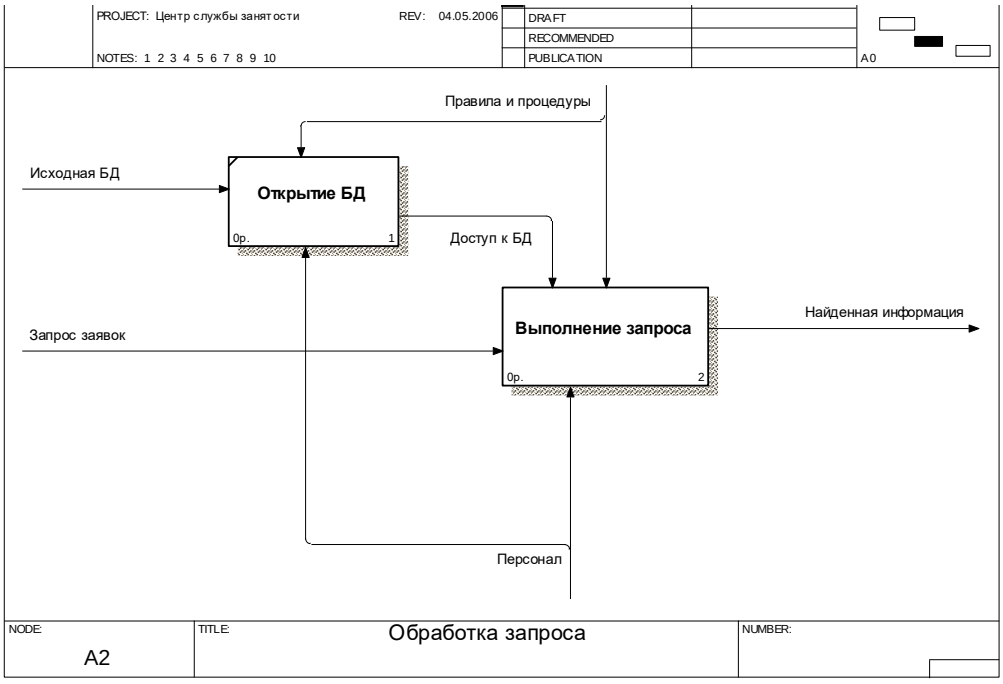

Рисунок 3 - Декомпозиция работы «Обработка запроса».

Если в процессе моделирования нужно осветить специфические стороны технологии предприятия, BPwin позволяет переключиться на любой ветви модели на нотацию IDEF3 или DFD и создать смешанную модель.

Диаграммы потоков данных (DFD) используются для описания документооборота и обработки информации. Нотация DFD включает такие понятия, как "внешняя ссылка" и "хранилище данных", что делает ее более удобной (по сравнению с IDEF0) для моделирования документооборота.

На рис. 5 представлена «Декомпозиция в нотации DFD «Выполнение запроса», описывающая деятельность по поиску информации в базе данных.

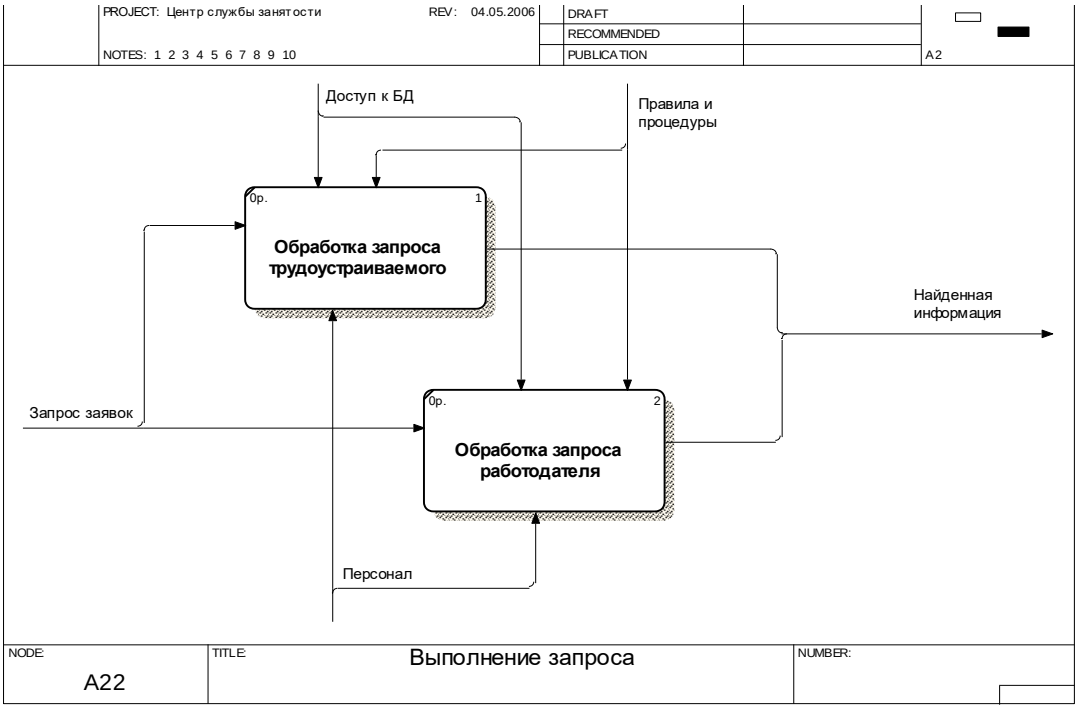

Рисунок 4 - Декомпозиции в нотации DFD «Выполнение запроса».

Все работы, представленные на диаграмме выполняются «Персоналом» в соответствие с перечнем обязанностей.

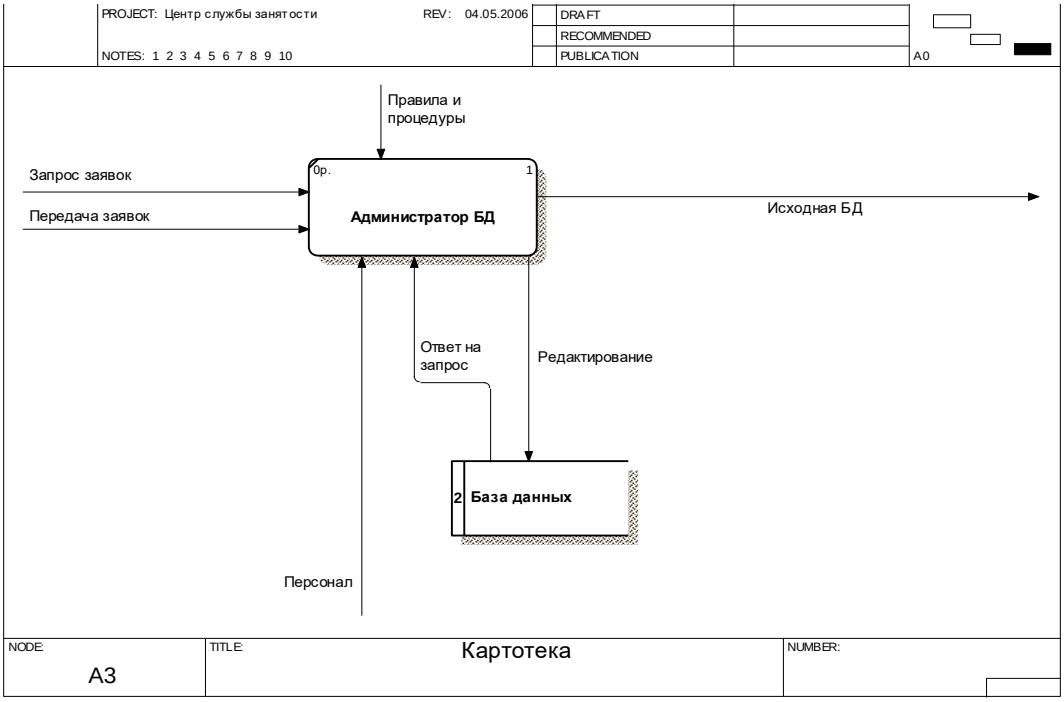

Рисунок 5 - Декомпозиции в нотации DFD «Картотека».

1.3. Постановка задачи

В современном мире в связи с большим количеством различных предприятий, их расширением, существуют проблемы подбора и управления персоналом. Большинству компаний на своем жизненном пути рано или поздно приходится встречаться с центрами занятости.

Услуги центра занятости могут понадобиться компании тогда, когда проводится массовый набор линейного персонала, сопровождаемый обработкой большого объема информации и большим количеством встреч и интервью. В этом случае, экономя ресурсы внутренних сотрудников, можно передать всю «черновую» работу рекрутерам агентства.

Другой ситуацией, когда наличие агентства в цепочке поиска и подбора персонала представляется необходимым, это организация подбора персонала в других регионах или странах. Сетевые рекрутинговые агентства, имеющие свои представительства или партнеров в интересующем регионе, несомненно, лучше знают местный рынок труда и могут провести отбор более квалифицированно. Одновременно снимаются издержки на организацию поездок сотрудников в командировки с целью поиска персонала.

Еще одним аргументом для привлечения к работе сотрудников рекрутингового агентства может быть подбор персонала в новых для компании сферах бизнеса. В этой ситуации, менеджер по персоналу или руководитель подразделения, вряд ли сможет квалифицированно оценить потенциал кандидата и его профессиональную успешность. Хотя бы потому, что нет «меры сравнения» внутри компании.

Типичной ситуацией, стимулирующей компании прибегать к услугам центра занятости, является ситуация цейтнота. Если, как выражаются менеджеры, «заказ горит», передача работ в агентство представляется единственно возможным выходом.

Ситуацией требующей безусловного привлечения сторонних экспертов и консультантов, является поиск кандидата на должность руководителя (топ менеджера) компании. И дело не только в том, что внутренний рекрутер не в состоянии грамотно оценить его потенциал, а скорее в том, что его мнение не является авторитетом для руководства компании. А выбирать (отбирать) кандидатуру будущего начальника, для подчиненного представляется не совсем этичным. Традиционно для решения таких задач руководство компании обращается к услугам консультантов executivesearch - высшего эшелона кадрового бизнеса.

И, наконец, серьезным основанием для привлечения центра занятости может стать слабость внутреннего отдела персонала. Ведь если в нем работает полтора человека, то нелепо ожидать, что помимо выполнения текущих задач они с должной эффективностью смогут выполнить работу, за которую в агентстве отвечает целая команда.

С точки зрения отраслевой принадлежности, центр занятости относится к нематериальному производству, так как занимается предоставлением услуг по трудоустройству населения, что не является материальным производством.

Тип и характер производства определить нельзя, потому что оказание услуг не имеет четкого технологического процесса, но можно предположить, что из всех имеющихся вариантов наиболее подходящим для нашего агентства является

дискретное производство, так его можно прервать в любой момент времени, и это не повлечет за собой никаких существенных изменений, кроме потери прибыли.

Технологический процесс: четко не существует, так как предприятие не производит материальную продукцию, но можно проследить цепочку, по которой осуществляется деятельность нашего центра занятости:

- соискатель заполняет анкету по трудоустройству;
- данная анкета отправляется в отдел обработки данных, где подыскиваются вакансии, соответствующие данным анкеты соискателя из отдела обработки данных анкета отправляется работодателю;
- если работодатель дает положительный ответ, то агентство связывается с соискателем и отправляет его в юридический отдел для составления договора;
- после составления договора соискателю предоставляются данные работодателя;
- проводится собеседование между соискателем и работодателем.

Требования, которые она предъявляет к программным средствам:

1. Функциональные требования.
2. Требования к удобству использования.
3. Требования к надежности.
4. Требования к производительности.
5. Требования к поддержке.

Это базовые требования, накладываемые на информационную систему используемой методологией разработки. Однако, помимо них, далее будут определены дополнительные требования, накладываемые в связи со спецификой рассматриваемых процессов и их исполнителей.

Требования к разработке базы данных и приложению:

- Проведение поэтапного проектирования базы данных.
- База данных должна отражать всю информацию о предметной области.

- Целостность базы данных (полнота и непротиворечивость).
- Многократное использование данных.
- Быстрый поиск и получение информации по запросам пользователей.
- Простота обновления данных.
- Уменьшение излишней избыточности данных.
- Защита данных от несанкционированного доступа, от искажения и уничтожения.
- Реализация отдельной формы с информацией о программе и разработчике.
- Обработка ввода данных от пользователя.
- Обработка исключений при работе с программой.
- Разработать не менее 6 таблиц в БД.
- По необходимости должны быть реализованы подсказки.

Требования к программному обеспечению:

- Стабильная работа без ошибок
- Интуитивно понятный интерфейс
- Все объекты системы должны быть логически сгруппированы.

Минимальные системные требования к аппаратному и программному обеспечениям:

- Процессор Intel Core I3-540 или выше;
- Свободная оперативная память не менее 4 Гб;
- Накопитель на жестком диске со свободным объемом памяти не менее 100 МБ;
- Клавиатура;
- Манипулятор «мышь»;
- Наличие одной из Windows:7/8/10;
- Наличие .Net Framework 4.7.2 и более поздней версии.

2. Проектная часть

2.1. Описание средств разработки

В качестве среды разработки выбран Visual Studio 2017.

Microsoft Visual Studio — линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms.

В отличие от своих конкурентов, Visual Studio имеет ряд очень полезных функций:

- Технология автодополнения текста;
- Функционал, который помогает найти ошибки в коде;
- Встроенный менеджер тестов;
- Менеджер плагинов, адаптеров, провайдеров;
- Система управления пакетами для платформ разработки Microsoft;
- Встроенный менеджер контроля версий;
- Архиватор проектов;

Возможность изменить внешний вид Visual Studio под себя.

В настоящее время C# чаще всего используются для создания коммерческих и бизнес-приложений. Этот языки подходят для большинства программистов, и в данной работе был сделан упор на него. Весомыми плюсами данного языка программирования, которые выделяют его среди других языков:

- Простой и лаконичный код;
- Большое количество библиотек, которые уже имеются в базе;
- Удобная отладка;
- Простая сборка проектов;
- Гибкость;
- Высокая структурированность.

Из-за высокой совместимости с Windows и с выбранной средой разработки, в работе в качество СУБД выбран Microsoft SQL Server.

Microsoft SQL Server — это РСУБД, которая разработана специалистами из Microsoft. В качестве основного языка запросов используется Transact-SQL. При этом Transact-SQL — это реализация стандарта ANSI/ISO по SQL, но имеющая некоторые расширения. Сегодня СУБД MS SQL широко применяется при работе с базами данных самых разных размеров, начиная от персональных и заканчивая крупными БД масштаба предприятия.

Основные достоинства:

- Переносимость;
- Наличие дополнительных возможностей;
- Масштабируемость;
- Производительность;
- Доступность;
- Безопасность;

Для выполнения практической части работы были применены следующие дополнительные средства и инструменты:

UML – язык для определения, визуализации, конструирования и документирования программных систем;

CASE-система для построения моделей системы – Erwin Data Modeler.

CASE-система для построения моделей бизнес-процессов – BPwin.

2.2. Проекирование и разработка баз данных

2.2.1. Проектирование концептуальной модели БД

Концептуальная модель базы данных – наглядная схема предметной области, для которой создается база данных. Она создается для дальнейшего проектирования базы данных и ее перевода в реляционную базу данных. На данной модели прописываются связи между объектами данных и их характеристиками в визуально удобном виде. Концептуальная модель является

как образом реальности, так и образом проектируемой базы данных для этой реальности.

При построении ER-модели были выделены следующие сущности, отношения и атрибуты (Рисунок 1):

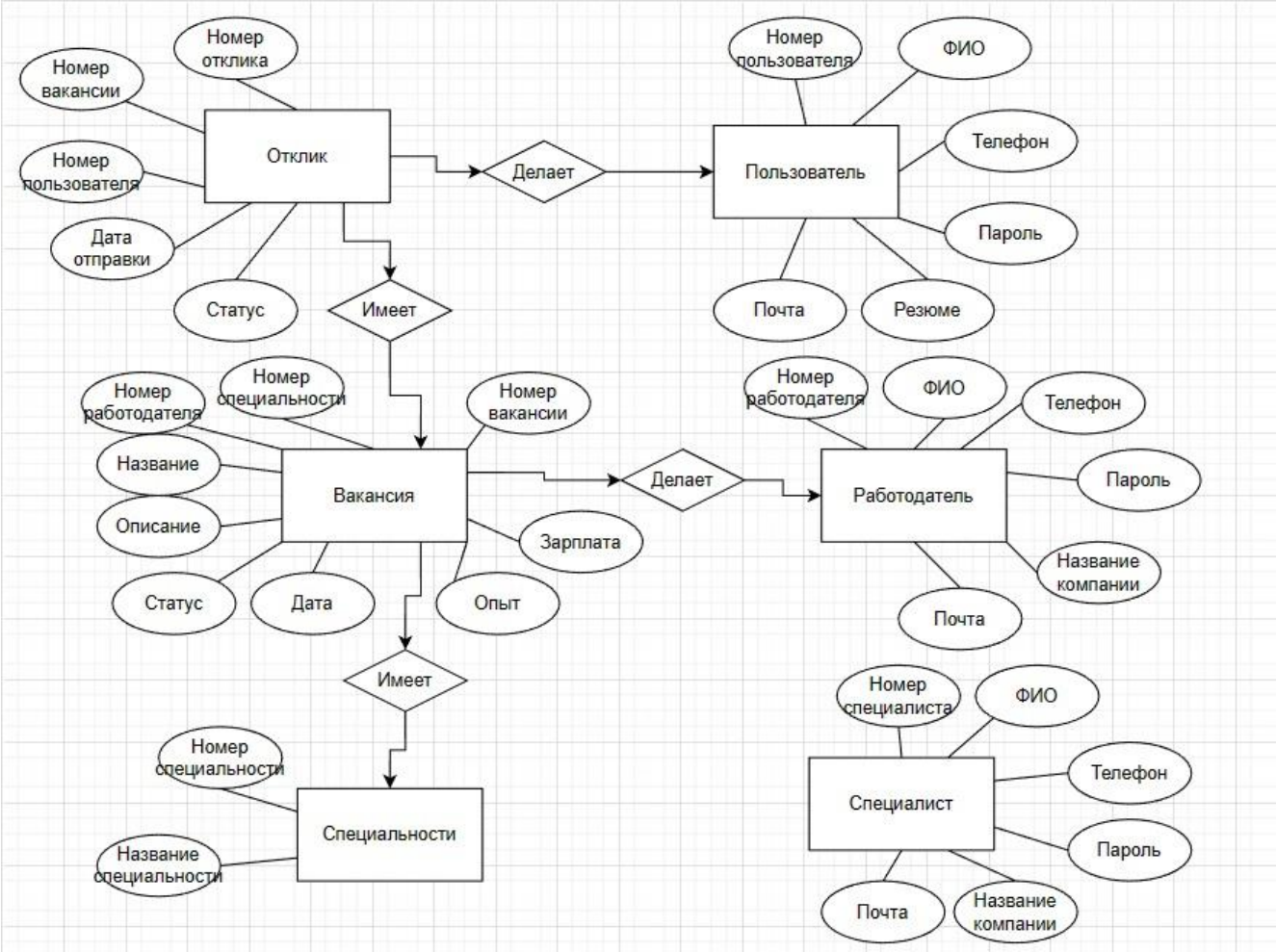

Рисунок 6 - «Концептуальная модель БД»

2.2.2. Проектирование логической модели БД

Логическая модель – схема базы данных на основе конкретной модели данных (в данном случае – на основе концептуальной модели). В нее входит набор схем отношений с указанием первичных ключей, связей между отношениями (первичными ключами), а также приведение к третьей нормальной форме. На данном этапе учитывается специфика конкретной модели данных и предметной области, но может не учитываться специфика конкретной СУБД. Целью построения такой модели является получение графического представления логической структуры исследуемой предметной области.

Результатом этапа логического проектирования является модель, изображенная на Рисунке 7, созданная с помощью средства ERwin.

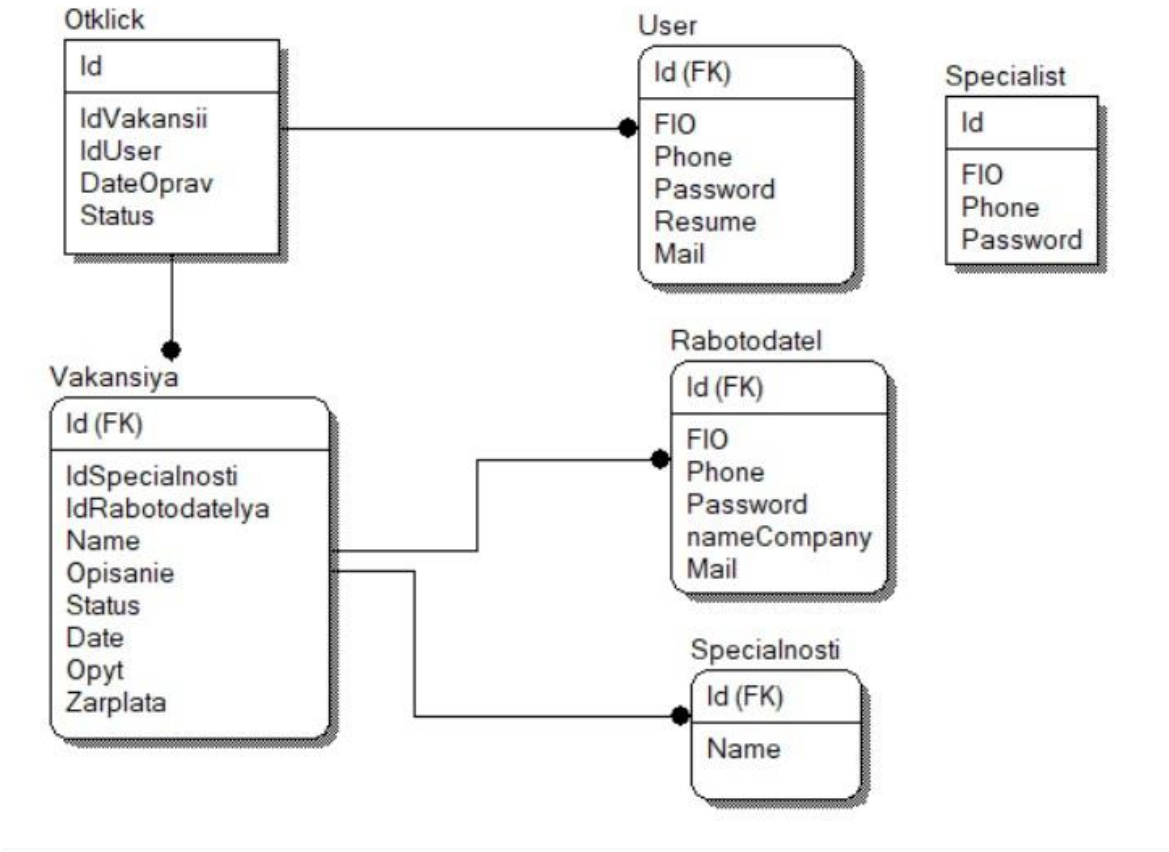

Рисунок 7 - «Логическая модель БД»

2.2.3. Проектирование физической модели БД

Физическая модель данных строится на основе логической модели, путем создания отдельной таблицы для каждой сущности. Связи между таблицами создаются так же, как на логическом уровне.

Физическая модель данных зависит от конкретной СУБД, фактически являясь отображением системного каталога. В физической модели содержится информация обо всех объектах базы дынных. Поскольку стандартов на объекты базы данных не существует, физическая модель зависит от конкретной реализации СУБД.

Результатами стадии разработки концептуальной модели данных является структура проектируемой ИС, концептуальная схема базы данных: логическая и физическая модели данных предметной области. Физическая модель генерируется

в СУБД, где создается база данных с названиями полей, таблицы, которой не содержат записей.

Так как логическая модель данных уже была создана, то для просмотра физической модели можно воспользоваться списком выбора, расположенным в правой части панели инструментов ERwin (вместо Logical выбрать Phisical).

Физическая модель БД определяет способ размещения данных в среде хранения и способы доступа к этим данным, которые поддерживаются на физическом уровне.

Физическая модель базы данных содержит все детали, необходимые конкретной СУБД для создания базы: наименования таблиц и столбцов, типы данных, определения первичных и внешних ключей и т. п.

На данном этапе необходимо определиться с типами данных полей и проверочными ограничениями.

Для построения физической модели базы данных использовалась та же программа «ERwin», что и для построения логической.

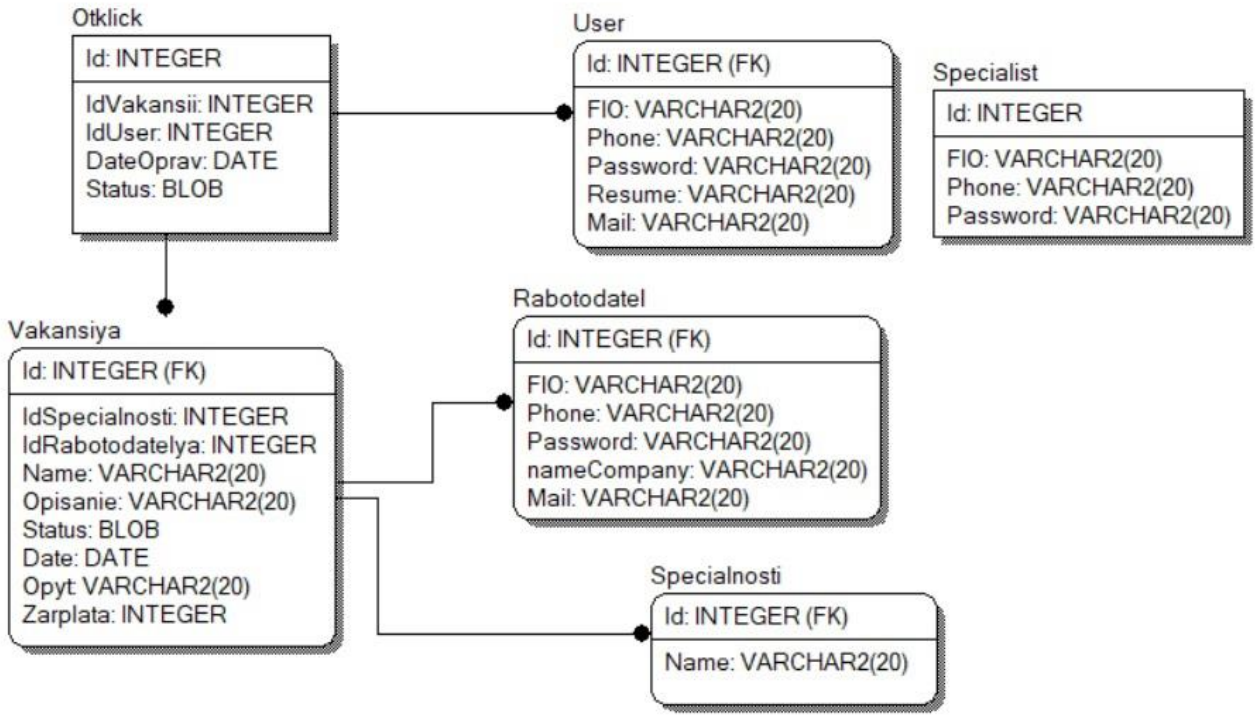

Рисунок 8 - «Физическая модель БД»

2.2.4. Реализация проекта в среде конкретной СУБД

Реализация проекта ИС для центра занятости в среде конкретной СУБД основана на представленных ранее логической и физической моделях. Среда конкретной СУБД, выбранная для реализации - Microsoft SQL Server Management Studio.

Так как уже созданы три таблицы модели базы данных, можно переходить к созданию самой базы данных. Первым делом для создания базы данных в Microsoft SQL Server нужно создать подключение к серверу, на котором мы уже и будем работать (рисунок 9).

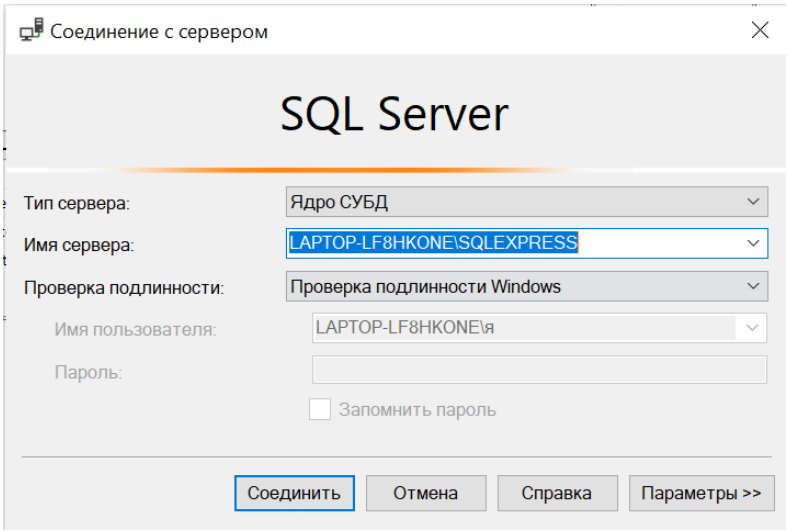

Рисунок 9 - «Подключение к серверу»

После подключения переходим к созданию базы данных, это можно сделать через запрос написав Create Database, либо через обычное создание путем нажатия кнопки New Database (рисунок 10). Данная база будет называться «EmploymentCenter».

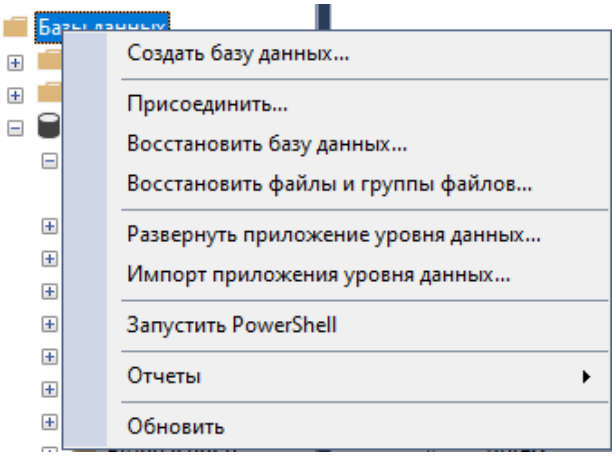

Рисунок 10 - «Контекстное меню»

Когда создали базу, то можно переходить к созданию таблиц в ней. Таблицы также можно сделать через запрос, написав Create Table.

С помощью встроенного окна запросов были созданы таблицы, входящие в базу данных. Скрипт добавления таблиц расположен в Приложении. После его выполнения была составлена диаграмма базы данных, представленная на Рисунке 11. На ней изображены все таблицы базы данных, их связь, зависимость и отмечены первичные ключи. При создании данной диаграммы использовались встроенные средства выбранной среды серверной части.

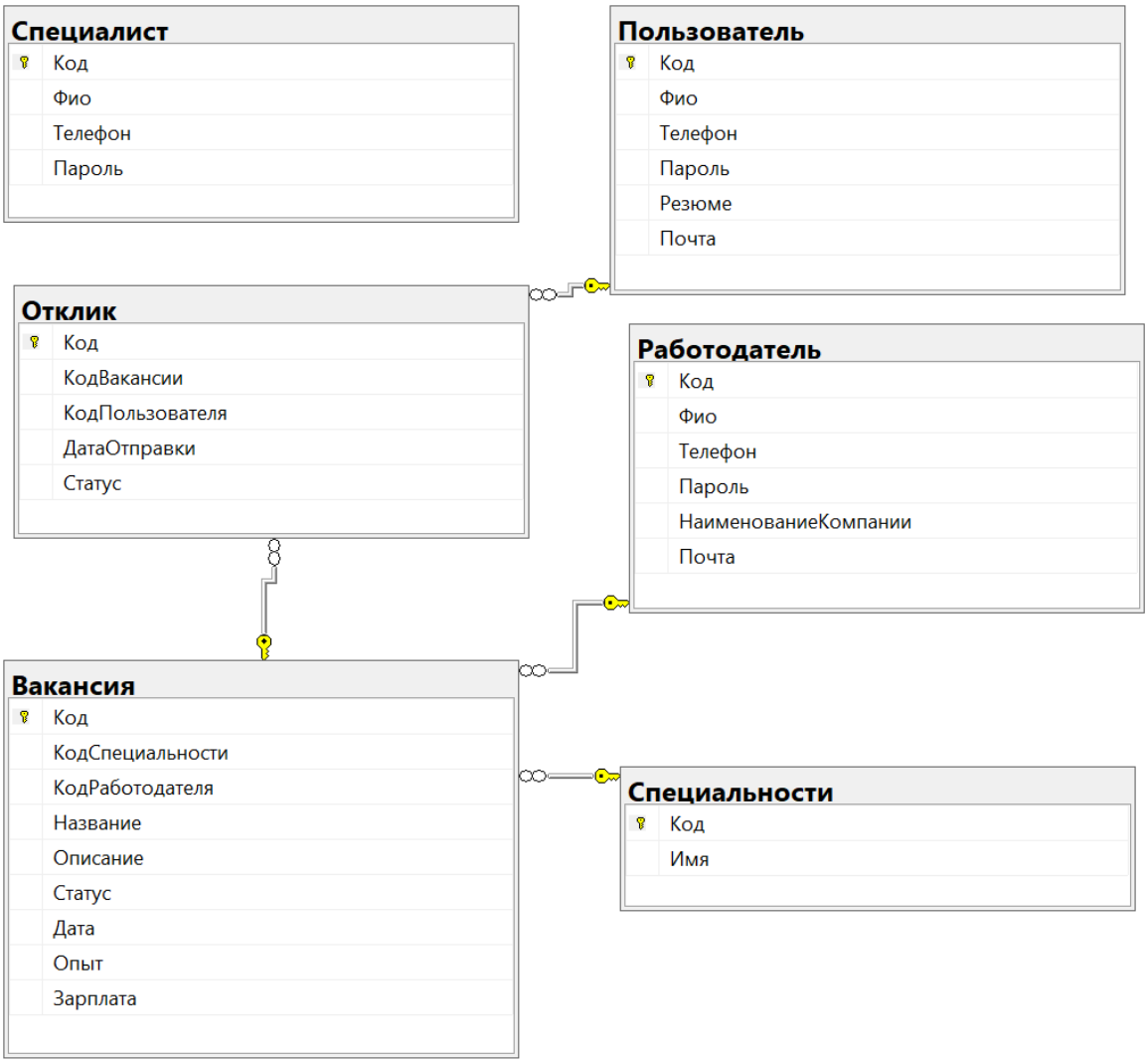

Рисунок 11 - «Диаграмма БД»

2.3. Создание клиентской части информационной системы

После постройки базы данных в Microsoft SQL Server переходим к следующему этапу, а именно, к построению программного приложения в Visual Studio 2019, но при создании приложения нужно помнить, что программа должна быть информативной и простой в использовании.

Первое с чего следует начать – это с того, что нужно создать проект и связать его с Microsoft SQL Server, чтобы можно было пользоваться БД.

Для этого необходимо подключиться к базе данных. В «Обозреватель решений» кликаем правой кнопкой мыши по проекту, нам откроется контекстное

меню в котором нам нужно выбрать пункт «Добавить» и далее «Создать элемент…», там выбрать ADO.net (рисунок 12)

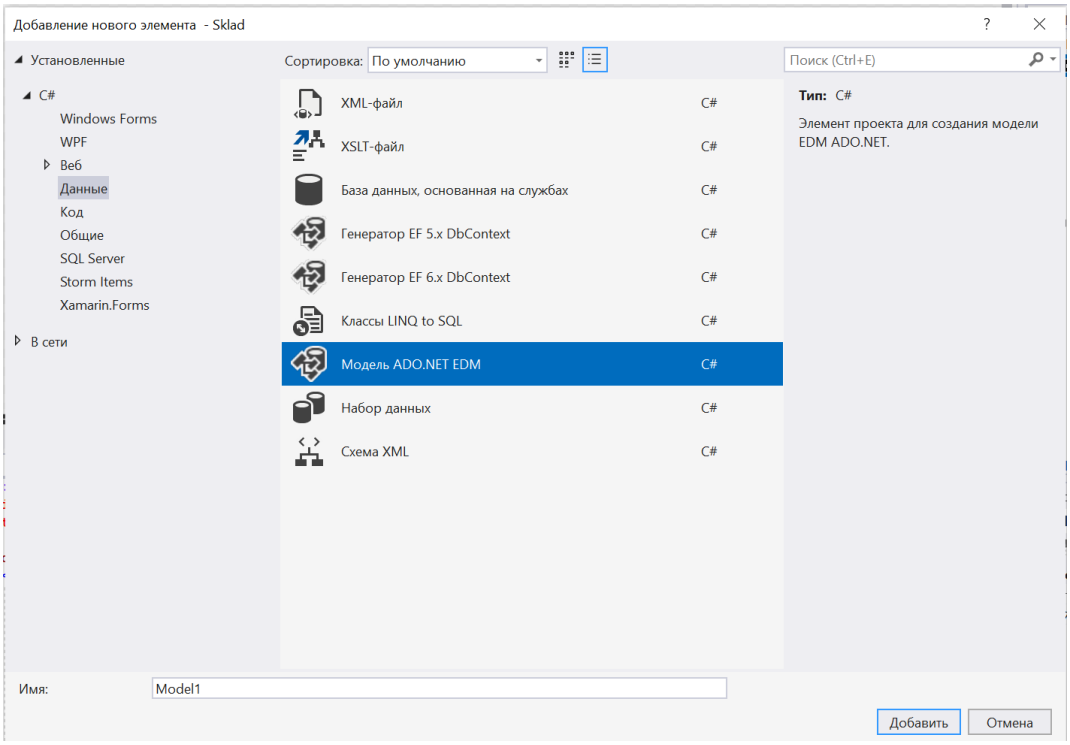

Рисунок 12 - Подключение ADO.NET

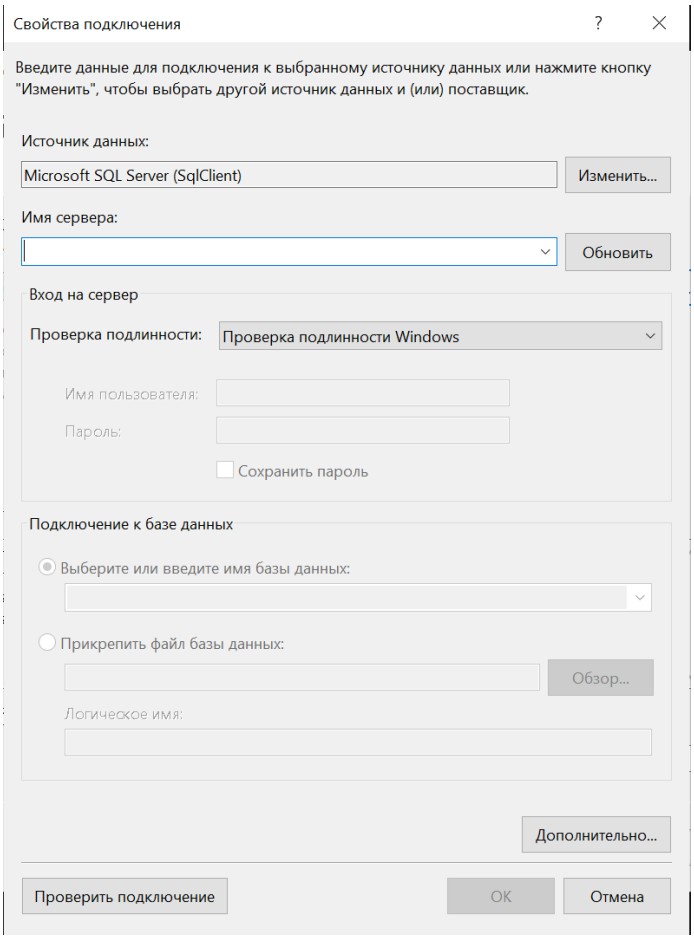

Рисунок 13 - «Окно добавления подключения»

После всех действий нужно проверить подключилась ли база данных, для этого нужно нажать на кнопку «Проверить подключение» и у вас на экране появиться окно в котором вам сообщат выполнена проверка подключения или нет (рисунок 14).

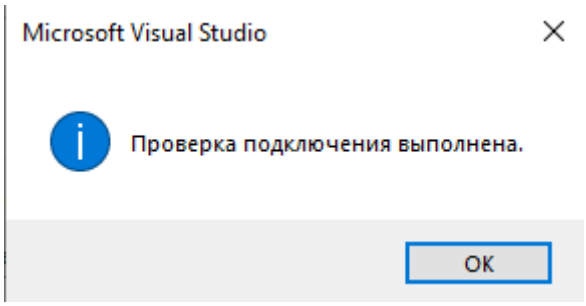

Рисунок 14 - «Окно проверки подключения»

Сначала мы в «Обозреватель решений» кликаем правой кнопкой мыши по проекту, нам откроется контекстное меню в котором нам нужно выбрать пункт «Добавить» и далее «Создать элемент…».

После того, как выполнили соединение, переходим к созданию страниц и заполнения их инструментами, а после и кодом. Для начала нужно создать страницы:

1. LoginPage, MyResponsePage, MyVacancyPage, NewVacancyPage, PersonalDataPage, RegUserPage, ResponseWorkersPage, SelectedVacancyPage, VacancyAddEditPage, VacancyList.

Следующим шагом нужно добавить элементы на эти страницы. Для этого приложения потребуются следующие элементы:

1. Label; MaterialRaisedButton; TextBox; PictureBox; DataGridView; ComboBox;

Label - предназначен для создания подписей к другим элементам управления или для вывода сообщений прямо на поверхности формы.

MaterialRaisedButton – кнопка, которая позволяет пользователю щелкнуть на неё для выполнения действия.

На элементе управления MaterialRaisedButton могут отображаться текст и изображения. При щелчке кнопки мышью элемент управления выглядит так, как будто его нажимают и отпускают.

TextBox – используется для ввода и редактирования текста.

DataGridView — это элемент управления, позволяющий отображать данные в табличном формате, в настраиваемой сетке.

ComboBox – позволяет занести в него коллекцию данных и выбрать их.

GroupBox – элемент позволяющий добиться чтобы элементы визуально выглядели сгруппировано.

После того как все инструменты были добавлены, размещены и подписаны, нужно переходить к написанию кода.

Далее спроектируем пользовательский интерфейс. Пользовательский интерфейс - это средство управления прикладной программой, с помощью которого происходит взаимодействие между пользователем и компьютером. Основная его цель - минимизировать общую информацию на экране и представить только то, что является необходимым для пользователя. С помощью интерфейса пользователь управляет работой программы формирует управленческие запросы и получает информацию о ходе работы программы.

Приложение является многооконным. При запуске появляется окно авторизации, из которого можно перейти к основным функциям программы.

Взаимодействие клиентского приложения и базы данных производилось с помощью разработанных страниц, позволяющих отображать таблицы и представления, корректировать отображаемые данные и вводить новые данные, а также удалять строки в таблицах и производить моментальный поиск.

С помощью специально подготовленных тестовых данных проверена заданная функциональность системы и подтверждена ее готовность работать в реальных условиях эксплуатации. Вместе с тем благодаря наличию сценариев на языке Transact-SQL, исходных модулей на языке C# и большим возможностям среды разработки всегда можно усовершенствовать как базу данных, так и клиентский интерфейс созданной информационной системы.

2.4. Тестирование информационной системы

Тестирование систем – важный этап производства ПО, направленный на детальное исследование программного кода и выявление ошибок в работе систем. Одна из главных целей тестирования – проверка соответствия работоспособности системы в целом или ее отдельных модулей ожиданиям заказчика. Тестирование ПО позволяет своевременно выявить дефекты кода, повысить надежность и отказоустойчивость системы, избежать финансовых и репутационных потерь, связанных с нестабильной работой готового решения.

Модульное тестирование, или юнит-тестирование (англ. unit testing) — процесс в программировании, позволяющий проверить на корректность отдельные модули исходного кода программы.

Каждый тест должен проверять только одну вещь. Если процесс слишком сложен (например, покупка в интернет магазине), разделите его на несколько частей и протестируйте их отдельно. Если вы не будете придерживаться этого правила, ваши тесты станут нечитаемыми, и вскоре вам окажется очень сложно их поддерживать.

2.5. Инструкции по эксплуатации информационной системы

1. Пользователями информационных систем (далее - ИС) являются сотрудники центра занятости населения пгт Михнево, в установленном порядке допущенные к работе в ИС.

2. Настоящая инструкция определяет обязанности, права и ответственность пользователей, допущенных к работе в ИС, в области обеспечения безопасности конфиденциальной информации, в том числе персональных данных (далее - ПДн).

3. При эксплуатации ИС пользователь обязан:

а) соблюдать конфиденциальность при работе с информацией в ИС;

б) руководствоваться требованиями настоящей инструкции, а также иных документов центра занятости населения пгт Михнево, регламентирующих обработку и защиту конфиденциальной информации, в том числе ПДн:

в) помнить свои личные пароли и идентификаторы;

г) руководствоваться требованиями инструкций по эксплуатации установленных средств вычислительной техники и средств защиты информации (далее - СЗИ); д) блокировать ввод-вывод информации на своем автоматизированном рабочем месте ИС (далее - АРМ) перед оставлением своего рабочего места (перерыва в работе) или выключать АРМ;

е) блокировать вывод информации на монитор АРМ при выходе в течение рабочего дня из помещения, в котором размещается ИС.

4. При эксплуатации ИС пользователю запрещается:

а) самостоятельно подключать к АРМ нештатные устройства;

б) самостоятельно вносить изменения в состав, конфигурацию и размещение АРМ;

в) самостоятельно вносить изменения в состав, конфигурацию и настройку программного обеспечения, установленного в АРМ;

г) самостоятельно вносить изменения в размещение, состав и настройку СЗИ;

д) сообщать устно, письменно или иным способом другим лицам пароли, передавать личные идентификаторы, ключевые дискеты и другие реквизиты доступа к ресурсам ИС.

5. Пользователь ИС имеет право:

а) обращаться к ответственному за обеспечение безопасности ПДн по вопросам защиты обрабатываемой в ИС информации и эксплуатации установленных СЗИ, а также с просьбой об оказании технической и методической помощи по использованию установленных программных и технических средств ИС;

б) обращаться к ответственному за организацию обработки ПДн по вопросам, связанным с выполнением требований законодательства РФ в области защиты конфиденциальной информации, в том числе ПДн.

6. Пользователь несет персональную ответственность за соблюдение требований законодательства РФ и документов центра занятости населения пгт

Михнево, определяющих порядок обработки и защиты конфиденциальной информации, в том числе ПДн.

2.5.1. Руководство по установке информационной системы

Для запуска и корректной работы программы, она должна быть запущена в одной папке с файлами: «EmploymentCenter.exe», «EmploymentCenter.sql». Где файл «EmploymentCenter.sql» содержит базу данных программы.

При запуске файла «EmploymentCenter.exe», откроется главное окно программы, после чего можно приступить к работе.

Для входа пользователь должен авторизоваться под своими личными данными. В диалоговое окно вводятся имя пользователя и пароль.

Для создания новой записи необходимо заполнить поля конкретными данными. Если в описанные выше поля будут внесены некорректные данные, то выведется на форму сообщение с текстом данной ошибки.

Для того, чтобы внести данные в таблицу необходимо нажать кнопку «Добавить запись».

Для того, чтобы отредактировать запись необходимо нажать кнопку «Режим редактирования», изменить в полях запись и сохранить редактированную запись нажав кнопку «Изменить».

Для удаления данных из таблицы используется кнопка «Удалить».

2.5.2. Руководство пользователя

Пользовательский интерфейс является своеобразным коммуникационным каналом, по которому осуществляется взаимодействие пользователя и компьютера.

Лучший пользовательский интерфейс – это такой интерфейс, которому пользователь не должен уделять много внимания и практически не замечать его. Такой интерфейс называют прозрачным – пользователь как бы смотрит сквозь него на свою работу.

Интерфейс системы должен быть понятным и удобным, не должен быть перегружен графическими элементами. Навигационные элементы должны быть выполнены в удобной для пользователя форме. Ввод-вывод данных системы, прием управляющих команд и отображение результатов их исполнения должны выполняться в интерактивном режиме. Интерфейс должен соответствовать современным эргономическим требованиям и обеспечивать удобный доступ к основным функциям и операциям системы.

Экранные формы должны проектироваться с учетом требований унификации:

– все экранные формы пользовательского интерфейса должны быть выполнены в едином графическом дизайне, с одинаковым расположением основных элементов управления и навигации;

– для обозначения сходных операций должны использоваться сходные графические значки, кнопки и другие управляющие (навигационные) элементы;

– внешнее поведение сходных элементов интерфейса (реакция на наведение указателя «мыши», переключение фокуса, нажатие кнопки) должны реализовываться одинаково для однотипных элементов.

Интерфейс программного продукта является простым и интуитивно понятным. Доступность командных кнопок определяет последовательность работы с объектами формы, набор проверок, используемых в программе, обеспечивает целостность данных.

При запуске программы открывается форма авторизации (рисунок 12), содержащая поля для ввода логина и пароля. В случае неверного ввода пароля на форме авторизации появится соответствующее сообщение.

При запуске программы будет запущено меню авторизации на котором можно войти в систему.

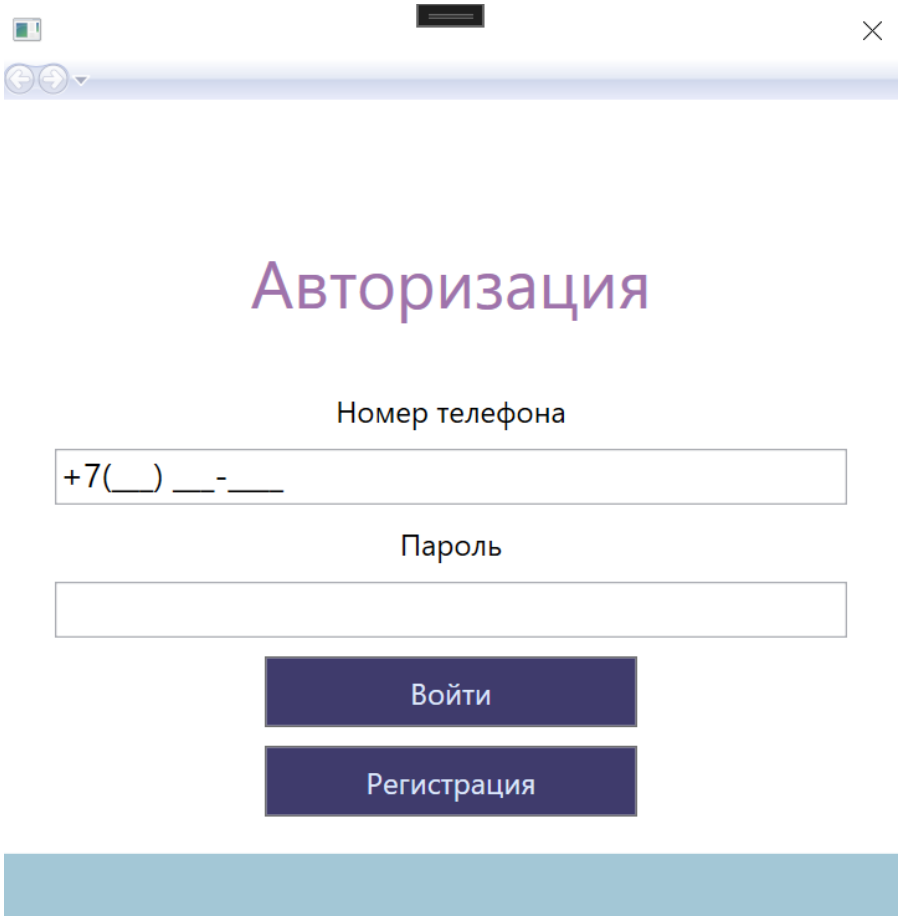

Рисунок 15 - Окно авторизации

Работодатель может добавить вакансию.

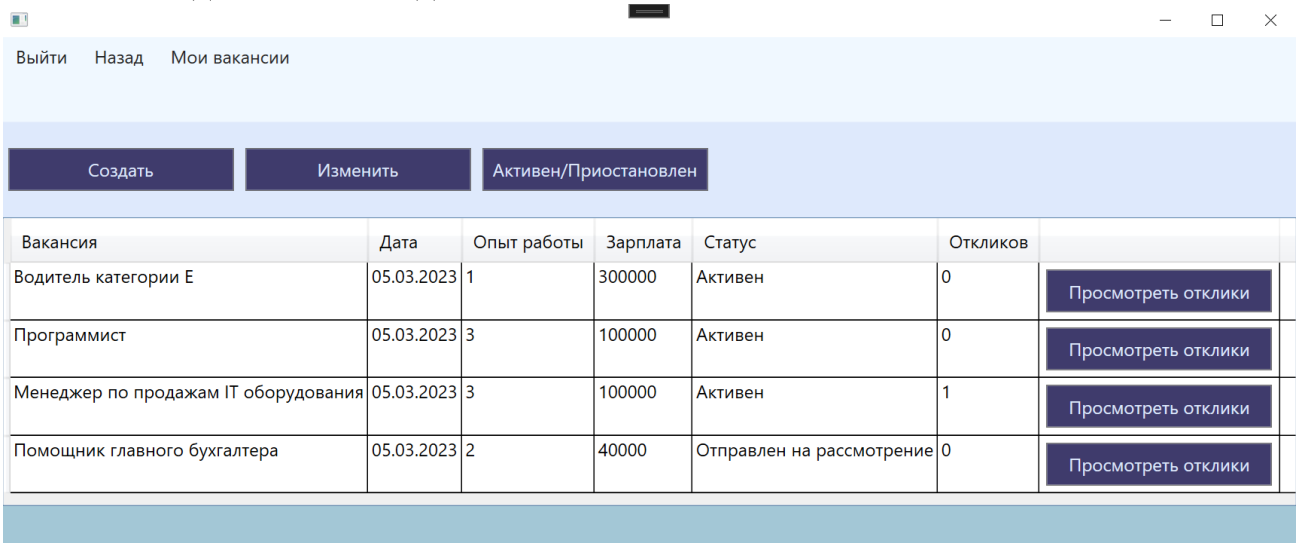

Рисунок 16 - Окно работодателя

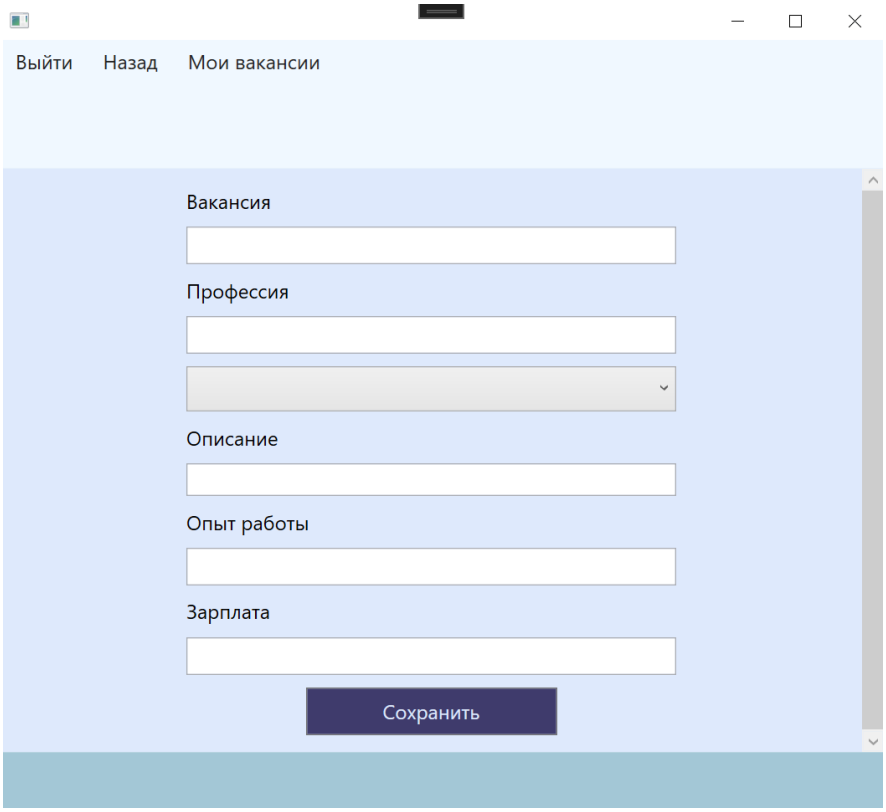

Рисунок 17 - Создание новой вакансии

Также можно войти под специалистом, который рассматривает новые вакансии. И может редактировать общий список вакансий.

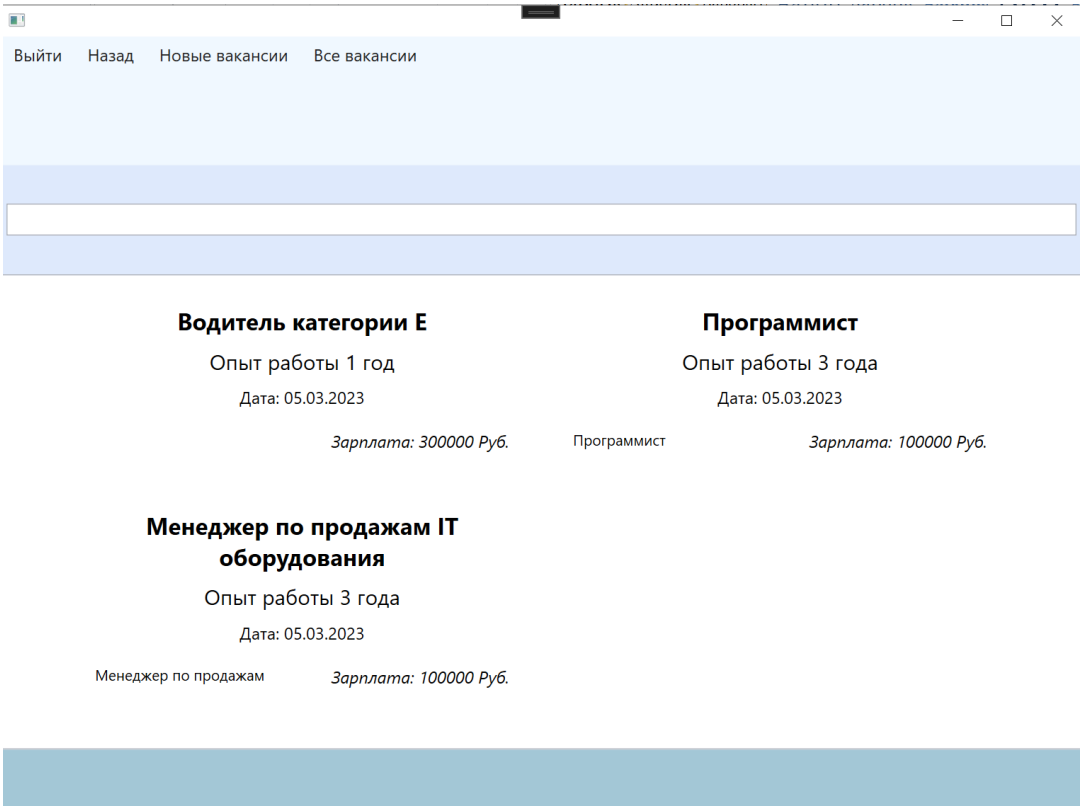

Рисунок 18 - Окно специалиста

Специалист должен одобрить новую вакансию, и только после этого ее увидит соискатель.

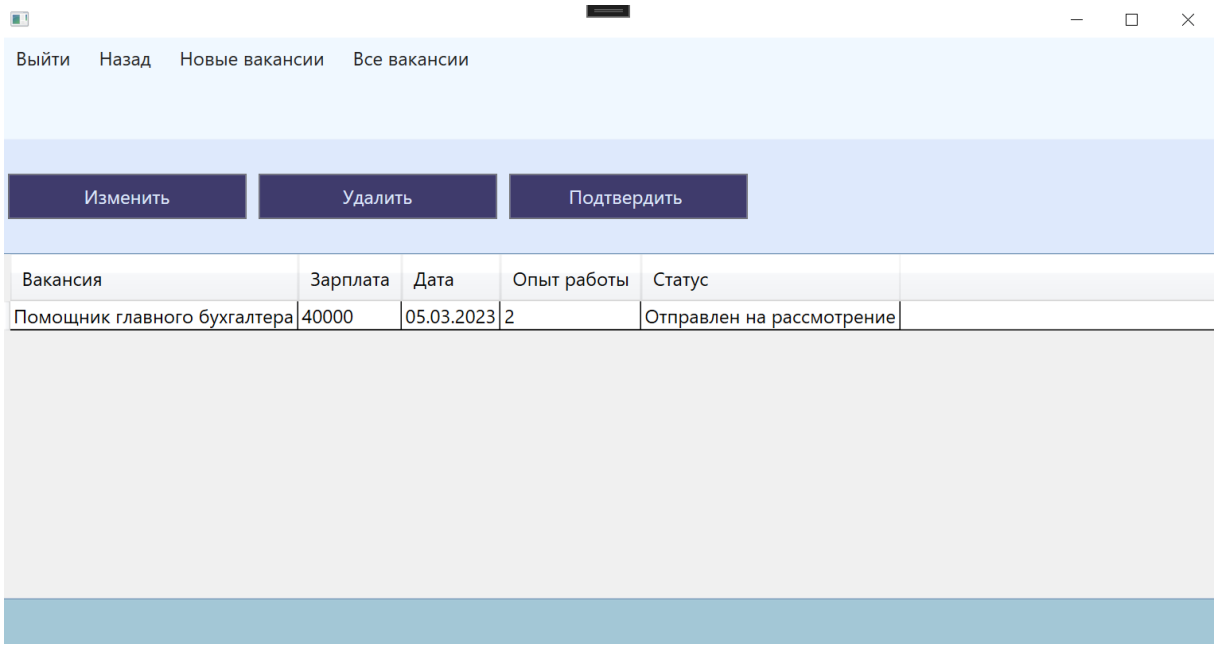

Рисунок 19 - Окно "Новые вакансии"

Данные о соискателе в окне «о себе»

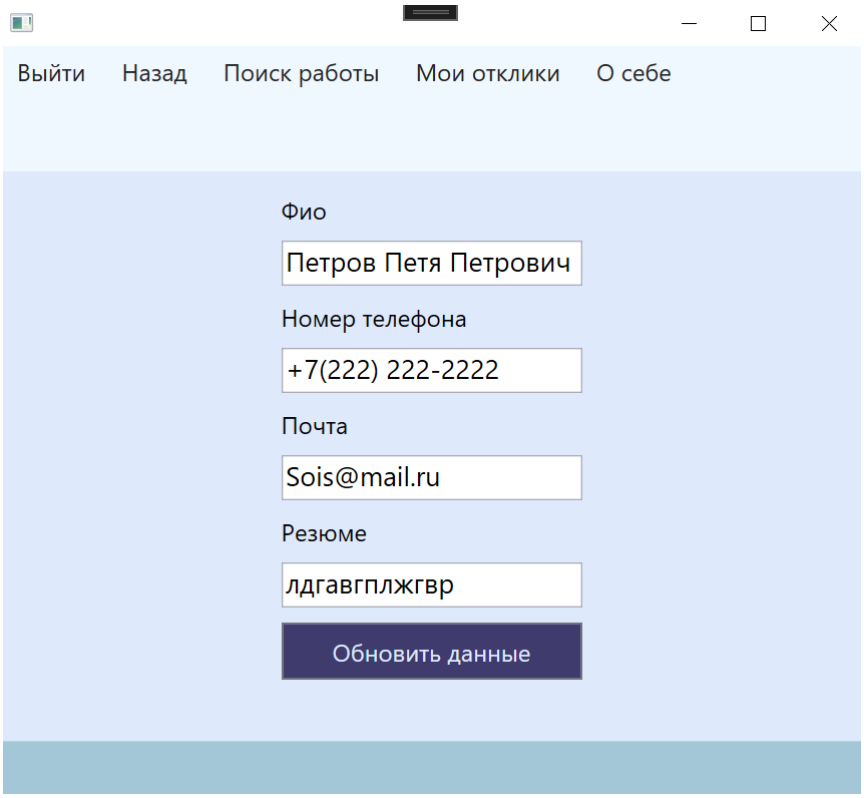

Рисунок 20 - "О себе"

Соискатель войдя под своим аккаунтом может найти себе подходящую вакансию.

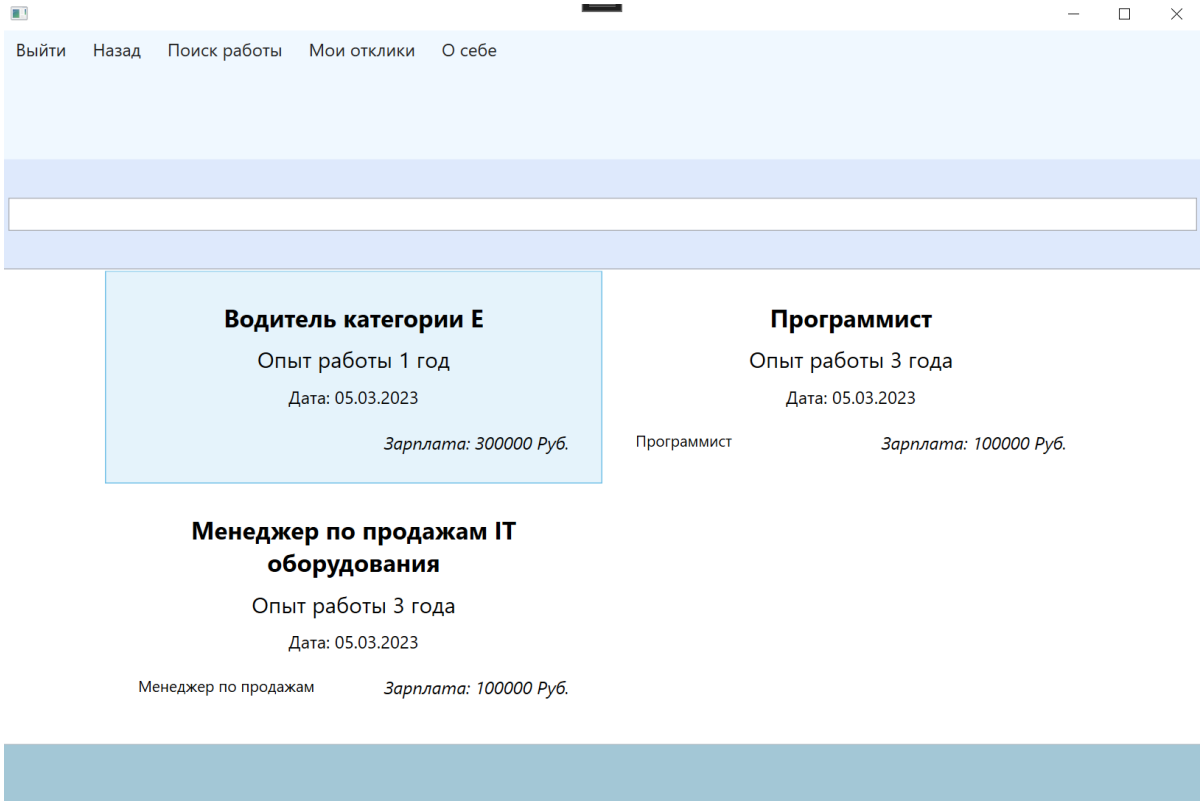

Рисунок 21 - Окно поиск работы

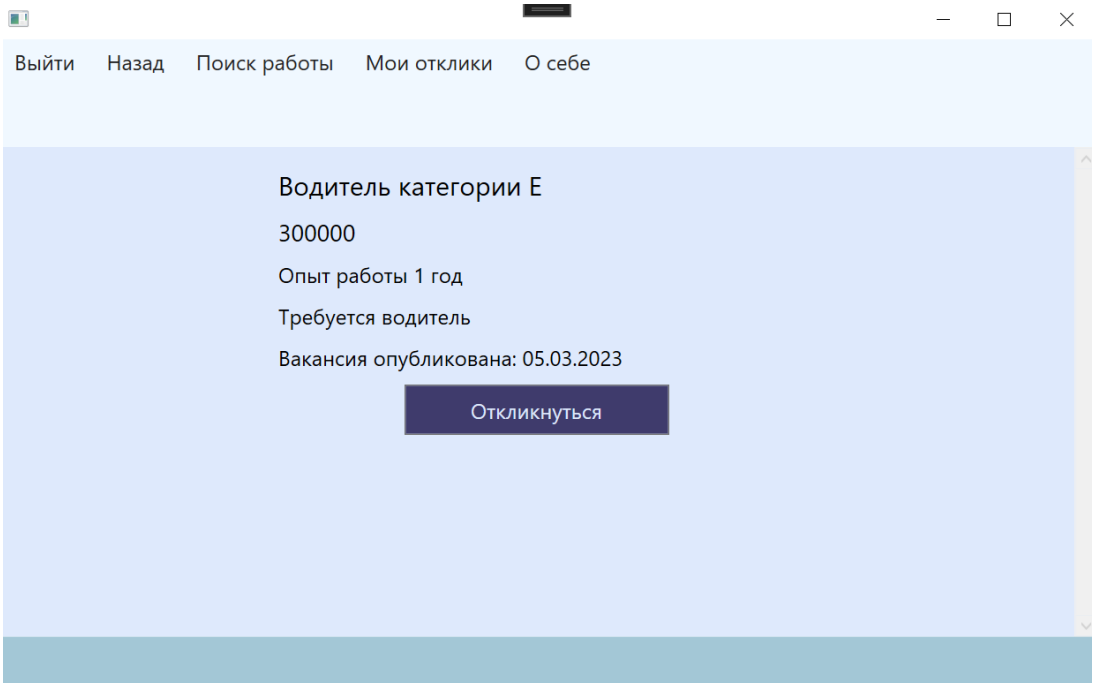

Рисунок 22 - окно выбора вакансии

Заключение

В процессе выполнения курсовой работы был выполнен анализ предметной области, в рамках которого был проведен обзор существующих систем автоматизации центра занятости населения, благодаря чему была выявлена структура и основные функции разрабатываемой конфигурации. Помимо этого, выполнен анализ функций и бизнес-процессов, который помог выявить слабые стороны управления организацией.

В данной работе был выполнен обзор систем управления базами данных (СУБД), затем было проведено исследование предметной области, составлен проект структуры базы данных и программного обеспечения, разработаны алгоритмы их функционирования.

Спроектированная база данных центра занятости населения дает возможность удобного ввода, редактирования, удаления и хранения данных. В данной курсовой работе была осуществлена задача по проектированию и созданию базы данных, а также клиент-серверного пользовательского приложения для ее администрирования с использованием клиент-серверной технологии. Проектная часть была разработана в среде Microsoft SQL Management Studio Server (SSMS) и Visual Studio (VS) .

Информационные задачи баз данных удовлетворяют целям деятельности предприятия, а именно:

- осуществляется удобный ввод данных в БД;

- возможность нахождения одних данных по другим;

В ходе выполнения курсовой работы были приобретены практические и теоретические знания и навыки в области создания баз данных, а также создания пользовательского приложения.

Таким образом, задачи, поставленные в начале курсовой работы, были выполнены, а цель - достигнута.

Список литературы

Печатные издания

1. Фёдорова Г.Н. Разработка, администрирование и защита баз данных: учебник для студ. учреждений сред. проф. образования - М.: Издательский центр «Академия», 2020. -208 с.
2. Нестеров С. А. «Базы данных»: учебник и практикум для сред. проф. образования – М.: Издательство Юрайт, 2020. - 230 с.
3. Новиков, Горшкова, Графеева «Основы технологий баз данных»: учебник для студ. сред. проф. образования - 2-е изд. - М.: ДМК Пресс, 2020. - 582 с.
4. Фуфаев Э.В. «Базы данных»: учеб. пособие дли студ. учреждений сред. проф. образования -11-е изд. - М.: Издательский центр «Академия», 2021. - 320 с.

Электронные издания (электронные ресурсы)

5. «Базы данных» курс лекций: Учебник. Автор/создатель Карпова И.П. Единое окно доступа к образовательным ресурсам.

https://publications.hse.ru/mirror/pubs/share/direct/259052819

6. Электронный учебный материал «Основы создания и администрирования базы данных средствами Microsoft SQL Server». Автор/создатель Э.М. Арымбекова. Единое окно доступа к образовательным ресурсам.

https://elar.rsvpu.ru/bitstream/123456789/29378/1/RSVPU_2019_307.pdf

7. Полное руководство по языку программирования C# и платформе .NET https://metanit.com/sharp/tutorial/
8. Лучший видеокурс по C# и .NET https://proglib.io/p/csharp-starter/

Приложения

Приложение 1

```
Create database EmploymentCenter
SET QUOTED_IDENTIFIER OFF;
GO
USE [EmploymentCenter];
GO
IF SCHEMA_ID(N'dbo') IS NULL
EXECUTE(N'CREATE SCHEMA [dbo]');
GO

IF
OBJECT_ID(N'[dbo].[FK_Вакансия_Работодатель]',
'F') IS NOT NULL
   ALTER TABLE [dbo].[Вакансия] DROP
CONSTRAINT [FK_Вакансия_Работодатель];
GO
IF
OBJECT_ID(N'[dbo].[FK_Вакансия_Специальности]',
'F') IS NOT NULL
   ALTER TABLE [dbo].[Вакансия] DROP
CONSTRAINT [FK_Вакансия_Специальности];
GO
IF OBJECT_ID(N'[dbo].[FK_Отклик_Вакансия]', 'F')
IS NOT NULL
   ALTER TABLE [dbo].[Отклик] DROP
CONSTRAINT [FK_Отклик_Вакансия];
GO
IF OBJECT_ID(N'[dbo].[FK_Отклик_Пользователь]',
'F') IS NOT NULL
   ALTER TABLE [dbo].[Отклик] DROP
CONSTRAINT [FK_Отклик_Пользователь];
GO
IF OBJECT_ID(N'[dbo].[sysdiagrams]', 'U') IS NOT
NULL
   DROP TABLE [dbo].[sysdiagrams];
GO
IF OBJECT_ID(N'[dbo].[Вакансия]', 'U') IS NOT
NULL
   DROP TABLE [dbo].[Вакансия];
GO
IF OBJECT_ID(N'[dbo].[Отклик]', 'U') IS NOT NULL
   DROP TABLE [dbo].[Отклик];
GO
IF OBJECT_ID(N'[dbo].[Пользователь]', 'U') IS NOT
NULL
   DROP TABLE [dbo].[Пользователь];
GO
IF OBJECT_ID(N'[dbo].[Работодатель]', 'U') IS NOT
NULL
   DROP TABLE [dbo].[Работодатель];
GO
IF OBJECT_ID(N'[dbo].[Специалист]', 'U') IS NOT
NULL
   DROP TABLE [dbo].[Специалист];
GO
IF OBJECT_ID(N'[dbo].[Специальности]', 'U') IS NOT
NULL
   DROP TABLE [dbo].[Специальности];
GO
-- Creating table 'sysdiagrams'
CREATE TABLE [dbo].[sysdiagrams] (
   [name] nvarchar(128)  NOT NULL,
   [principal_id] int  NOT NULL,
   [diagram_id] int IDENTITY(1,1) NOT NULL,
   [version] int  NULL,
   [definition] varbinary(max)  NULL
);
GO
-- Creating table 'Специалист'
CREATE TABLE [dbo].[Специалист] (
   [Код] int IDENTITY(1,1) NOT NULL,
   [Фио] nvarchar(50)  NULL,
   [Телефон] nvarchar(50)  NULL,
   [Пароль] nvarchar(50)  NULL
);
GO
-- Creating table 'Специальности'
CREATE TABLE [dbo].[Специальности] (
   [Код] int IDENTITY(1,1) NOT NULL,
   [Имя] nvarchar(50)  NULL
);
GO
-- Creating table 'Вакансия'
CREATE TABLE [dbo].[Вакансия] (
   [Код] int IDENTITY(1,1) NOT NULL,
   [КодСпециальности] int  NULL,
   [КодРаботодателя] int  NULL,
   [Название] nvarchar(50)  NULL,
   [Описание] nvarchar(max)  NULL,
   [Статус] bit  NULL,
   [Дата] datetime  NULL,
   [Опыт] int  NULL,
   [Зарплата] nvarchar(20)  NULL
);
GO
-- Creating table 'Отклик'
CREATE TABLE [dbo].[Отклик] (
   [Код] int IDENTITY(1,1) NOT NULL,
   [КодВакансии] int  NOT NULL,
   [КодПользователя] int  NOT NULL,
   [ДатаОтправки] datetime  NOT NULL,
   [Статус] int  NOT NULL
);
GO
-- Creating table 'Пользователь'
CREATE TABLE [dbo].[Пользователь] (
   [Код] int IDENTITY(1,1) NOT NULL,
   [Фио] nvarchar(50)  NULL,
   [Телефон] nvarchar(50)  NULL,
   [Пароль] nvarchar(50)  NULL,
   [Резюме] nvarchar(max)  NULL,
   [Почта] nvarchar(50)  NULL
);
GO
```

```
-- Creating table 'Работодатель'
CREATE TABLE [dbo].[Работодатель] (
    [Код] int IDENTITY(1,1) NOT NULL,
    [Фио] nvarchar(50)  NULL,
    [Телефон] nvarchar(50)  NULL,
    [Пароль] nvarchar(50)  NULL,
    [НаименованиеКомпании] nvarchar(50)  NULL,
    [Почта] nvarchar(50)  NULL
);
GO
-- Creating primary key on [diagram_id] in table
'sysdiagrams'
ALTER TABLE [dbo].[sysdiagrams]
ADD CONSTRAINT [PK_sysdiagrams]
    PRIMARY KEY CLUSTERED ([diagram_id] ASC);
GO

-- Creating primary key on [Код] in table 'Специалист'
ALTER TABLE [dbo].[Специалист]
ADD CONSTRAINT [PK_Специалист]
    PRIMARY KEY CLUSTERED ([Код] ASC);
GO

-- Creating primary key on [Код] in table
'Специальности'
ALTER TABLE [dbo].[Специальности]
ADD CONSTRAINT [PK_Специальности]
    PRIMARY KEY CLUSTERED ([Код] ASC);
GO

-- Creating primary key on [Код] in table 'Вакансия'
ALTER TABLE [dbo].[Вакансия]
ADD CONSTRAINT [PK_Вакансия]
    PRIMARY KEY CLUSTERED ([Код] ASC);
GO

-- Creating primary key on [Код] in table 'Отклик'
ALTER TABLE [dbo].[Отклик]
ADD CONSTRAINT [PK_Отклик]
    PRIMARY KEY CLUSTERED ([Код] ASC);
GO

-- Creating primary key on [Код] in table 'Пользователь'
ALTER TABLE [dbo].[Пользователь]
ADD CONSTRAINT [PK_Пользователь]
    PRIMARY KEY CLUSTERED ([Код] ASC);
GO

-- Creating primary key on [Код] in table 'Работодатель'
ALTER TABLE [dbo].[Работодатель]
ADD CONSTRAINT [PK_Работодатель]
    PRIMARY KEY CLUSTERED ([Код] ASC);
GO

-- Creating foreign key on [КодРаботодателя] in table
'Вакансия'
ALTER TABLE [dbo].[Вакансия]
ADD CONSTRAINT [FK_Вакансия_Работодатель]
    FOREIGN KEY ([КодРаботодателя])
    REFERENCES [dbo].[Работодатель]
        ([Код])
    ON DELETE NO ACTION ON UPDATE NO
ACTION;
GO

-- Creating non-clustered index for FOREIGN KEY
'FK_Вакансия_Работодатель'
CREATE INDEX [IX_FK_Вакансия_Работодатель]
ON [dbo].[Вакансия]
    ([КодРаботодателя]);
GO

-- Creating foreign key on [КодСпециальности] in table
'Вакансия'
ALTER TABLE [dbo].[Вакансия]
ADD CONSTRAINT [FK_Вакансия_Специальности]
    FOREIGN KEY ([КодСпециальности])
    REFERENCES [dbo].[Специальности]
        ([Код])
    ON DELETE NO ACTION ON UPDATE NO
ACTION;
GO

-- Creating non-clustered index for FOREIGN KEY
'FK_Вакансия_Специальности'
CREATE INDEX [IX_FK_Вакансия_Специальности]
ON [dbo].[Вакансия]
    ([КодСпециальности]);
GO

-- Creating foreign key on [КодВакансии] in table
'Отклик'
ALTER TABLE [dbo].[Отклик]
ADD CONSTRAINT [FK_Отклик_Вакансия]
    FOREIGN KEY ([КодВакансии])
    REFERENCES [dbo].[Вакансия]
        ([Код])
    ON DELETE NO ACTION ON UPDATE NO
ACTION;
GO

-- Creating non-clustered index for FOREIGN KEY
'FK_Отклик_Вакансия'
CREATE INDEX [IX_FK_Отклик_Вакансия]
ON [dbo].[Отклик]
    ([КодВакансии]);
GO

-- Creating foreign key on [КодПользователя] in table
'Отклик'
ALTER TABLE [dbo].[Отклик]
ADD CONSTRAINT [FK_Отклик_Пользователь]
    FOREIGN KEY ([КодПользователя])
    REFERENCES [dbo].[Пользователь]
        ([Код])
    ON DELETE NO ACTION ON UPDATE NO
ACTION;
GO

-- Creating non-clustered index for FOREIGN KEY
'FK_Отклик_Пользователь'
CREATE INDEX [IX_FK_Отклик_Пользователь]
ON [dbo].[Отклик]
    ([КодПользователя]);
GO
```

```
using EmploymentCenter.Model;
using EmploymentCenter.Service;
using EmploymentCenter.Windows;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace EmploymentCenter.Pages
{
    /// <summary>
    /// Логика взаимодействия для LoginPage.xaml
    /// </summary>
    public partial class LoginPage : Page
    {
        public LoginPage()
        {
            InitializeComponent();
        }

        private void RegClick(object sender,
RoutedEventArgs e)
        {
            LoginWin.Frame.Navigate(new RegUserPage());
        }

        private void LoginClick(object sender,
RoutedEventArgs e)
        {
            var list =
EmploymentCenterEntities.GetContext().Пользователь.
ToList();
            foreach (var item in list)
            {
                if (TBoxPhone.Text == item.Телефон &&
PassBox.Password == item.Пароль)
                {
                    Session.User1 = item;
                    Auth(1);
                    return;
                }
            }
            var list1 =
EmploymentCenterEntities.GetContext().Работодатель.
ToList();
            foreach (var item in list1)
            {
                if (TBoxPhone.Text == item.Телефон &&
PassBox.Password == item.Пароль)
                {
                    Session.User2 = item;
                    Auth(2);
                    return;
                }
            }

            var list2 =
EmploymentCenterEntities.GetContext().Специалист.T
oList();
            foreach (var item in list2)
            {
                if (TBoxPhone.Text == item.Телефон &&
PassBox.Password == item.Пароль)
                {
                    Session.User3 = item;
                    Auth(3);
                    return;
                }
            }

            MessageBox.Show("Пользователь не
найден!");
        }

        private void Auth(int i)
        {
            AppWin appWin = new AppWin(i);
            appWin.Show();
           var a =  Window.GetWindow(this);
            a.Close();
        }
    }
}

using EmploymentCenter.Model;
using EmploymentCenter.Service;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace EmploymentCenter.Pages
{
    /// <summary>
    /// Логика взаимодействия для
MyResponsePage.xaml
    /// </summary>
    public partial class MyResponsePage : Page
    {
        public MyResponsePage()
        {
            InitializeComponent();
```

```
        }

        private void Page_Loaded(object sender,
RoutedEventArgs e)
        {
            DGridClient.ItemsSource =
EmploymentCenterEntities.GetContext().Отклик.Where
(q => q.КодПользователя ==
Session.User1.Код).ToList();
        }
    }
}
using EmploymentCenter.Model;
using EmploymentCenter.Service;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace EmploymentCenter.Pages
{
    /// <summary>
    /// Логика взаимодействия для
MyVacancyPage.xaml
    /// </summary>
    public partial class MyVacancyPage : Page
    {
        public MyVacancyPage()
        {
            InitializeComponent();
        }

        private void Page_Loaded(object sender,
RoutedEventArgs e)
        {
            DGridClient.ItemsSource =
EmploymentCenterEntities.GetContext().Вакансия.Whe
re(q => q.КодРаботодателя ==
Session.User2.Код).ToList();
        }

        private void AddClick(object sender,
RoutedEventArgs e)
        {
            PageNavigator.Frame.Navigate(new
VacancyAddEditPage(null));
        }

        private void EditClick(object sender,
RoutedEventArgs e)
        {
            PageNavigator.Frame.Navigate(new
VacancyAddEditPage(DGridClient.SelectedItem as
Вакансия));
        }

        private void DelClick(object sender,
RoutedEventArgs e)
        {
            if (DGridClient.SelectedItem == null)
            {
                return;
            }
            Вакансия vacancy = DGridClient.SelectedItem
as Вакансия;
            if (vacancy.Статус == null)
            {
                MessageBox.Show("Вы не можете это
сделать! Данная вакансия находится на
рассмотрении!");
            }
            else
            {
                vacancy.Статус = !vacancy.Статус;

EmploymentCenterEntities.GetContext().SaveChanges();
                Page_Loaded(null, null);
            }

        }

        private void ViewResponsePageClick(object
sender, RoutedEventArgs e)
        {
            PageNavigator.Frame.Navigate(new
ResponseWorkersPage(DGridClient.SelectedItem as
Вакансия));
        }
    }
}
using EmploymentCenter.Model;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace EmploymentCenter.Pages
{
    /// <summary>
    /// Логика взаимодействия для
NewVacancyPage.xaml
    /// </summary>
    public partial class NewVacancyPage : Page
    {
```

```
        public NewVacancyPage()
        {
            InitializeComponent();
        }

        private void EditClick(object sender,
RoutedEventArgs e)
        {
            if (DGridClient.SelectedItem == null)
            {
                MessageBox.Show("Нужно выбрать
запись!");
                return;
            }
            PageNavigator.Frame.Navigate(new
VacancyAddEditPage(DGridClient.SelectedItem as
Вакансия));
        }

        private void OkClick(object sender,
RoutedEventArgs e)
        {
            if (DGridClient.SelectedItem == null)
            {
                MessageBox.Show("Нужно выбрать
запись!");
                return;
            }
            var vacancy = DGridClient.SelectedItem as
Вакансия;
            vacancy.Статус = true;

EmploymentCenterEntities.GetContext().SaveChanges();
            Page_Loaded(null, null);
        }

        private void DelClick(object sender,
RoutedEventArgs e)
        {
            if (DGridClient.SelectedItem == null)
            {
                MessageBox.Show("Нужно выбрать
запись!");
                return;
            }
            var vacancy = DGridClient.SelectedItem as
Вакансия;

EmploymentCenterEntities.GetContext().Вакансия.Rem
ove(vacancy);

EmploymentCenterEntities.GetContext().SaveChanges();
            MessageBox.Show("Вакансия удалена");
            Page_Loaded(null, null);

        }

        private void Page_Loaded(object sender,
RoutedEventArgs e)
        {
            DGridClient.ItemsSource =
EmploymentCenterEntities.GetContext().Вакансия.Whe
re(q => q.Статус == null).ToList();
        }
    }
}
using EmploymentCenter.Model;
using EmploymentCenter.Service;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace EmploymentCenter.Pages
{
    /// <summary>
    /// Логика взаимодействия для
PersonalDataPage.xaml
    /// </summary>
    public partial class PersonalDataPage : Page
    {
        public PersonalDataPage()
        {
            InitializeComponent();
            DataContext = Session.User1;
        }

        private void SaveClick(object sender,
RoutedEventArgs e)
        {

EmploymentCenterEntities.GetContext().SaveChanges();
            MessageBox.Show("Данные были
обновлены!");
        }
    }
}
using EmploymentCenter.Model;
using EmploymentCenter.Windows;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace EmploymentCenter.Pages
{
```

```
    /// <summary>
    /// Логика взаимодействия для RegUserPage.xaml
    /// </summary>
    public partial class RegUserPage : Page
    {
        public RegUserPage()
        {
            InitializeComponent();
            _user = new Пользователь();
            _user1 = new Работодатель();
            CBoxRole.Items.Add("Соискатель");
            CBoxRole.Items.Add("Работодатель");
            CBoxRole.SelectedIndex = 0;


        }
        private Пользователь _user;
        private Работодатель _user1;

        private void CBoxRole_SelectionChanged(object
sender, SelectionChangedEventArgs e)
        {
            if (CBoxRole.SelectedIndex == 0)
            {
                UserBar.Visibility = Visibility.Visible;
                User2Bar.Visibility = Visibility.Collapsed;
                DataContext = _user;
            }
            else if (CBoxRole.SelectedIndex == 1)
            {
                UserBar.Visibility = Visibility.Collapsed;
                User2Bar.Visibility = Visibility.Visible;
                DataContext = _user1;
            }

        }

        private void SaveClick(object sender,
RoutedEventArgs e)
        {
            if
(EmploymentCenterEntities.GetContext().Работодатель.
Any(q => q.Телефон == TBoxPhone.Text) ||
EmploymentCenterEntities.GetContext().Пользователь.
Any(q => q.Телефон == TBoxPhone.Text))
            {
                MessageBox.Show("Данный номер телефона
занят!");
                return;
            }

            foreach (var item in TBoxPhone.Text)
            {
                if (item == '_')
                {
                    MessageBox.Show("Заполните номер
телефона!");
                    return;
                }
            }

            if (PassBox1.Password != PassBox2.Password)
            {
                MessageBox.Show("Введенные пароли не
совпадают!");
                return;
            }
            if (CBoxRole.SelectedIndex == 0)
            {
                _user.Пароль = PassBox1.Password;

EmploymentCenterEntities.GetContext().Пользователь.
Add(_user);
            }

            else if (CBoxRole.SelectedIndex == 1)
            {
                _user1.Пароль = PassBox1.Password;

EmploymentCenterEntities.GetContext().Работодатель.
Add(_user1);
            }



EmploymentCenterEntities.GetContext().SaveChanges();
            MessageBox.Show("Регистрация прошла
успешно!");
            LoginWin.Frame.NavigationService.GoBack();
        }

        private void BackClick(object sender,
RoutedEventArgs e)
        {
            LoginWin.Frame.NavigationService.GoBack();
        }
    }
}
using EmploymentCenter.Model;
using EmploymentCenter.Service;
using EmploymentCenter.Windows;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace EmploymentCenter.Pages
{
    /// <summary>
    /// Логика взаимодействия для
ResponseWorkersPage.xaml
    /// </summary>
    public partial class ResponseWorkersPage : Page
    {
        public ResponseWorkersPage(Вакансия вакансия)
        {
```

```
            InitializeComponent();
            _vacancy = вакансия;
        }
        private Вакансия _vacancy;

        private void ResumeClick(object sender,
RoutedEventArgs e)
        {
            Пользователь user = (DGridClient.SelectedItem
as Отклик).Пользователь;
            ResumeWin resumeWin = new
ResumeWin(user);
            resumeWin.Show();
        }

        private void PageLoaded(object sender,
RoutedEventArgs e)
        {
            DGridClient.ItemsSource =
EmploymentCenterEntities.GetContext().Отклик.Where
(q => q.Вакансия.Код == _vacancy.Код && q.Статус
== 0).ToList();
        }

        private void AcceptClick(object sender,
RoutedEventArgs e)
        {

EmploymentCenterEntities.GetContext().SaveChanges();
            MessageBox.Show("Что то делаю");
            Отклик отклик = DGridClient.SelectedItem as
Отклик;
            отклик.Статус =
(int)StatusResponseEnum.Принят;

EmploymentCenterEntities.GetContext().SaveChanges();
            MessageBox.Show(отклик.Код.ToString());

            Mail.SendMail(отклик.Пользователь.Почта,
отклик);
            PageLoaded(null, null);



        }

        private void DeclineClick(object sender,
RoutedEventArgs e)
        {
            Отклик отклик = DGridClient.SelectedItem as
Отклик;
            отклик.Статус =
(int)StatusResponseEnum.Отклонен;

EmploymentCenterEntities.GetContext().SaveChanges();
            PageLoaded(null, null);
        }


    }
}
using EmploymentCenter.Model;
using EmploymentCenter.Service;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace EmploymentCenter.Pages
{
    /// <summary>
    /// Логика взаимодействия для
SelectedVacancyPage.xaml
    /// </summary>
    public partial class SelectedVacancyPage : Page
    {
        public SelectedVacancyPage(Вакансия
вакансия,bool isAdmin)
        {
            InitializeComponent();
            _IsAdmin = isAdmin;
            _Vacancy = вакансия;
            DataContext = _Vacancy;
            if (isAdmin)
            {
                BtnSend.Visibility = Visibility.Collapsed;
                BtnDel.Visibility = Visibility.Visible;
            }
        }
        private Вакансия _Vacancy;
        private bool IsNotSended;
        private bool _IsAdmin;

        private void SendClick(object sender,
RoutedEventArgs e)
        {
            if (IsNotSended)
            {

                Отклик отклик = new Отклик
                {
                    Пользователь = Session.User1,
                    Вакансия = _Vacancy,
                    ДатаОтправки = DateTime.Now,
                    Статус = 0
                };

EmploymentCenterEntities.GetContext().Отклик.Add(о
тклик);

EmploymentCenterEntities.GetContext().SaveChanges();
                Page_Loaded(null, null);
            }
            else
            {
```

```
            PageNavigator.Frame.Navigate(new
MyResponsePage());
        }



    }

    private bool CheckSend()
    {
        bool x = 0 ==
EmploymentCenterEntities.GetContext().
            Отклик.Where(q => q.КодВакансии ==
_Vacancy.Код && q.КодПользователя ==
Session.User1.Код)
            .Count();
        if (x)
        {
            // если не отправил
            BtnSend.Content = "Откликнуться";
        }
        else
        {
            // если отправил
            BtnSend.Content = "Отправлен";
        }
        return x;
    }

    private void Page_Loaded(object sender,
RoutedEventArgs e)
    {
        if (_IsAdmin)
        {
            return;
        }
        IsNotSended = CheckSend();
    }

    private void DelClick(object sender,
RoutedEventArgs e)
    {
        MessageBoxResult dialogResult =
MessageBox.Show( "Вы действительно хотите
удалить эту вакансию?", "Внимание!",
MessageBoxButton.YesNo);
        if (dialogResult == MessageBoxResult.Yes)
        {
          var a=
EmploymentCenterEntities.GetContext().Отклик.Where
(q => q.КодВакансии == _Vacancy.Код).ToList();

EmploymentCenterEntities.GetContext().Отклик.Remov
eRange(a);

EmploymentCenterEntities.GetContext().SaveChanges();

EmploymentCenterEntities.GetContext().Вакансия.Rem
ove(_Vacancy);

EmploymentCenterEntities.GetContext().SaveChanges();
            PageNavigator.Frame.Navigate(new
VacancyList(true));
            MessageBox.Show("Вакансия была
удалена!");
        }
        else if (dialogResult == MessageBoxResult.No)
        {
            //do something else
        }
    }
  }
}
using EmploymentCenter.Model;
using EmploymentCenter.Service;
using System;
using System.Collections.Generic;
using System.Data.Entity.Infrastructure;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace EmploymentCenter.Pages
{
  /// <summary>
  /// Логика взаимодействия для
VacancyAddEditPage.xaml
  /// </summary>
  public partial class VacancyAddEditPage : Page
  {
    public VacancyAddEditPage(Вакансия вакансия)
    {
      InitializeComponent();
      if (вакансия == null)
      {
        _Vacancy = new Вакансия {
          Дата=DateTime.Now,
          Работодатель = Session.User2
        };
      }
      else
      {
        _Vacancy = вакансия;
      }
      DataContext = _Vacancy;
    }
    private List<Специальности> _SpecList;
    private Вакансия _Vacancy;
    private void Page_Loaded(object sender,
RoutedEventArgs e)
    {
      _SpecList =
EmploymentCenterEntities.GetContext().Специальност
и.ToList();
      CBoxProf.ItemsSource = _SpecList;

    }
```

```
        private void TBoxSearch_TextChanged(object
sender, TextChangedEventArgs e)
        {
            if (TBoxSearch.Text == "")
            {
                CBoxProf.ItemsSource = _SpecList;
                CBoxProf.SelectedItem = null;
                return;
            }
            CBoxProf.IsDropDownOpen = true;
            string Text = TBoxSearch.Text;
            CBoxProf.ItemsSource =
EmploymentCenterEntities.GetContext().Специальност
и.Where(q => q.Имя.Contains(Text)).ToList();
        }

        private void SaveClick(object sender,
RoutedEventArgs e)
        {
            if (_Vacancy.Код == 0)
EmploymentCenterEntities.GetContext().Вакансия.Add(
_Vacancy);

EmploymentCenterEntities.GetContext().SaveChanges();
            MessageBox.Show("Информация сохранена");
            PageNavigator.Back();
        }
    }
}
using EmploymentCenter.Model;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace EmploymentCenter.Pages
{
    /// <summary>
    /// Логика взаимодействия для VacancyList.xaml
    /// </summary>
    public partial class VacancyList : Page
    {
        public VacancyList(bool IsAdmin)
        {
            InitializeComponent();
            _IsAdmin = IsAdmin;
        }
        bool _IsAdmin;
        private List<Вакансия> _VacancyList;

        private void Page_Loaded(object sender,
RoutedEventArgs e)
        {
            _VacancyList =
EmploymentCenterEntities.GetContext().Вакансия.Whe
re(q => q.Статус.Value).ToList();
            LView.ItemsSource = _VacancyList;
        }

        private void LView_SelectionChanged(object
sender, SelectionChangedEventArgs e)
        {
            PageNavigator.Frame.Navigate(new
SelectedVacancyPage(LView.SelectedItem as
Вакансия,_IsAdmin));
        }

        private void TBoxSearch_TextChanged(object
sender, TextChangedEventArgs e)
        {
            if (TBoxSearch.Text == "")
            {
                LView.ItemsSource = _VacancyList;
                return;
            }
            string text = TBoxSearch.Text.ToLower();
            LView.ItemsSource =
EmploymentCenterEntities.GetContext().Вакансия.Whe
re(q => q.Статус.Value &&
(q.Название.ToLower().Contains(text) ||
(q.Специальности.Имя.ToLower().Contains(text)))).To
List();
        }
    }
}
```