

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ»**
(Н И У « Б е л Г У »)

ИНСТИТУТ ИНЖЕНЕРНЫХ ТЕХНОЛОГИЙ И ЕСТЕСТВЕННЫХ НАУК

Кафедра прикладной информатики и информационных технологий

**РАЗРАБОТКА АВТОМАТИЗИРОВАННОЙ ПОДСИСТЕМЫ ПОИСКА
ВАКАНСИЙ В ЦЕНТРЕ ЗАНЯТОСТИ НАСЕЛЕНИЯ**

Выпускная квалификационная работа

**студента очной формы обучения
направления подготовки 09.03.03 Прикладная информатика
4 курса группы 07001204
Жудина Владимира Александровича**

Научный руководитель:
Зайцева Н.О.

БЕЛГОРОД 2016

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	3
1 ИЗУЧЕНИЕ И АНАЛИЗ ИСПОЛЬЗОВАНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ В ЦЕНТРЕ ЗАНЯТОСТИ НАСЕЛЕНИЯ.....	5
1.1 Общая характеристика структуры и деятельности центра занятости населения.....	5
1.2 Анализ существующих автоматизированных систем обслуживания клиентов в центрах занятости.....	11
1.3 Обоснование необходимости автоматизации и цели создания подсистемы поиска вакансий в центре занятости населения.....	14
2 ОБОСНОВАНИЕ ПРОГРАММНЫХ РЕШЕНИЙ АВТОМАТИЗИРОВАННОЙ ПОДСИСТЕМЫ ПОИСКА ВАКАНСИЙ.....	24
2.1 Техническое и кадровое обеспечение подсистемы поиска вакансий	24
2.2 Характеристика функций подсистемы поиска вакансий.....	30
2.3 Информационное обеспечение подсистемы поиска вакансий.....	33
2.4. Программное обеспечение подсистемы поиска вакансий.....	37
3 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ	39
3.1 Создание базы данных для подсистемы	39
3.2 Создание интерфейса подсистемы поиска вакансий	42
3.3 Тестирование программы.....	52
3.4 Оценка экономической эффективности внедрения подсистемы..	60
ЗАКЛЮЧЕНИЕ	65
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	67
ПРИЛОЖЕНИЕ	72

ВВЕДЕНИЕ

Важнейшей задачей комплексного преобразования экономики России и Белгородской области является решение проблем безработицы и занятости. Необходимость решения задач по содействию занятости разных категорий граждан путём повышения эффективности деятельности центров занятости населения обусловлена колебанием спроса и предложения рабочей силы на рынке труда.

Это предопределяет необходимость планирования мероприятий по реализации государственной политики в области регулирования рынка труда, развития электронных услуг, включая мероприятия по информационной поддержке трудоустройства. Следует отметить, что в службу занятости населения Белгородской области ежегодно обращается от 90 до 130 тыс. граждан за предоставлением государственных услуг. Из них по вопросу трудоустройства – более 50 тыс. человек. Это требует значительного увеличения скорости процесса обработки, накопления и распространения данных. Становится просто необходимой эксплуатация автоматизированных средств, дающая возможность обрабатывать, хранить и эффективно распределять собранные данные, в том числе информацию, о наличии вакантных мест на предприятиях и организациях. Эффективная работа по преодолению безработицы, исходя из современных требований, имеет прямую зависимость от оснащения организации информационными средствами, в том числе, автоматизированным учётом на базе компьютерных систем.

Как показывает анализ деятельности центров занятости, сегодня имеет место проблема ограниченного применения возможностей ПК в трудоустройстве граждан, испытывающих трудности в поиске работы.

Разрешению этой проблемы будет посвящаться данная выпускная квалификационная работа, целью исследования которой является разработка и обоснование автоматизированной подсистемы поиска вакансий.

Объектом исследования является ОКУ «Белгородский центр занятости населения».

Предмет исследования: информационная поддержка трудоустройства населения.

В соответствии с целью задачами выпускной квалификационной работы будут являться:

- анализ структуры и организации деятельности Белгородского центра занятости населения;
- выявление проблем информационной поддержки трудоустройства населения;
- оценка эффективности разработанной автоматизированной подсистемы поиска вакансий.

Выпускная квалификационная работа состоит из 3 разделов:

- 1) Изучение и анализ исследования информационных систем в центре занятости населения;
- 2) Обоснование программных решений автоматизированной подсистемы поиска вакансий;
- 3) Программная реализация.

ВКР включает в себя: 3 раздела; 10 таблиц, 44 рисунка, список использованных источников, приложение.

1 ИЗУЧЕНИЕ И АНАЛИЗ ИСПОЛЬЗОВАНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ В ЦЕНТРЕ ЗАНЯТОСТИ НАСЕЛЕНИЯ

1.1 Общая характеристика структуры и деятельности центра занятости населения

Обеспечение более полной и эффективной занятости населения является из одной из наиважнейших задач любого демократического общества.

В России эту задачу решает государственная система службы занятости. Анализ ее структуры свидетельствует о наличии трех уровней подчиненности:

- 1 уровень – федеральная служба занятости;
- 2 уровень - региональные организации службы занятости;
- 3 уровень местные организации службы занятости, в том числе городские, районные организации, их филиалы, бюро, центры и т.д.

Следует отметить нацеленность государственной политики занятости в стране не только на регулирование общих процессов в сфере труда, но и на осуществление их в рамках мер, разработанных для регулирования организации региональных рынков труда. В данном исследовании мы остановимся на изучении проблем информатизации деятельности центров занятости II уровня, то есть региональной службы занятости. Как отмечается в исследовании Стрельченко Е.А., похожие по разным регионам тенденции функционирования рынков труда обеспечивают эффект при едином подходе к решению проблем занятости, особенно в расчёте на перспективу[1]. Для каждого региона характерны собственные дифференцированные подходы к регулированию рынка труда, установлены свои приоритеты. Остановимся на приоритетах регулирования рынка труда, характерных для Центрально-Черноземного региона. Согласно данным литературных источников к ним относится совершенствование профессиональной переподготовки

высвобождаемых работников. Обратим внимание на тот факт, что создание новых рабочих мест является актуальной задачей. На уровне региона осуществляется ориентация на агропромышленный профиль подготовки и переподготовки работников, содействие самозанятости населения, усиление агропромышленной организации региона [2].

Реализация этих приоритетных направлений требует от центра занятости населения новых подходов к хранению, структурированию, систематизации больших объемов данных.

Мы полагаем, что кроме развития системных устройств, средств передачи данных, памяти необходимых для применения значительных объемов хранимой информации, нужны средства обеспечения диалога ЭВМ - человек. Эти средства позволят пользователю запрашивать информацию, пополнять базу данных, модифицировать ее, читать файлы и принимать решения на основании хранимых данных. Системы управления базами данных (СУБД) является специализированным средством, обеспечивающие эти функции.

Спрос на вакансии и спрос на работников является постоянным, поэтому деятельность центров занятости нуждается в автоматизации ведения банка данных клиентов. Центру при большом потоке информации о вакансиях, который меняется сложно оперативно и точно предоставлять необходимые данные, используя только бумагу и ручку. Поэтому актуальной является автоматизация деятельности центра занятости, в частности тех отделов, которые непосредственно работают с клиентами и подбирают для них вакансии. Именно это направление деятельности является предметом нашего исследования.

Мы полагаем, что автоматизация деятельности центра занятости способствует реализации стратегии и тактики Правительства области в сфере труда и занятости населения. Оптимальное использование трудового потенциала области за счёт создания гибкого рынка труда.

Как отмечается в программе «Содействие занятости населения Белгородской области на 2014-2020 годы» объединения усилий органов государственной власти Белгородской области, органов местного самоуправления области, общественных объединений, всех социальных партнеров области будет способствовать решению актуальных задач рынка труда с использованием программно-целевых методов и их информационного обеспечения в рамках государственной программы [3].

Таким образом, решение имеющихся проблем рынка труда возможно только путём реализации государственной программы, направленной на проведение государственной политики в сфере занятости населения, ее конкретизацию на региональном уровне с учётом условий Белгородской области, а также на решение специфических проблем занятости населения. В этот период управлением по труду и занятости населения области будет осуществляться деятельность по повышению качества осуществляемых мер, что будет содействовать эффективной реализации прав граждан на полную, продуктивную и свободно избранную занятость за счёт адресного и гибкого механизма финансирования и оказания государственных услуг в области содействия занятости населения. Координация государственной программы с программами социального развития области, а также с инвестиционными и социальными проектами: предусматривает предоставление ряда государственных услуг в области занятости. Реализацию задач поставленных в программе «Содействие занятости населения Белгородской области на 2014-2020 годы» осуществляет областное казенное учреждение Белгородской области «Белгородский центр занятости». Структура Областного казенного учреждения «Белгородский центр занятости населения» представлена на рисунке 1.1.

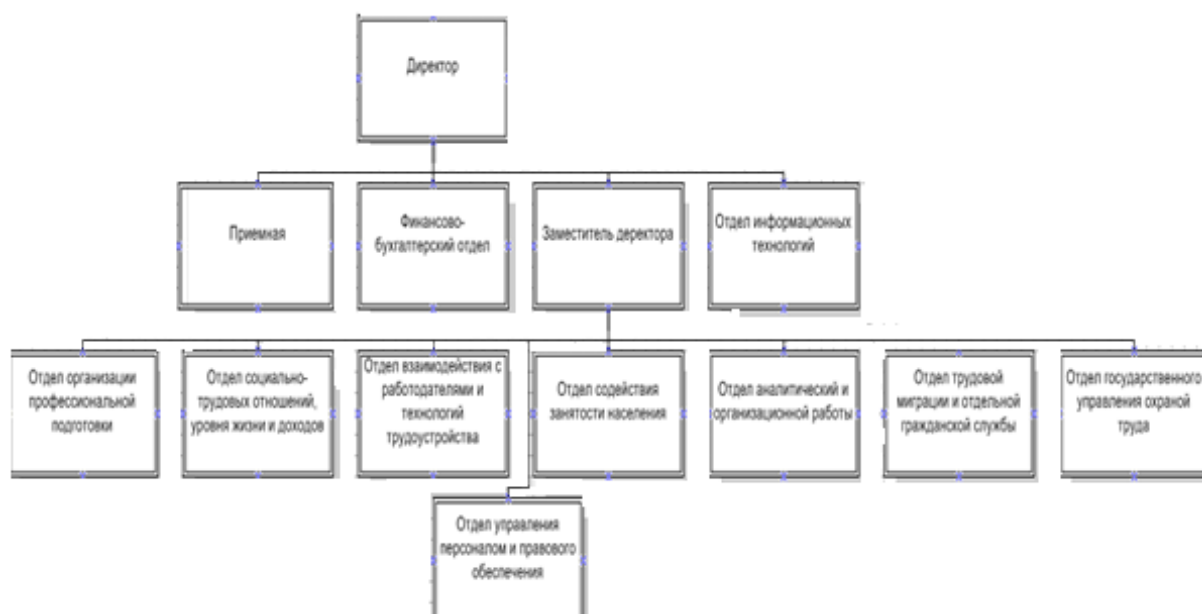


Рисунок 1.1 – Структура Областного казенного учреждения «Белгородский центр занятости населения»

Анализ материалов сайта Белгородского центра занятости населения [4] позволил выделить основные задачи и функции управления отделов службы занятости. К ним относятся:

- осуществление на территории области и в пределах своей компетенции государственной политики в области социально-трудовых отношений и содействия занятости населения;
- повышение качества жизни, доходов населения, оплаты труда работников областных бюджетных учреждений, повышение качества управления охраной труда;
- реализация деятельности по изучению потребности в кадрах и специалистах для отраслей промышленности экономики социальной сферы;
- координация деятельности центров занятости, включая реализацию государственных гарантий в области занятости населения и региональных программ содействия занятости населения.

Как отмечается в государственной программе Белгородской области

«Содействие занятости населения Белгородской области на 2014-2020 годы» анализ современных тенденций развития сферы содействия занятости населения Белгородской области выявил ее преимущества и недостатки.

К преимуществам, в частности относятся:

- осуществление государственной политики в области содействия занятости населения и наличие инструментария по регулированию рынка труда;

- многообразие реальных возможностей по удовлетворению потребностей всех участников рынка труда, в том числе наличие мотивации к самозанятости у населения;

- рост интереса и потребностей работодателей, к квалифицированным рабочим кадрам с высокой мотивацией к труду; реализация мер по регулированию спроса и предложения рабочей силы;

- наличие высококвалифицированных специалистов, работающих в сфере занятости, реализующих государственную политику занятости.

К проблемам можно отнести:

- низкий уровень квалификации рабочей силы и заработной платы предлагаемых рабочих мест, отсутствие результативных способов воздействия на работодателей;

- существующая система налогообложения, отчислений в страховые фонды, высокие процентные ставки;

- чрезмерная регламентация административных мер по предоставлению государственных услуг.

Выявление преимуществ и проблем сферы содействия занятости важно, прежде всего, для глубокого анализа возможностей и внешних факторов, влияющих на реализацию мер содействия занятости населения.

К возможностям следует отнести:

- наличие устойчивых тенденций в государственной политике, направленных на повышение мобильности и развитие трудовых ресурсов, организацию профессионального обучения и дополнительного

профессионального образования населения за счёт бюджетных средств и под конкретные рабочие места. Реализация мер по защите рынка труда;

- развитие потенциала предприятий и системы государственных услуг по сохранению и созданию новых рабочих мест, по стимулированию трудовой мобильности, самозанятости, содействию в трудоустройстве, профессиональном образовании кадров, отвечающих требованиям инновационной экономики;

- осуществление и развитие спроса и предложения на рынке труда, в том числе рост числа граждан, имеющих высокий уровень профессионального образования, высокую квалификацию, опыт работы и мотивации к труду.

К внешним факторам угроз можно отнести:

- наличие рисков возникновения кризисов в экономике региона;
- возможность снижения мотивации занятости граждан и неспособности обеспечения населения рабочими местами в соответствии с квалификацией;

- наличие рисков введения нормативно-правовых и финансовых ограничений в предоставлении государственных услуг в сфере занятости населения.

Несмотря на то, что среди факторов, способствующих эффективной реализации задач и функций центра занятости, не выделен фактор автоматизации, в центре занятости создан отдел информационных технологий, который осуществляет функцию обслуживания локальной сети, обеспечивает обработку данных, централизацию хранения, что позволяет сберечь значительные средства, а главное время для получения информации.

В следующем параграфе мы остановимся на анализе используемых в практике деятельности центров занятости автоматизированных систем обслуживания населения [5].

1.2 Анализ существующих автоматизированных систем обслуживания клиентов в центрах занятости

В современных центрах занятости используется автоматизированная обработка данных. Анализ программного обеспечения свидетельствует о том, что предпочтение отдается стандартным приложениям Microsoft Office, где все данные можно обрабатывать в табличном редакторе Excel, а также вести БД в Access, не утруждая себя разработкой специального пользовательского интерфейса. Центр занятости содержит штат обслуживающего персонала (отдел информационных технологий) обеспечивающих деятельность по автоматизации, разработку программного обеспечения предусмотренного для данного учреждения.

Изучение опыта деятельности региональных центров занятости свидетельствует о том, что существует несколько вариантов программ по автоматизации их работы. Остановимся на их краткой характеристике.

Для передачи информации через сеть Интернет по защищенным каналам ориентирована программа «Электронный работодатель». Программа не нуждается в постоянном подключении, доступ в Интернет нужен в момент отправки и приема информационных пакетов. В отсутствие подключения к сети Интернет на рабочем месте пользователя, занимающегося вводом данных в систему, он может обмениваться подготовленными информационными пакетами с рабочим местом, подключенным к Интернет. «Электронный работодатель» и сформировать необходимый отчет в электронном виде методом выбора значений из общероссийских и ведомственных классификаторов, применяемых в службе занятости населения: Методом выбора значений из общероссийских и ведомственных классификаторов, применяемых в службе занятости населения осуществляется ввод данных в ПО. Для этого используется: общероссийский классификатор профессий рабочих, должностей служащих и тарифных

разрядов (ОКПДТР), общероссийский классификатор форм собственности (ОКФС), Общероссийский классификатор организационно-правовых форм (ОКОПФ), общероссийский классификатор специальностей по образованию (ОКСО) [6].

Пользователю помогают корректно заполнить формы подсказки, которыми снабжены поля для ввода данных.

Обратим внимание на то, что при сохранении отчетов осуществляется проверка на правильность введенной информации.

В «Электронном работодателе» есть возможность хранения сведений, кроме отчетов, необходимых для службы занятости населения, которые входят в состав формы П-4 (сведения о численности, движении работников, использовании рабочего времени), предназначенной для передачи в Госкомстат. Их можно напечатать [7].

Благодаря информационной поддержке службы занятости, нет необходимости личного присутствия работодателя или его представителя в центрах занятости населения.

Все это в целом снижает трудозатраты, повышает ввода и передачи информации от работодателей в службу занятости населения.

Остановимся на характеристике программного комплекса «Катарсис» версия 8.0, который относится к новому поколению базового программного обеспечения в линейке решений компании для автоматизации органов исполнительной власти субъекта РФ, осуществляющих полномочия в области содействия занятости населения. Он обеспечивает слаженную работу на всех уровнях службы занятости и представляет собой единую региональную информационную систему, на уровне субъекта в online режиме поддерживаются региональные банки данных: личных дел, организаций, вакансий, учебно-производственной базы региона, договоров. Обеспечивается предоставление государственных услуг гражданам и работодателям (в том числе, в электронном виде), взаимодействие с другими ведомствами в рамках СМЭВ.

Благодаря современной платформе, используемой в разработке комплекса обеспечивается высокий уровень надежности и отказоустойчивости, быстрый доступ к хранимой информации, простота интеграции с другими системами посредством электронных услуг и защиты высокого качества от несанкционированного доступа к персональным данным получателей государственных услуг.

Автоматизированная система «Мониторинг рынка труда» - это инструментарий для создания регионального хранилища данных на основе первичной информации из "Программного Комплекса "КАТАРСИС" версий 7 и 8", а также для формирования регионального сегмента Регистра получателей услуг в сфере занятости населения. Система предназначена для оперативного и всестороннего анализа рынка труда региона [8] .

Автоматизированная система "Мониторинг рынка труда" расширяет и дополняет функциональные возможности Программного Комплекса "КАТАРСИС". Программный продукт является инструментом анализа сведений о порядке и результате работы с гражданами в центрах занятости населения региона на основе формируемого регионального хранилища данных, содержащего переработанные и упорядоченные данные из Программного Комплекса "КАТАРСИС". Высокая скорость обработки данных в системе достигается за счет использования OLAP-технологий анализа данных. Кроме всего перечисленного, АС "Мониторинг рынка труда" является системой поддержки принятия решений, в связи с чем, система наиболее востребована в управлениях региональных служб занятости населения и крупных городских центрах занятости населения.

В основе системы два подхода к анализу данных:

-Списки личных дел (вакансий, работодателей), отобранные по специальным условиям с возможностью просмотра их карточек. В системе имеется несколько сотен готовых запросов к данным. Пользователь задает значения параметрам запроса и получает требуемый список для анализа;

– Аналитические отчеты. Подход основан на современной технологии многомерного анализа больших массивов данных OLAP. Отчеты формируются буквально за несколько кликов мышью. Пользователь выбирает аналитические показатели, указывает, в каких разрезах следует поместить в отчет их значения и тут же получает готовый отчет[9].

1.3 Обоснование необходимости автоматизации и цели создания подсистемы поиска вакансий в центре занятости населения

Анализ деятельности Белгородского центра занятости показал, что ежедневно по проблемам трудоустройства обращается большое количество населения. Однако существуют проблемы своевременного информирования о наличии рабочих мест, что вызвало необходимость создания подсистемы поиска вакансий в центре занятости населения.

В ходе разработки автоматизированной подсистемы была детализирована работа центра занятости населения с применением стандарта IDEF0. На рисунке 1.2 представлена контекстная диаграмма «как есть».

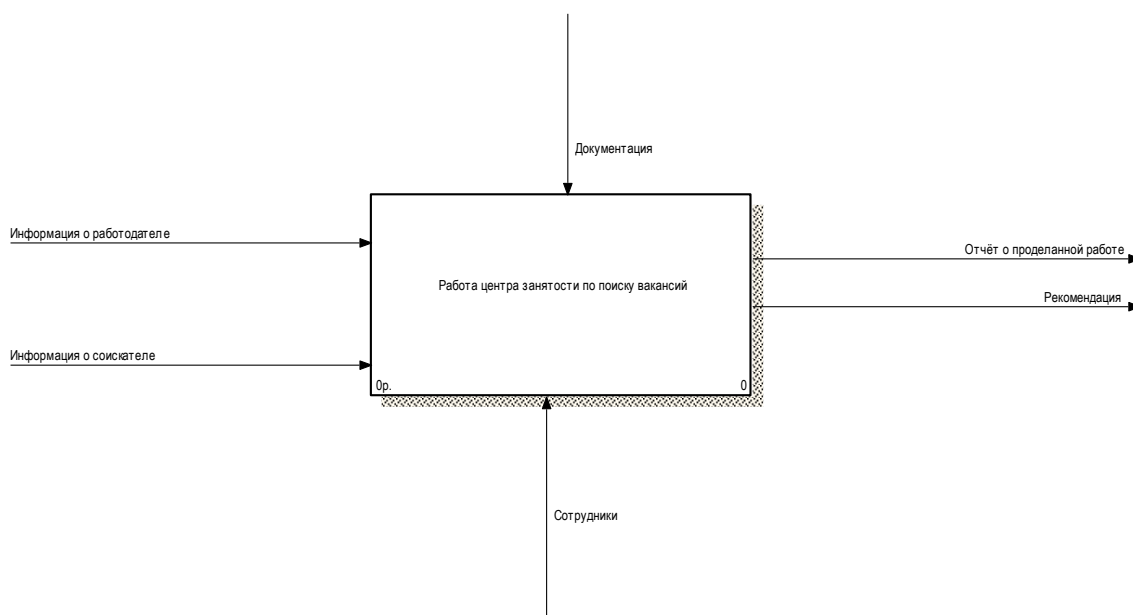


Рисунок 1.2 – IDEF0-диаграмма A0 – контекстная диаграмма работы центра занятости по поиску вакансий

Декомпозируем контекстную диаграмму на 4 функциональных блока:

- «Сбор информации от клиента»;
- «Предоставление вариантов клиенту»;
- «Подготовка к собеседованию»;
- «Проведение собеседования».

На рисунке 1.3 представлена IDEF0-диаграмма A1 – контекстная декомпозиция подсистемы поиска вакансий в центре занятости населения.

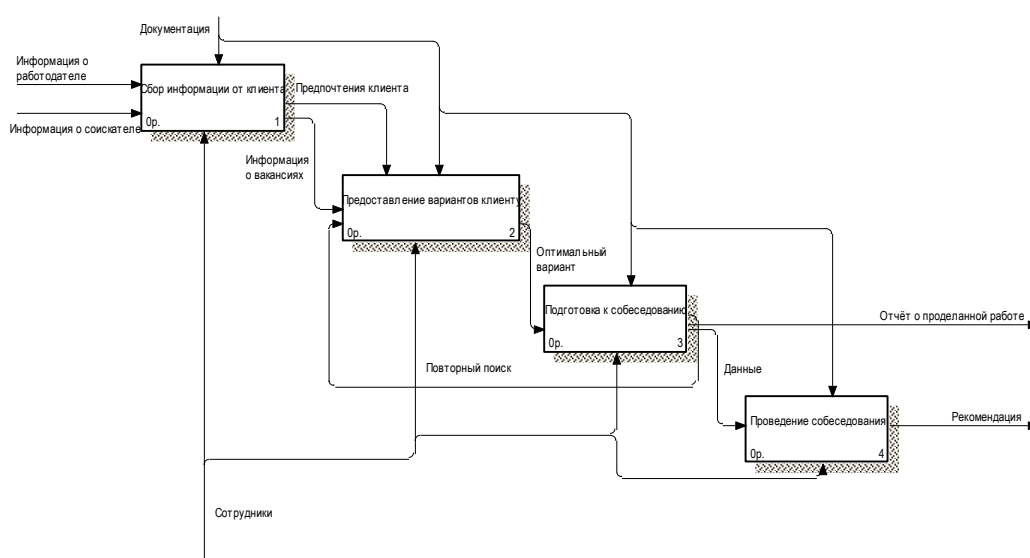


Рисунок 1.3 – IDEF0-диаграмма A1 – контекстная декомпозиция подсистемы поиска вакансий в центре занятости населения

Декомпозируем функциональный блок «Сбор информации от клиента» еще на 3 действия:

- Распределение клиентов;
- Обработка информации;
- Работа с информацией от клиента.

На рисунке 1.4 продемонстрирована IDEF0-диаграмма A2 – контекстная декомпозиция «Сбор информации от клиента».

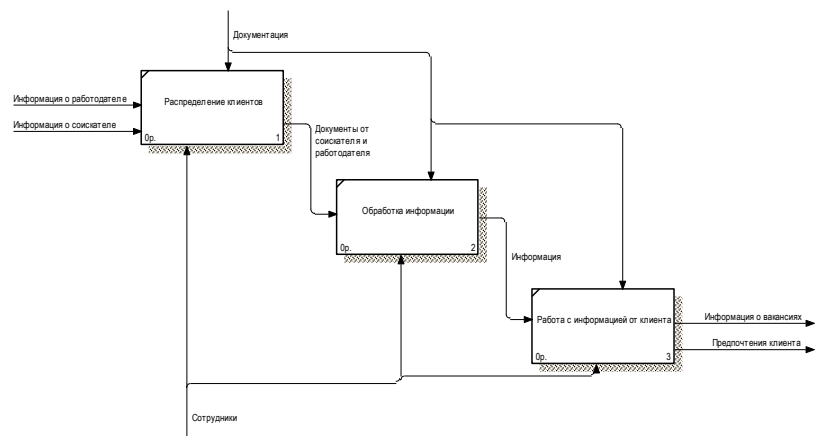


Рисунок 1.4 – IDEF0-диаграмма A2 – контекстная декомпозиция «Сбор информации от клиента»

На данном рисунке видно, что в блок «Распределение клиентов» входит информация от работодателя и информация от кандидата на работу, далее происходит обработка информации, потом сотрудники работают с информацией клиента

На рисунке 1.5 продемонстрирована IDEF0-диаграмма A2 – контекстная декомпозиция «Предоставление вариантов клиенту».

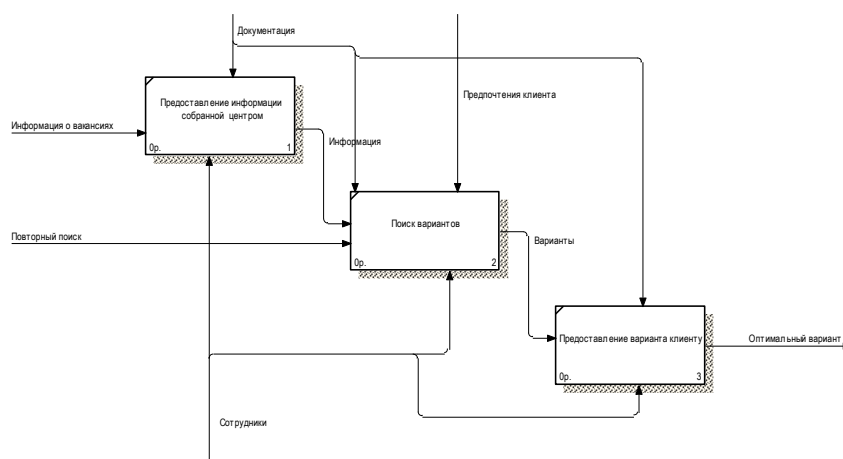


Рисунок 1.5 – IDEF0-диаграмма A2 – контекстная декомпозиция «Предоставление вариантов клиенту»

Входящей информацией в блок «Предоставление информации собранной центром» является «Информация о вакансиях», далее информация

переходит в блок «Поиск вариантов» в дальнейшем варианты поступают в блок «Предоставление варианта клиенту», а из данного блока выходящей стрелкой является «Оптимальный вариант»

На рисунке 1.6 продемонстрирована IDEF0-диаграмма A2 – контекстная диаграмма декомпозиция «Подготовка к собеседованию».

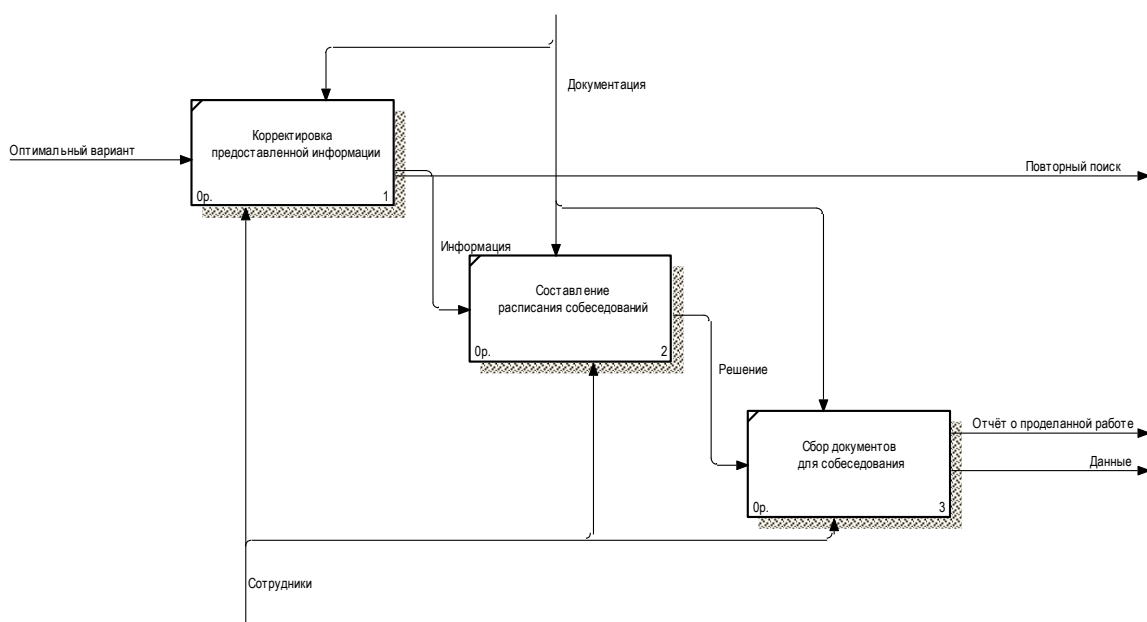


Рисунок 1.6 – IDEF0-диаграмма A2 – контекстная диаграмма декомпозиция «Подготовка к собеседованию»

Входящей информацией в блок «Корректировка предоставленной информации» является «Оптимальный вариант» затем происходит составление расписания собеседования, а последним этапом в подготовке к собеседованию является сбор документов для собеседования, выходящими стрелками из этого блока предусмотрены «Данные» и «Отчёт о проделанной работе».

На рисунке 1.7 продемонстрирована IDEF0-диаграмма A2 – контекстная декомпозиция «Проведение собеседования».

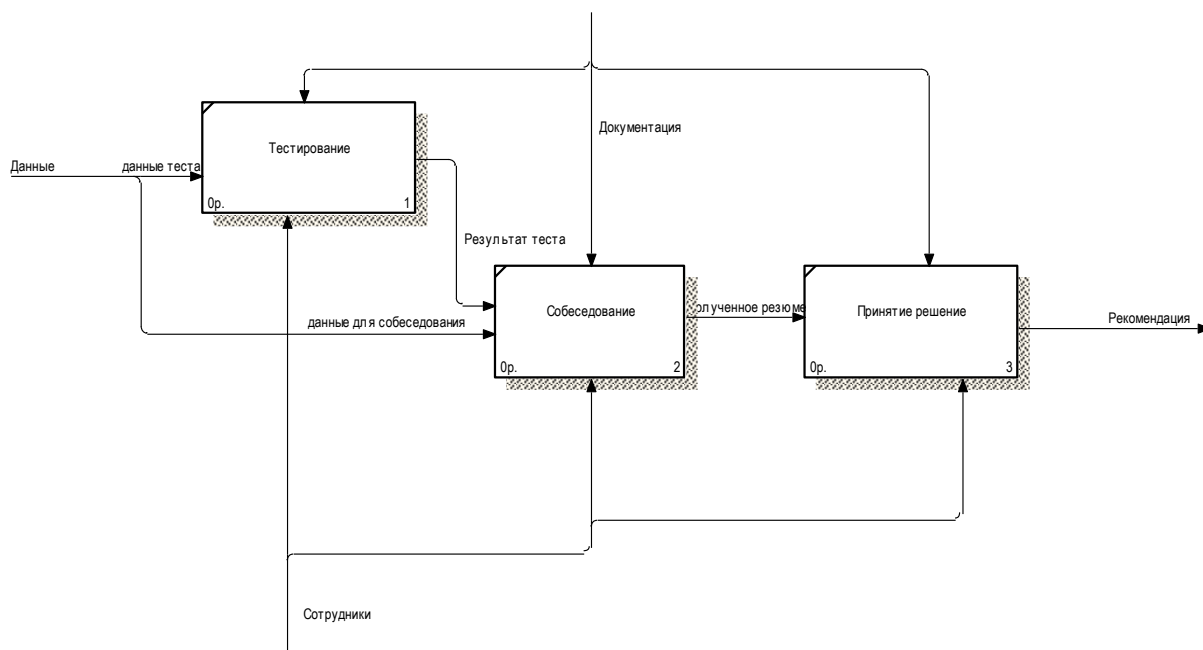


Рисунок 1.7 – IDEF0-диаграмма A2 – контекстная декомпозиция «Проведение собеседования»

Проведение собеседования делится на 3 блока:

- «Тестирование»;
- «Собеседование»;
- «Принятие решения».

Входящей информацией являются данные, необходимые для тестирования, после прохождения теста происходит опрос результат, которого влияет на принятие решения, в итоге формируется рекомендация.

Далее декомпозируем деятельность центра занятости населения, с учётом внедрения подсистемы поиска вакансий.

На рисунке 1.8 продемонстрирована IDEF0-диаграмма A2 – контекстная диаграмма работы центра занятости населения с учётом внедрения подсистемы поиска вакансий.

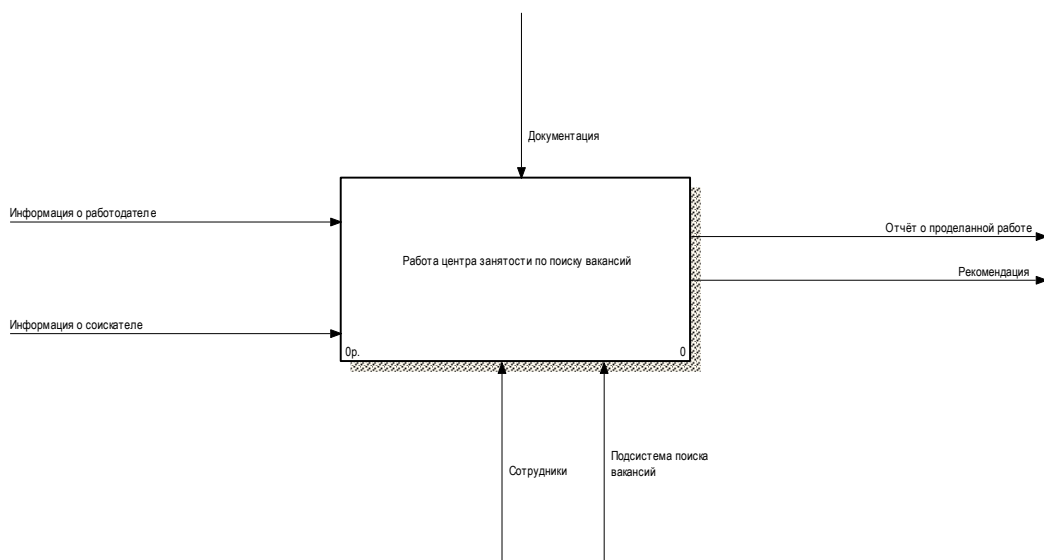


Рисунок 1.8 – IDEF0-диаграмма A2 – контекстная диаграмма работы центра занятости населения с учётом внедрения подсистемы поиска вакансий

На рисунке 1.9 продемонстрирована IDEF0-диаграмма A0 – декомпозиция работы в центре занятости с учётом внедрения.

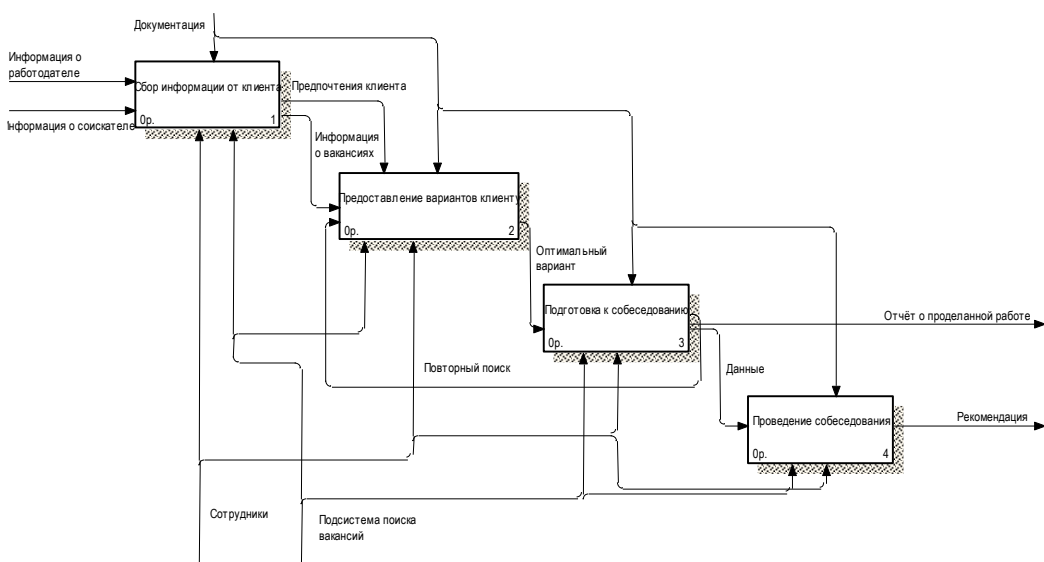


Рисунок 1.9 – IDEF0-диаграмма A0 – декомпозиция работы в центре занятости с учётом внедрения

На рисунке 1.9 видно, что подсистема поиска вакансий присутствует во всех блоках декомпозиции, из чего следует, что с ее помощью можно

сократить затраты на время работы с клиентом, а также сократить трудозатраты при поиске вакансий.

На рисунке 1.10 продемонстрирована IDEF0-диаграмма A0 – декомпозиция «Сбор информации от клиента».

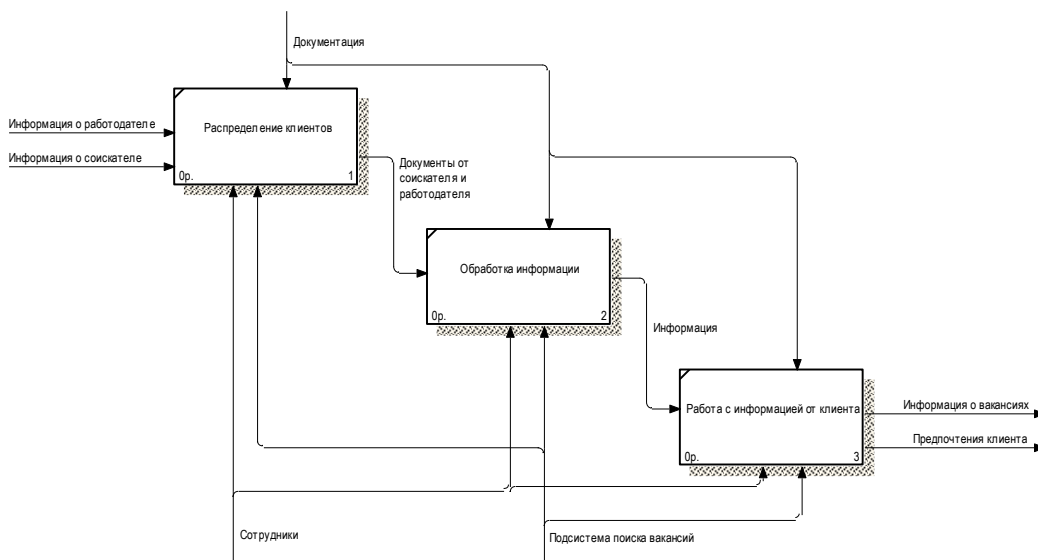


Рисунок 1.10 – IDEF0-диаграмма A0 – декомпозиция «Сбор информации от клиента»

На рисунке 1.10 видно, что с помощью подсистемы поиска вакансии происходит распределение клиентов, далее следует обработка информации представленной клиентом, затем с помощью подсистемы сотрудники работают с информацией от клиента.

На рисунке 1.11 IDEF0-диаграмма A0 – декомпозиция «Предоставление вариантов клиенту».

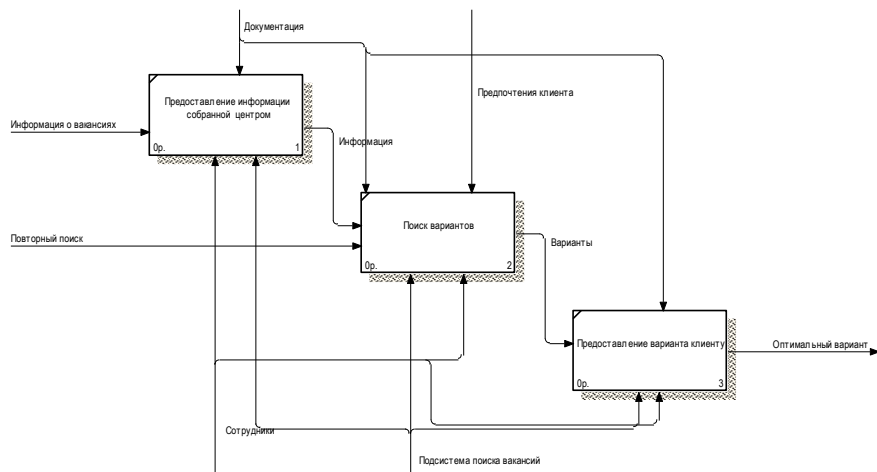


Рисунок 1.11 – IDEF0-диаграмма A0 – декомпозиция «Предоставление вариантов клиенту»

На рисунке 1.11 видно, что при помощи программного обеспечения предоставляют информацию собранную центром занятости по вакансиям, поиск вариантов происходит в подсистеме, далее после работы с подсистемой происходит предоставление вариантов клиенту.

На рисунке 1.12 продемонстрирована IDEF0-диаграмма A0 – декомпозиция «Подготовка к собеседованию».

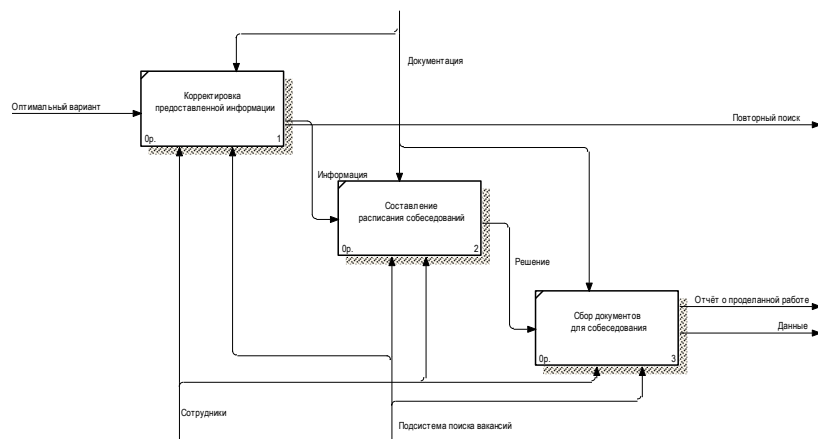


Рисунок 1.12 – IDEF0-диаграмма A0 – декомпозиция «Подготовка к собеседованию»

На рисунке 1.12 показано, что в подсистеме производится корректировка предоставленной информации, затем составляется расписание собеседование на основе данных в подсистеме.

На рисунке 1.13 продемонстрирована IDEF0-диаграмма A0 – декомпозиция «Проведение собеседования»

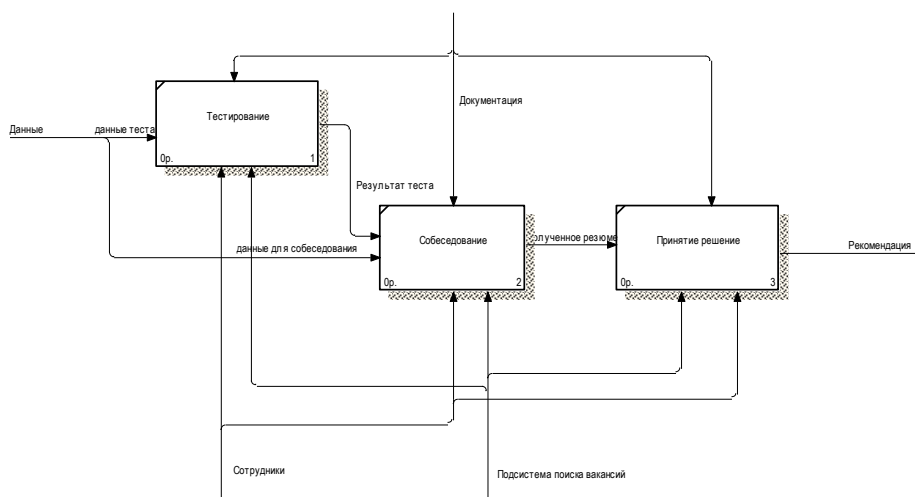


Рисунок 1.13 – IDEF0-диаграмма A0 – декомпозиция «Проведение собеседования»

На рисунке 1.13 показано, что тестирование проходит с помощью подсистемы поиска вакансий, впоследствии результаты теста хранятся в подсистеме поиска вакансий и данные используются при собеседовании.

Выводы по первому разделу:

Как показывает изучение, и анализ использования информационных систем в центре занятости населения информационная поддержка трудоустройства граждан осуществляется за счёт внедрения современных автоматизированных программ:

- развитие электронных услуг позволяет экономить время, трудовые ресурсы, расширяет возможности трудоустройства граждан. Всё это в целом обеспечивает решение актуальных задач государственной и региональной политики в сфере занятости населения;

- предпосылкой создания автоматизированной подсистемы поиска вакансий послужили существующие проблемы;

- преимуществами разработанной подсистемы поиска вакансий в центре занятости населения является:

- 1) доступность для людей с ограниченными возможностями;
- 2) уменьшение времени работы с клиентом;
- 3) меньшая занятость персонала;
- 4) быстрый доступ к базе данных вакансий или работников.

2 ОБОСНОВАНИЕ ПРОГРАММНЫХ РЕШЕНИЙ АВТОМАТИЗИРОВАННОЙ ПОДСИСТЕМЫ ПОИСКА ВАКАНСИЙ

2.1 Техническое и кадровое обеспечение подсистемы поиска вакансий

Для успешного функционирования подсистемы поиска вакансий в Белгородском казенном учреждении необходимо обеспечить порядок взаимодействия персонала центра занятости, работодателей и клиентов, так как они являются основными пользователями подсистемы. Исходя из штатного расписания в состав персонала, необходимого для обеспечения эксплуатации автоматизированной подсистемы поиска вакансий в центре занятости населения в рамках соответствующих подразделений, необходимо выделение следующих ответственных лиц:

- администратор базы данных - 1 человек;
- отдел профессиональной подготовки - 1 человек;
- отдел взаимодействия с работодателями и технологии устройства - 1 человек;
- отдел содействия занятости населения - 1 человек;
- отдел информационных технологий 1 человек.

Данные лица должны выполнять следующие функциональные обязанности.

Администратор базы данных – на всем протяжении функционирования подсистемы следить и обеспечивать работоспособность базы данных.

Отдел содействия занятости населения – на всем протяжении функционирования подсистемы редактирование и внесение данных для тестирования[10].

Отдел взаимодействия с работодателями и технологии устройства – на всем протяжении функционирования подсистемы обеспечивает внесение данных от возможных работодателей и работа с данными работодателя.

Отдел информационных технологий – на всем протяжении функционирования подсистемы обеспечивает обслуживание внутренней информационной системы и администрирование сети.

Отдел профессиональной подготовки на всем протяжении функционирования подсистемы обеспечивает подготовку соискателей к работе с подсистемой поиска вакансий.

Как мы отметили выше техническое обеспечение системы осуществляется персоналом исходя из функциональных обязанностей. Для защиты от ошибочных действий персонала нами предусмотрена система подтверждения пользователя при просмотре данных, разработано полное и доступное руководство пользователя.

Уровень надежности достигается согласованным применением организационных, организационно-технических мероприятий и программно-аппаратных средств.

Надежность подсистемы поиска вакансий обеспечивается применением технических средств, системного и базового программного обеспечения, соответствующих классу решаемых задач, а также своевременным выполнением процессов администрирования системы КХД.

Безусловна при соблюдении правил эксплуатации и технического обслуживания программно-аппаратных средств повышается ее эффективность[11].

Требуется также предварительное обучение пользователей и обслуживающего персонала. О чем говорилось выше.

Нами предусмотрены меры предупреждения экстренных ситуаций

Время устранения отказа будет следующим:

– при перерыве и выходе за установленные пределы параметров электропитания - не более 15 минут;

- при перерыве и выходе за установленные пределы параметров программного обеспечения - не более 2 часов;

- при выходе из строя АПК ХД - не более 5 часов.

Система соответствует следующим параметрам:

- среднее время восстановления 1 час - определяется как сумма всех времен восстановления за заданный календарный период, поделенные на продолжительность этого периода;

- коэффициент готовности W - определяется как результат отношения средней наработки на отказ к сумме средней наработки на отказ и среднего времени восстановления;

- время наработки на отказ 115 часов - определяется как результат отношения суммарной наработки Системы к среднему числу отказов за время наработки.

Средняя наработка на отказ АПК не должна быть меньше 50 часов.

Под аварийной ситуацией понимается аварийное завершение процесса, выполняемого той или иной подсистемой КХД, а также «зависание» этого процесса[12].

При работе системы возможны следующие аварийные ситуации, которые влияют на надежность работы системы:

- сбой в электроснабжении сервера;

- сбой в электроснабжении рабочей станции пользователей системы;

- сбой в электроснабжении обеспечения локальной сети (поломка сети);

- ошибки Системы ХД, не выявленные при отладке и испытании системы;

- сбои программного обеспечения сервера.

Эффективность эксплуатации подсистемы поиска вакансий во многом определяется надежностью оборудования. К нему предъявляются следующие требования:

- в качестве аппаратных платформ используются средства с повышенной надежностью;

- применение технических средств соответствующих классу решаемых задач.

К надежности электроснабжения предъявляются следующие требования:

- с целью повышения отказоустойчивости системы в целом необходима обязательная комплектация серверов источником бесперебойного питания с возможностью автономной работы системы;

- подсистема укомплектована оповещением Администраторов о переходе на автономный режим работы;

- обеспечено бесперебойное питание активного сетевого оборудования.

Надежность аппаратных и программных средств подсистемы поиска вакансий обеспечивается за счет следующих организационных мероприятий:

- предварительного обучения пользователей и обслуживающего персонала;

- своевременного выполнения процессов администрирования;

- соблюдения правил эксплуатации и технического обслуживания программно-аппаратных средств;

- своевременное выполнение процедур резервного копирования данных.

Таким образом, надежность программного обеспечения подсистем обеспечивается с помощью общесистемного ПО и разрабатываемого ПО, а также проведением комплекса мероприятий отладки, поиска и исключения ошибок. Необходимо ведение журналов системных сообщений и ошибок по подсистемам для последующего анализа и изменения конфигурации.

Тестирование по проверке выполнения требований надежности проведено на этапе проектирования расчетным путем, а на этапах испытаний и эксплуатации проверена ее работоспособность[13].

Обратимся к технической характеристике подсистемы.

Интерфейс обеспечивает удобную для конечного пользователя подсистему формирования и визуализации отчетности данных она отвечает следующим требованиям[14].

В части внешнего оформления интерфейсы подсистем типизированы, обеспечено наличие локализованного (русскоязычного) интерфейса пользователя[15].

В части процедур ввода-вывода данных:

– реализована возможность многомерного анализа данных в табличном виде.

Условия эксплуатации, а также виды и периодичность обслуживания технических средств системы соответствуют требованиям по эксплуатации, техническому обслуживанию, ремонту и хранению, изложенным в документации завода-изготовителя (производителя) на них[16].

Технические средства Системы и персонал размещаются в существующих помещениях. Внешняя среда по климатическим условиям должна соответствовать ГОСТ 15150-69 «Машины, приборы и другие технические изделия. Организация автоматизированных рабочих мест должна соответствовать требованиям ГОСТ 21958-76 «Система "Человек-машина". Общим эргономическим требованиям должны отвечать зал и кабины операторов, взаимное расположение рабочих мест.

Предусмотрена трехфазная четырехпроводная сеть с глухо заземленной нейтралью 380/220 В (+10-15)% частотой 50 Гц (+1-1) Гц. для электропитания технических средств. Через сетевые розетки с заземляющим контактом каждое техническое средство питается однофазным напряжением 220 с частотой 50 Гц[17].

Для обеспечения выполнения требований по надежности в центре занятости населения создан комплект запасных изделий и приборов (ЗИП).

Техническое обеспечение персонального компьютера, необходимое для успешного функционирования подсистемы перечислено в таблице 1.

Таблица 1 - Характеристика технического обеспечения персонального компьютера

Наименование	Характеристика
Материнская плата M5A99X EVO R2.0	ASUS ATXM5A99X EVO R2
Процессор	AMD Phenom x4 3.4 ghz
Жесткие диски	HDD IDE Western digital green 2tb
Модули памяти. Flash-накопители	DDR-память: DIMM DDR3 Kingston 1333 ghz Flash-карты: SD Card Transcend 8gb
Видеокарты	ASUS Nvidia Geforce gtx 460 SE 1 gb
Звуковая карта	C-Media 8738 4-channel PCI(oem)
Приводы CD-R/RW	CD-RW+DVD Teac 52*32*52*16 DW-552G(oem)
Приводы DVD-ROM/RW	DVD-RW/+RW NEC ND-2500A(oem)
Дисководы	Fdd 3.5" 1.44 Mb NEC black
Корпуса	AeroColler Strike X
Модемы	Zyxel keenetic lite ii
Клавиатура	Genius KB-110 usb 2.0
Мышь оптическая.	лазерная мышь logitech m100 black usb 2.0
Колонки.	Genius SP-S110U Black
Сетевой фильтр	Сетевой фильтр (5 розеток) Pilot GL 2м
Монитор	19" Samsung syncMaster LS24HUBCBL
Кабели	Кабель FireWire IEEE 1394 4p4p 1.9 м
Расходные материалы	Аксессуары. Диски CD-RW бумага

Сервер базы данных развернут на HP9000 SuperDome №1, минимальная конфигурация которого: CPU: 16 (32 core); RAM: 128 Gb; HDD: 500 Gb; Network Card: 2 (2 Gbit); Fiber Channel: 4.

Сервер сбора, обработки и загрузки данных развернут на HP9000 SuperDome №2, минимальная конфигурация которого:

CPU: 8 (16 core); RAM: 32 Gb; HDD: 100 Gb; Network Card: 2 (1 Gbit); Fiber Channel: 2.

Сервер приложений развернут на платформе HP Integrity, минимальная конфигурация которого: CPU: 6 (12 core); RAM: 64 Gb; HDD: 300 Gb; Network Card: 3 (1 Gbit).

Приведенные сервера подключены к дисковому массиву HP XP с организацией сети хранения данных. Минимальный объем свободного пространства для хранения данных на дисковом массиве составляет 5 Тб.

Таким образом, для успешного использования программного продукта необходимо обеспечить кадровые организационные и технические условия, перечисленные в этом параграфе [18].

2.2 Характеристика функций подсистемы поиска вакансий

Исходя из целей и задач, предоставленных в пункте 1.3. при проектировании подсистемы поиска вакансий в центре занятости населения для обеспечения взаимодействия частей подсистемы, подлежащих автоматизации был определен комплекс функций и задач, который представлен в таблицах 2-4.

В этом пункте приводятся перечень задач, функций или их комплексов (в том числе обеспечивающих взаимодействие частей подсистемы), которые подлежат автоматизированию. При создании подсистемы в две или более очереди - отдельных задач или функций, вводимых в действие в 1-й и последующих очередях.

В таблице 2 описаны функции и задачи, реализуемые в подсистеме поиска вакансий.

Таблица 2 - Функции и задачи, реализуемые в подсистеме поиска вакансий

Функция	Задача
Сбор информации от клиента	Распределение клиентов
	Сбор информации от определенного клиента
	Работа с информацией клиента
Предоставление вариантов клиенту	Предоставление информации
	Поиск вариантов интересующих клиента
	Предоставление данных
Предоставление данных выбранного варианта	Ведение журналов клиентов
	Отбор информации согласно распределению
Проведение собеседования	Тест
	Опрос

Временной регламент реализации каждой функции, задачи (или комплекса задач).

Требования к качеству реализации каждой функции (задачи или комплекса задач), форме представления выходной информации, характеристики необходимой точности и времени выполнения, одновременность выполнения групп функций, достоверности выдачи результатов.

В таблице 3 продемонстрированы задачи и временный регламент для подсистемы.

Таблица 3 - Задачи и временный регламент для подсистемы

Задача	временный регламент
1	2
Распределение клиентов	Весь период функционирования системы, при возникновении необходимости изменения процессов сбора, обработки и загрузки данных
Сбор информации от определенного клиента	Весь период функционирования системы, при возникновении необходимости модификации регламента загрузки данных
Работа с информацией клиента	Весь период функционирования системы, при возникновении необходимости изменения расписания процессов
Предоставление информации	Ежедневно, после появления всех извлечённых данных
Поиск вариантов интересующих клиента	Регулярно, при работе подсистемы
Предоставление данных	Весь период функционирования системы, при возникновении необходимости изменения процессов сбора, обработки и загрузки данных
Ведение журналов клиентов	Весь период функционирования системы, при возникновении необходимости изменения процессов сбора, обработки и загрузки данных
Отбор информации согласно распределению	Период функционирования системы, при возникновении необходимости изменения процессов сбора, обработки и загрузки данных
Тест	Весь период функционирования системы, при возникновении необходимости изменения процессов сбора, обработки и загрузки данных
Опрос	Весь период функционирования системы, при возникновении необходимости изменения процессов сбора, обработки и загрузки данных

При формировании задач необходимо также описать их реализацию форму представления выходной информации, также необходимо дать пояснение о характеристике точности и времени выполнения.

В таблице 4 даны пояснения для реализации функций и задач.

Таблица 4 - Реализация задач, их формы представления и формы представления выходной информации

Задача	Форма представления выходной информации	Характеристики точности и времени выполнения
1	2	3
Создание, редактирование и удаление процессов сбора, обработки и загрузки данных	В стандарте интерфейса ETL средства	Определяется регламентом эксплуатации
Формирование последовательности выполнения процессов сбора, обработки и загрузки данных (регламентов загрузки данных)	В стандарте интерфейса ETL средства	Определяется регламентом эксплуатации
Определение и изменение расписания процессов сбора, обработки и загрузки данных	В стандарте интерфейса ETL средства	Определяется регламентом эксплуатации
Обработка и преобразование извлечённых данных	Текстовый файл. Данные в структурах БД	Данные преобразованы для загрузки в структуры модели ХД. Не более 2 часов
Поддержка медленно меняющихся измерений	Данные в структурах БД	Данные сохранены по правилам поддержки медленно меняющихся измерений соответствующего типа
Ведение журналов результатов сбора, обработки и загрузки данных	Текстовые файлы	В момент выполнения сбора, обработки и загрузки данных
Оперативное извещение пользователей о всех нештатных ситуациях в процессе работы подсистемы	Текстовый файл, оконное сообщение, email	Не позднее 15 минут после возникновения нештатной ситуации

Таким образом выделение функций и задач будет влиять на уменьшение затрат времени работы с подсистемой, повышает эффективность при работе с клиентами, увеличивает продуктивность взаимодействия с клиентами центра занятости.

2.3 Информационное обеспечение подсистемы поиска вакансий

При проектировании подсистемы поиска вакансий было предусмотрено то, что области постоянного хранения и витрин данных строятся на основе многомерной модели данных, подразумевающей выделение отдельных измерений и фактов с их анализом по выбранным измерениям.

Структура хранения данных в подсистеме поиска вакансий состоит из следующих основных областей:

- область временного хранения данных;
- область постоянного хранения данных;
- область витрин данных.

Информационный обмен между компонентами подсистемы поиска вакансий реализован следующим образом[19].

Таблица 5 - Информационный обмен между компонентами

	сбор, обработка и загрузка данных	хранение данных	формирование и визуализации отчетности
сбор, обработка и загрузка данных		X	
хранение данных	X		X
формирование и визуализация отчетности		X	

Состав данных для осуществления информационного обмена по каждой смежной системе определен на стадии проектирования. подсистема не закрыта для смежных систем и поддерживает возможность экспорта данных в смежные системы через интерфейсные таблицы или файлы данных.

Для реализации хранения данных используется промышленная СУБД Firebird .

Процесс сбора, обработки и передачи данных в системе определяется регламентом процессов сбора, преобразования и загрузки данных, разрабатываемом при проектировании. Информация в базе данных системы сохраняется при возникновении аварийных ситуаций, связанных со сбоями электропитания.

Система имеет бесперебойное электропитание, обеспечивающее её нормальное функционирование в течение 15 минут в случае отсутствия внешнего энергоснабжения, и 5 минут дополнительно для корректного завершения всех процессов.

Резервное копирование данных осуществляется на регулярной основе, в объёмах, достаточных для восстановления информации в подсистеме.

Контроль данных происходит следующим образом, подсистема протоколирует все события, связанные с изменением своего информационного наполнения, и имеет возможность в случае сбоя в работе восстанавливать свое состояние, используя ранее запротоколированные изменения данных.

Хранение данных:

- хранение исторических данных в системе производится не более чем за 5 (пять) предыдущих лет. По истечению данного срока данные должны переходить в архив;

- исторические данные, превышающие пятилетний порог, хранятся на ленточном массиве с возможностью их восстановления.

К обновлению и восстановлению данных:

- для сервера сбора, обработки и загрузки данных обеспечено резервное копирование его бинарных файлов (None) раз в 2 недели и хранение копии на протяжении 2-х месяцев;

- для сервера базы данных обеспечено резервное копирование его бинарных файлов раз в 2 недели и хранение копии на протяжении 2-х месяцев;

– для данных хранилища данных обеспечено резервное копирование и архивацию на ленточный массив в следующие промежутки времени:

- 1) холодная копия - ежеквартально;
- 2) логическая копия - ежемесячно (конец месяца);
- 3) инкрементальное резервное копирование - еженедельно (воскресение);
- 4) архивирование – ежеквартально.

В модели данных подсистемы поиска вакансий используется язык определения данных. Посредством ЯОД определяются подразделения данных, типовые структуры и правила их композиции. А также присваиваются имена данным, определяются типы элементов данных посредством задания присущих им свойств, учреждаются ключи базы данных. С помощью ЯОД определяются отношения между данными, упорядоченность данных внутри их совокупностей, правила проверки достоверности данных и замки защиты от неправомерного использования их.

Ниже в таблице приведены характеристики информационного обеспечения.

Таблица 6 - Характеристики информационного обеспечения

Наименование	Характеристика
Язык определения данных (ЯОД, DDL)	Формальный язык, используемый в модели данных для определения структуры баз данных
Язык управления данными (ЯУД, DML)	Совокупность языковых средств для организации доступа к данным в модели и соответствующей СУБД
Словарь данных (системный каталог)	Специальная система в составе БД, содержащая информацию обо всех ресурсах системы
Нормативно-справочная информация	Устав центра занятости населения

Язык управления данными – представляет собой совокупность языковых средств для организации доступа к данным в модели и соответствующей СУБД. Имеет возможность выступать в образе языка

запросов, обеспечивающий информационное обслуживание пользователей хранилища данных, или же быть расширением некоторого языка программирования, называемого базовым (включающим) языком, с конструкциями и понятиями которого ЯУД должен быть согласован.

Словарь данных (системный каталог) – специальная система в составе БД, содержащая информацию обо всех ресурсах системы. В словаре данных интегрированы метаданные – данные об объектах базы данных, позволяющие упростить способ доступа к ним и управление ими. Перед доступом к реальным данным СУБД обращается к системному каталогу.

Состав глоссария данных:

- База данных: состоит из названия базы данных и имени файла;
- Описание назначения и общего содержания БД, а также лиц, которые могут ею пользоваться. Список приложений, работающих с базой, информация о других БД, использующих данные из этой базы. Если имеются, то сюда же включаются диаграммы БД.
- Область данных: название группы, к которой принадлежат таблицы;
- Таблица: таблицы, входящие в область данных;
- Доступ: права пользователей на доступ к таблице;
- Запись: общее определение элементов данных;
- Первичный ключ: поле (поля) первичного ключа;
- Индекс: описание индекса первичного ключа;
- Внешние ключи: внешние ключевые поля;
- Индекс: индексы внешних ключей.

Следовательно, важным условием успешного функционирования подсистемы поиска вакансий является ее информационное обеспечение[20].

2.4. Программное обеспечение подсистемы поиска вакансий

Для программного обеспечения подсистемы использованы покупные программные средства:

Перечень программных средств включает в себя:

- Firebird;
- C++ Builder 6;
- Allfusion process modeler 7.3 bpwin;
- Allfusion Erwin data modeler r7;
- IbExpert.

СУБД имеет возможность установки на ОС Windows 7x64.

ETL-средство имеет возможность установки на ОС Windows 7x64.

BI-приложение имеет возможность установки на ОС Windows 7x64.

Качество программных средств обеспечивает функциональность, надежность, эффективность функционирования подсистемы поиска вакансий

Рассмотрим, как осуществляется обеспечение качества программных средств[21].

Функциональность обеспечивается выполнением подсистемы всех ее функций:

- надежность обеспечивается с помощью предупреждения ошибок
- предотвращения ошибок в готовых программных средствах;
- легкость применения обеспечивается за счет применения покупных программных средств;
- эффективность обеспечивается за счет принятия подходящих, верных решений на разных этапах разработки программных средств и подсистемы в целом;
- сопровождаемость обеспечивается за счет высокого качества документации по сопровождению, а также за счет использования в программном тексте описания объектов и комментариев; использованием

осмысленных (мнемонических) и устойчиво различимых имен объектов; размещением не больше одного оператора в строке текста программы; избеганием создания фрагментов текстов программ с неочевидным или скрытым смыслом.

Также на каждом этапе в разработке программных средств проводится проверка правильности принятых решений по разработке и применению готовых ПС.

Необходимость согласования вновь разрабатываемых программных средств с фондом алгоритмов и программ отсутствует[22].

В таблице 7 приведены наименование и характеристики используемых программных средств.

Таблица 7 - Наименование и характеристики используемых средств

Наименование	Характеристика
Операционная система	Microsoft Windows 7 x64: Стандартные программы. Служебные программы. Связь и развлечения. Специальные возможности
Инструментальное ПО	C++Builder 6.0, Process Modeler v7; IBExpert Data Modeler
Пакет прикладных программ	Microsoft Office: Microsoft Office Excel; Microsoft Office Word;
Архиватор	WinRAR, WinZIP
Информационно-поисковые системы	Yandex

Анализ программного обеспечения свидетельствует о том, что предпочтения отдается стандартным приложениям Microsoft Office, где все данные можно обрабатывать в табличном редакторе Excel, а также вести БД в Access, не утруждая себя. Таким образом, с помощью программных средств нам удалось создать новый программный продукт – подсистему поиска вакансий, в котором разработан специальный пользовательский интерфейс.

3 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

3.1 Создание базы данных для подсистемы

Для оптимальной работы подсистемы необходимо создать базу данных, которая будет поддерживаться персоналом центра занятости населения.

База данных - организованная в соответствии с определёнными правилами и поддерживаемая в памяти компьютера совокупность данных, характеризующая актуальное состояние некоторой предметной области и используемая для удовлетворения информационных потребностей пользователей. Для функционирования объекта хранения информации необходимо создать модель базы данных.

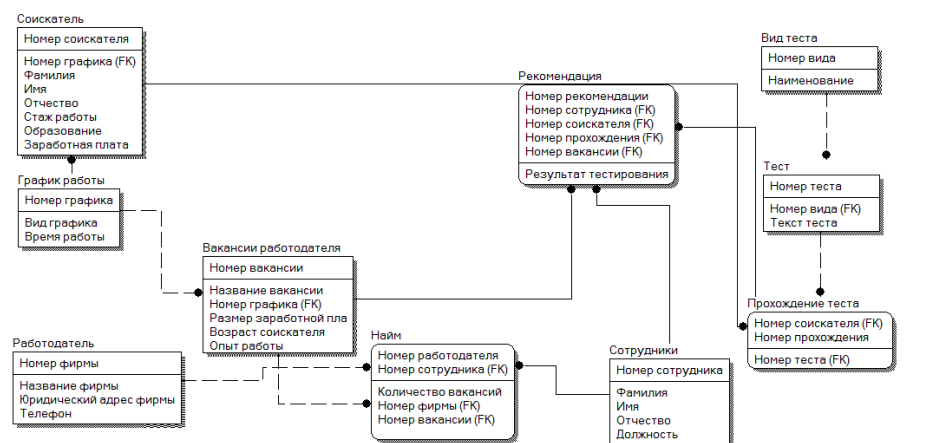


Рисунок 3.1 – Логическая модель базы данных в подсистеме поиска вакансий

Сущности базы данных:

- соискатель – хранится информация о соискателе работы, обратившемся в центр занятости населения;
- график работы – хранится информация о графике работы приемлемом для соискателя и требуемый для вакансии;
- работодатель – хранится информация о фирме, которая ищет соискателя по нужной вакансии;

- вакансии работодателя – хранится информация о доступных вакансиях, требуемых для работодателя;
- вид теста – в данной сущности хранится информация о виде теста, которые можно пройти в центре занятости населения;
- тест – хранится информация для прохождения теста;
- прохождение теста – хранится информация о соискателе и о тесте, который он проходил;
- сотрудники – хранится информация о сотрудниках, которые работают в центре занятости;
- найм – хранится информация о количестве вакансий необходимых для найма работодателю;
- рекомендация – хранится информация о рекомендации центра занятости по определенному соискателю.

На основе логической модели данных создаётся реляционная модель базы.

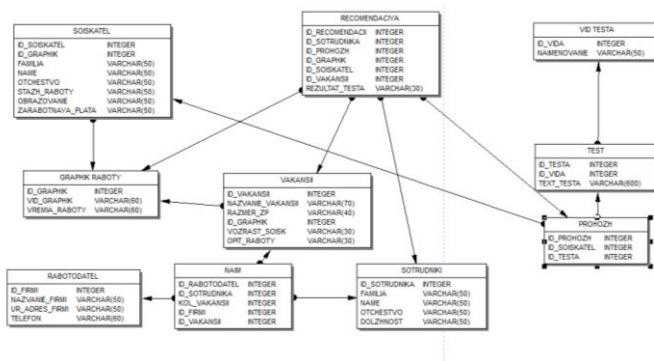


Рисунок 3.2 – Реляционная модель базы данных

Далее с помощью программного обеспечения «IbExpert» создали таблицы с нужными параметрами, генераторами и триггерами. На рисунке показана созданная таблица «Соискатель»

#	PK	BK	UNQ	Название	Тип	Домен	Длина	Точ...	Подтип	Массив	Не ну...	Кодировка	Коллате
1	1			ID_SOISKATEL	INTEGER						<input checked="" type="checkbox"/>		
2				ID_GRAPHIK	INTEGER						<input checked="" type="checkbox"/>		
3				FAMILIA	VARCHAR		50				<input type="checkbox"/>	WIN1251	WIN1251
4				NAME	VARCHAR		50				<input type="checkbox"/>	WIN1251	WIN1251
5				OTCHESTVO	VARCHAR		50				<input type="checkbox"/>	WIN1251	WIN1251
6				STAZH_RABOTY	VARCHAR		50				<input type="checkbox"/>	WIN1251	WIN1251
7				OBRAZOVANIE	VARCHAR		50				<input type="checkbox"/>	WIN1251	WIN1251
8				ZARABOTNAYA_PLATA	VARCHAR		50				<input type="checkbox"/>	WIN1251	WIN1251

Рисунок 3.3 – Созданная таблица «Соискатель»

На рисунке продемонстрирована таблица «Соискатель» с первичным ключом «Id soiskatel», также в таблице хранятся данные о фамилии, имени, отчестве, стаже работы, образовании и заработной плате соискателя.

Впоследствии были созданы другие объекты, которые необходимы для разработки подсистемы. В таблице 8 описаны сущности, первичные ключи, состав сущностей и типы полей.

Таблица 8 – Сущности.

Сущность	Первичный ключ	Состав сущности	Тип строки
1	2	3	4
Соискатель	«Id_soiskatel»	Номер соискателя	Integer
		Номер графика	Integer
		Фамилия	Varchar
		Имя	Varchar
		Отчество	Varchar
		Стаж работы	Varchar
		Образование	Varchar
		Зарботная плата	Varchar
Работодатель	«Id_firmi»	Номер фирмы	Integer
		Название фирмы	Varchar
		Юридический адрес фирмы	Varchar
		Телефон	Varchar
Сотрудники	«Id_sotrudnika»	Номер сотрудника	Integer
		Фамилия	Varchar
		Имя	Varchar
		Отчество	Varchar
Тест	«Id_testa»	Номер теста	Integer
		Номер вида	Integer
		Текст теста	Varchar
Вид теста	«Id_vida»	Номер вида	Integer
		Наименование	Varchar

Продолжение таблицы 8

1	2	3	4
Прохождение теста	«Id_prohozh»	Номер прохождения	Integer
		Номер соискателя	Integer
		Номер теста	Integer
Вакансии	«Id_vakansii»	Номер вакансии	Integer
		Название вакансии	Varchar
		Номер графика	Varchar
		Возраст соискателя	Varchar
		Опыт работы	Varchar
		Заработная плата	Varchar
Найм	«Id_rabotodatel»	Номер работодателя	Integer
		Номер сотрудника	Integer
		Количество вакансий	Integer
		Номер фирмы	Integer
		Номер вакансии	Integer
Рекомендация	«Id_recomendacii»	Номер рекомендации	Integer
		Номер сотрудника	Integer
		Номер прохождения	Integer
		Номер соискателя	Integer
		Номер графика	Integer
		Номер вакансии	Integer
		Результат теста	Integer
График работы	«Id_graphik»	Номер графика	Integer
		Вид графика	Varchar
		Время работы	Varchar

В таблице 8 продемонстрированы все сущности необходимые для подсистемы поиска вакансий.

3.2 Создание интерфейса подсистемы поиска вакансий

Для любого приложения важен интерфейс, это является «лицом» любой программы, поэтому для реализации было выбрано программное обеспечение Builder C++ описанное ранее.

На рисунке 3.4 продемонстрированы компоненты DataModule с их помощью происходит соединение с таблицами.

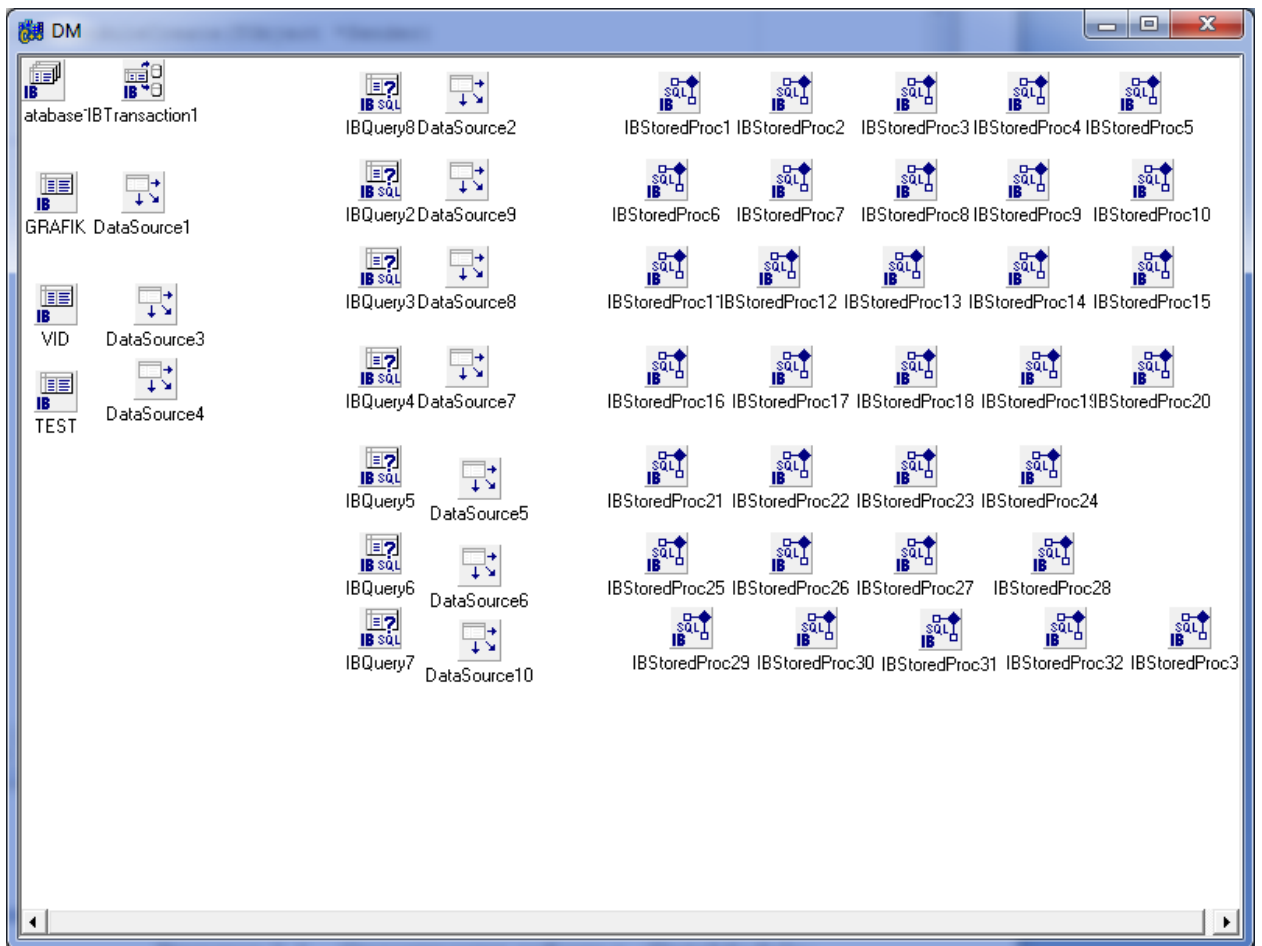


Рисунок 3.4 – Организация таблиц в «DataModule»

IBDatabase – используется для установки с базой данных.

IBTransaction – для управления транзакции.

Создали 11 компонентов «IBDatabase», затем в компоненте поменяли свойство «DatabaseName» на путь сохраненного файла базы данных, а свойству «Name» присвоили название таблицам базы.

В свойстве «TableName» выбрали таблицу соответствующую названию «IBDatabase». Добавили компоненты «DataSource» в свойстве «DataSet» указали названия таблиц. Добавили компоненты запросов для получения данных из таблиц и связали с компонентом «DataSource».

Перейдём к созданию меню программы.

На форму помещаем компонент MainMenu, двойным нажатием на компонент выделили пустое поле и поменяли свойство «Caption» на удобное название ссылок на формы, сгруппировали для удобного использования. Для

корректной работы ссылок необходимо подключить их к формам, сделали это с помощью функции «Include Unit Hdr» как на форме базы, так и на форме главного меню.

На рисунках продемонстрирован интерфейс, созданный с помощью «MainMenu». Ссылки на таблицы сделаны неактивными, пока пользователь не найдёт своё «ФИО» и не введёт пароль для входа в систему.

На главной форме реализован вход для сотрудников, для этого были использованы объекты «Panel», «ComboBox», «Label», «Edit», «CheckBox».

На рисунке 3.5 продемонстрирован интерфейс главного меню подсистемы.

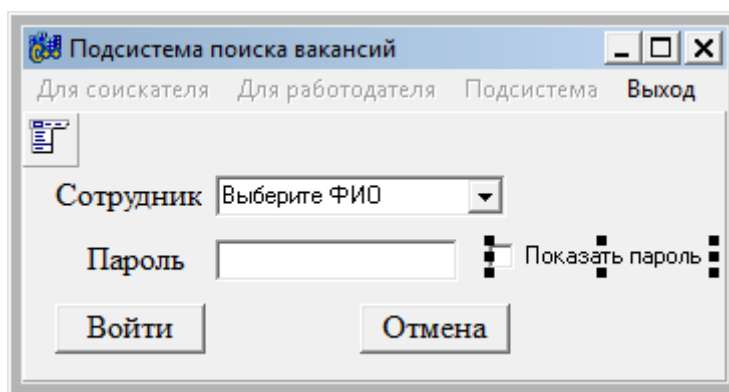


Рисунок 3.5 – Интерфейс главного меню

На рисунке 3.6 продемонстрирована вкладка для соискателя

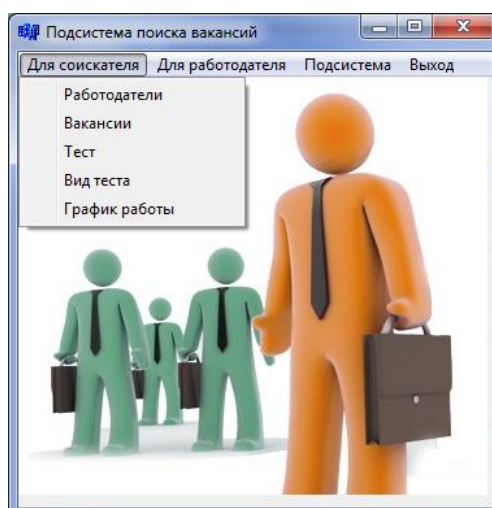


Рисунок 3.6 – Вкладка «Для соискателя»

Вкладка «Для соискателя» имеет 5 ссылок на формы, подключенные к базе данных.

На рисунке 3.7 продемонстрирована вкладка «Для работодателя».

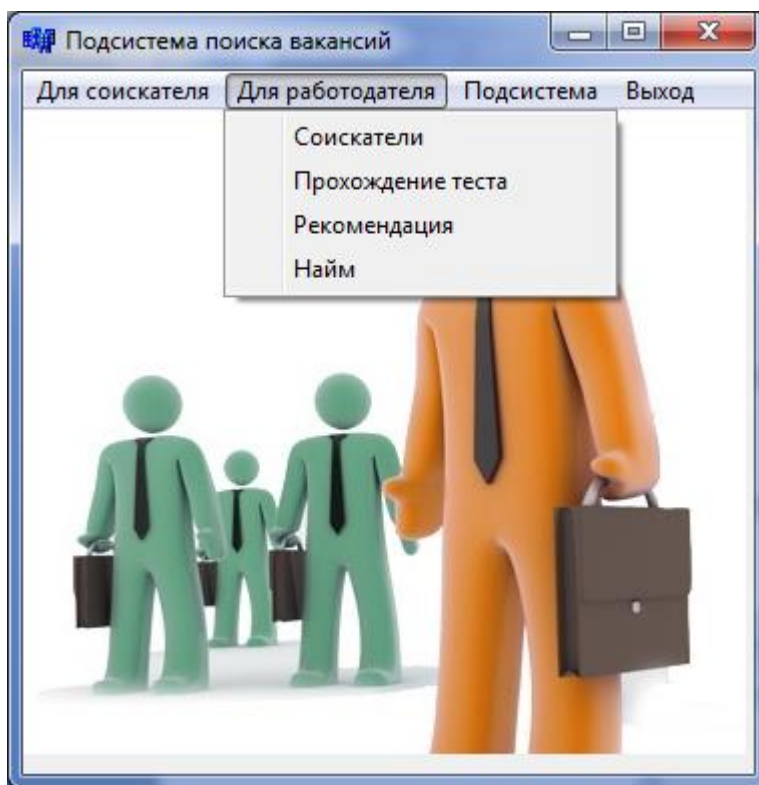


Рисунок 3.7 – Вкладка «Для работодателя»

Вкладка «Для работодателя» имеет 4 ссылки, на формы, подключенные к базе.

Вкладка «Подсистема» имеет ссылки на формы, подключенные к базе.

На рисунке 3.8 продемонстрирована вкладка «Подсистема».

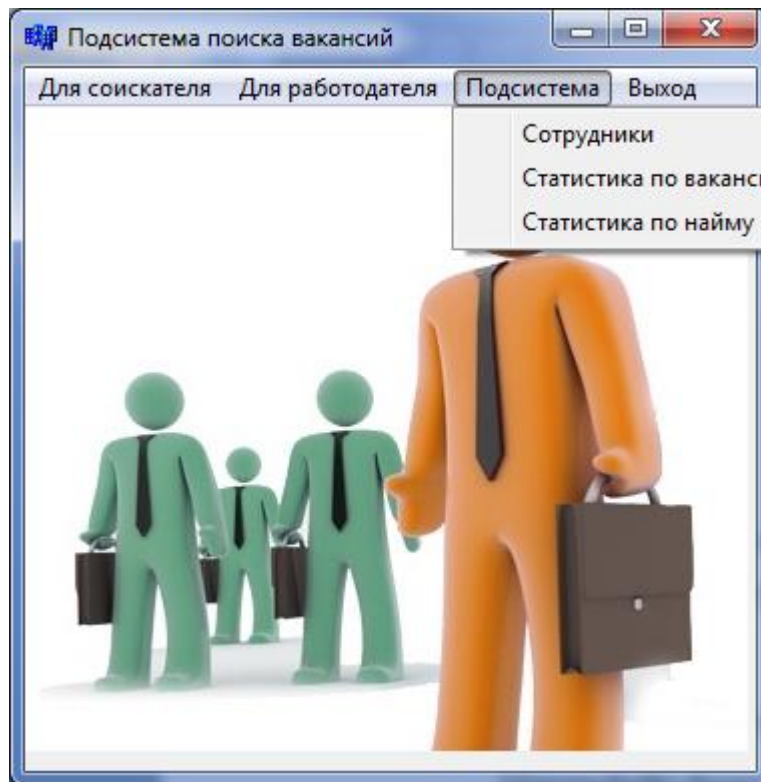


Рисунок 3.8– Вкладка «Подсистема»

В главном меню программы реализован выход из приложения.

Интерфейс формы работодателя представлен на рисунке 3.9.

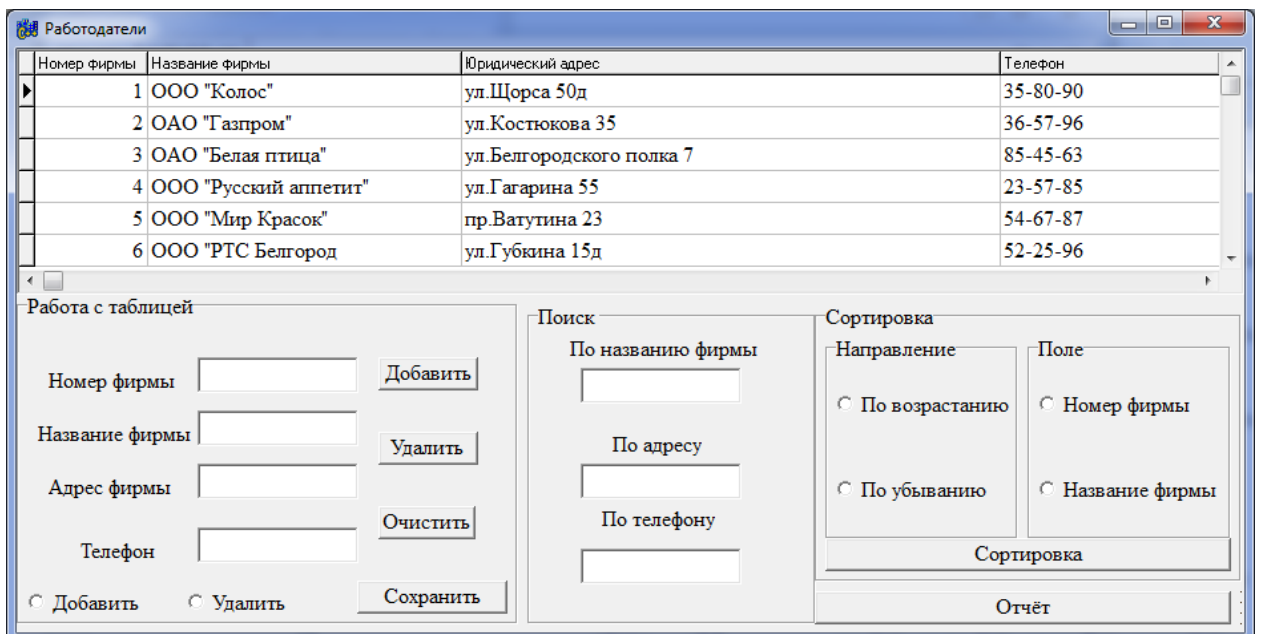


Рисунок 3.9 – Интерфейс формы «Работодатели»

Форма «Работодатели» отражает данные из одноименной таблицы.

Создали новую форму, подключили к главному меню, в свойстве «Caption» присвоили название «Работодатели». Добавили компонент «DBGrid» он необходим для вывода данных из базы, соединили его с «DataSource» в свойстве «Align» выбрали значение «AllTop» и выровняли компонент.

Добавим на форму 2 компонента «GroupBox» один нужен для поиска в таблице другой для работы с таблицей.

«GroupBox» - компонент представляет собой контейнер объектов, служит для группировки функций.

В «GroupBox» поместили 4 объекта «Button» в свойстве «Caption» присвоили название функций работы с таблицей: добавить, удалить, сохранить и очистить, после добавили 4 компонента «Label» и «Edit» они помогут при работе с таблицей. В «Edit» вводится информация для добавления или удаления.

Создали еще один компонент «GroupBox» поместили объект «Button» и «RadioGroup» в свойстве «Item» указали строки для сортировки.

Аналогичным способом организовали интерфейс к остальным формам рисунки 3.10- 3.18.

На рисунке 3.10 показан интерфейс формы «График работы» отображает данные о виде графика и времени работы (см. рисунок 3.10).

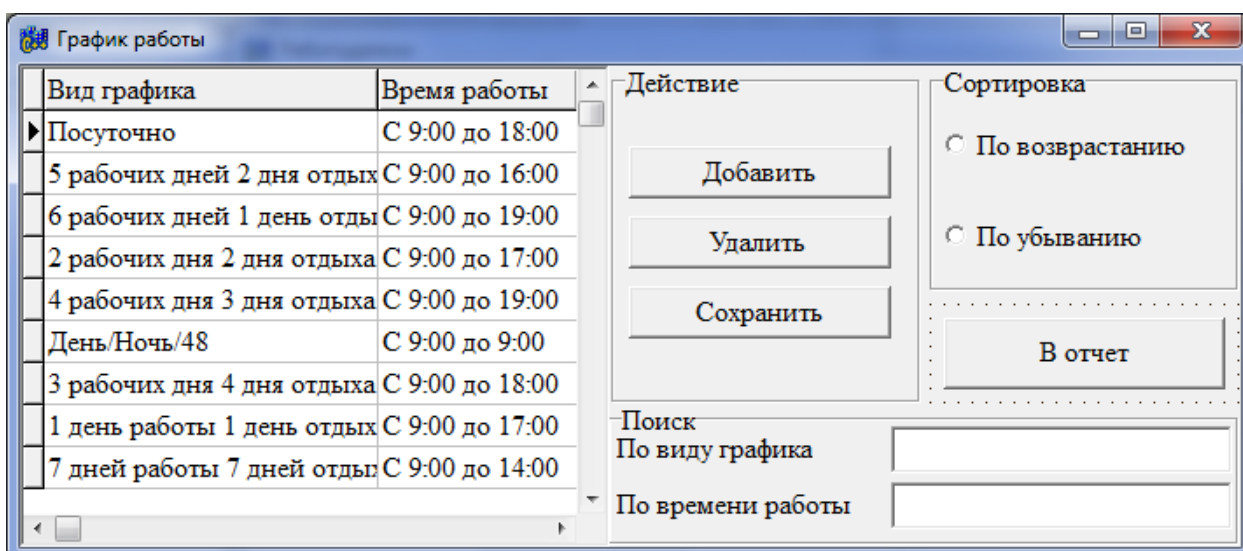


Рисунок 3.10 – Интерфейс формы «График работы»

На рисунке 3.11 показан интерфейс формы «Найм».

На форме продемонстрированы данные о номера работодателя, номере сотрудника центра, количестве вакансий требуемых для работодателя, номер фирмы работодателя и номер нужной вакансии.

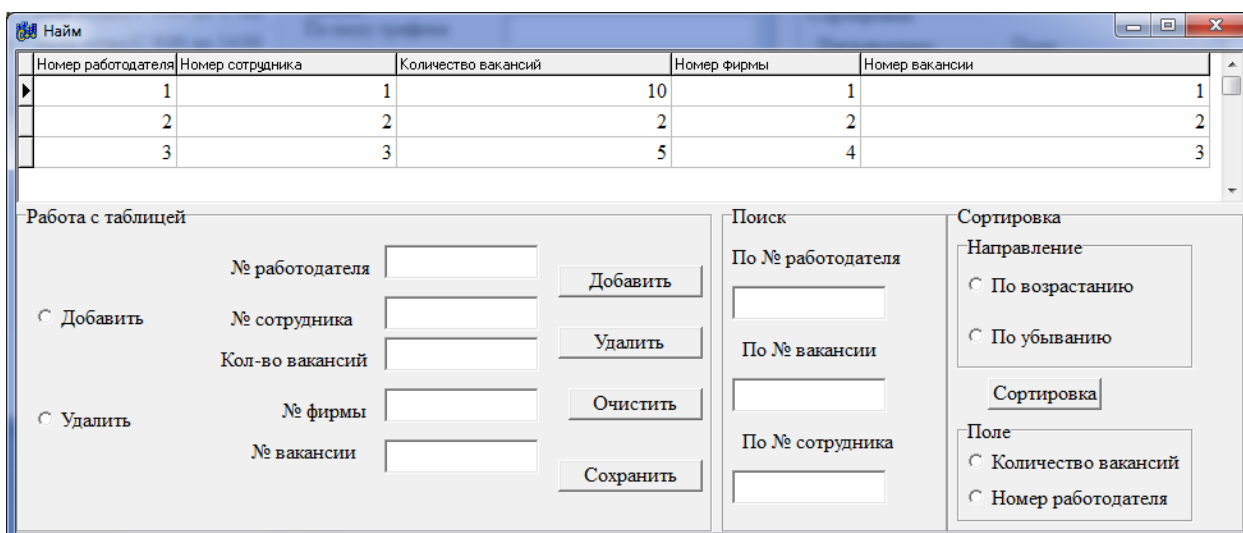


Рисунок 3.11 – Интерфейс формы «Найм»

На рисунке 3.12 показан интерфейс формы «Прохождение теста».

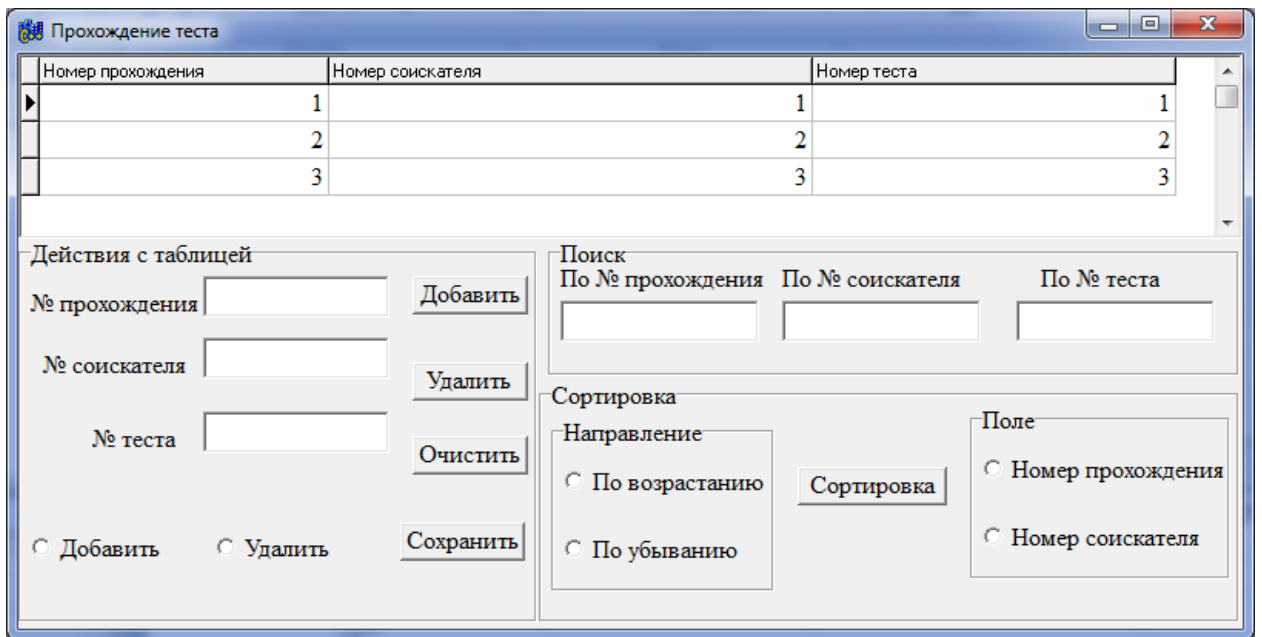


Рисунок 3.12 – Интерфейс формы «Прохождение теста»

На рисунке 3.13 показан интерфейс формы «Вид теста».

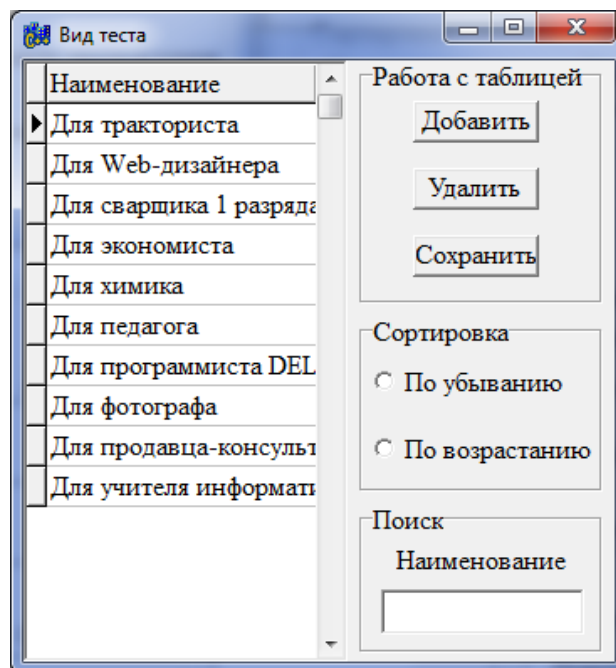


Рисунок 3.13 – Интерфейс формы «Вид теста»

На рисунке 3.14 показан интерфейс формы «Рекомендация».

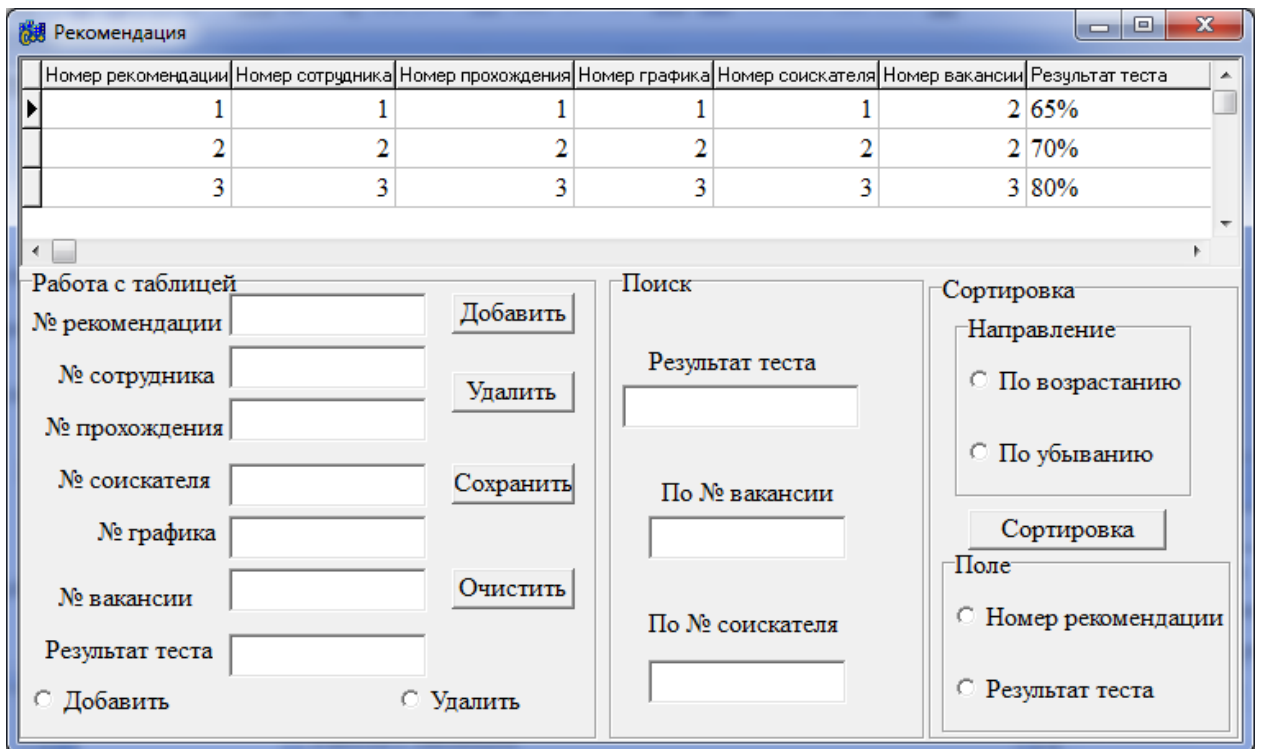


Рисунок 3.14 – Интерфейс формы «Рекомендация»

На рисунке 3.15 показан интерфейс формы «Соискатели».

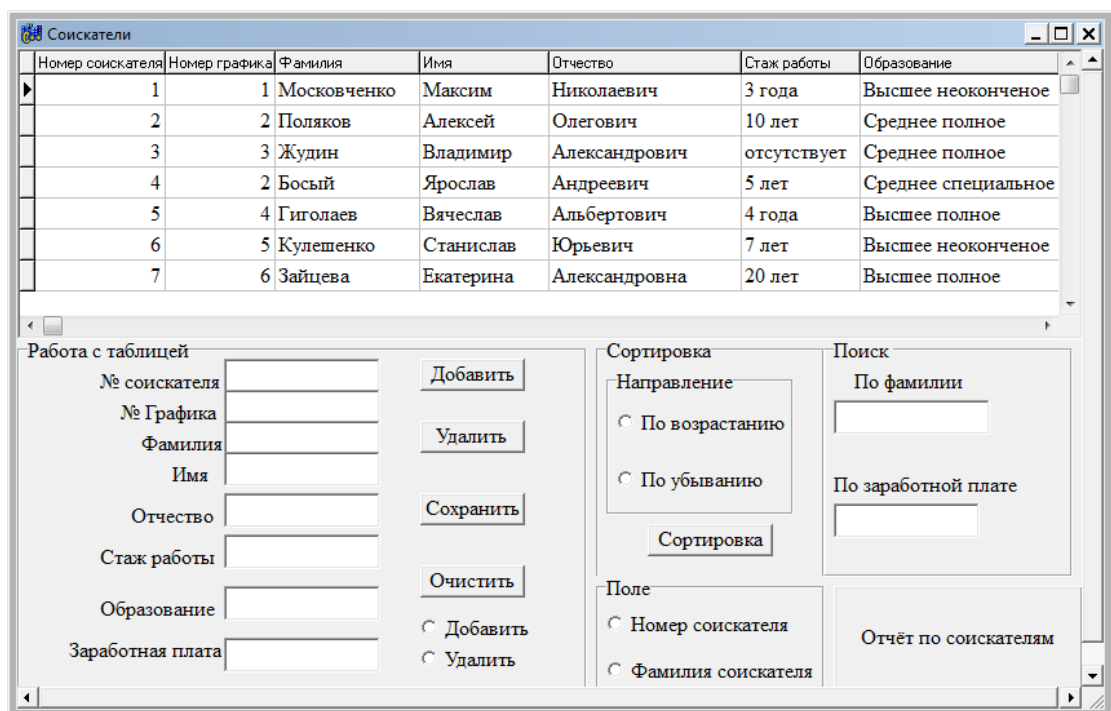


Рисунок 3.15 – Интерфейс формы «Соискатели»

На рисунке 3.16 показан интерфейс формы «Сотрудники».

На форме «Сотрудники» представляются данные о сотрудниках, а именно
номер сотрудника, фамилия, имя отчество, должность.

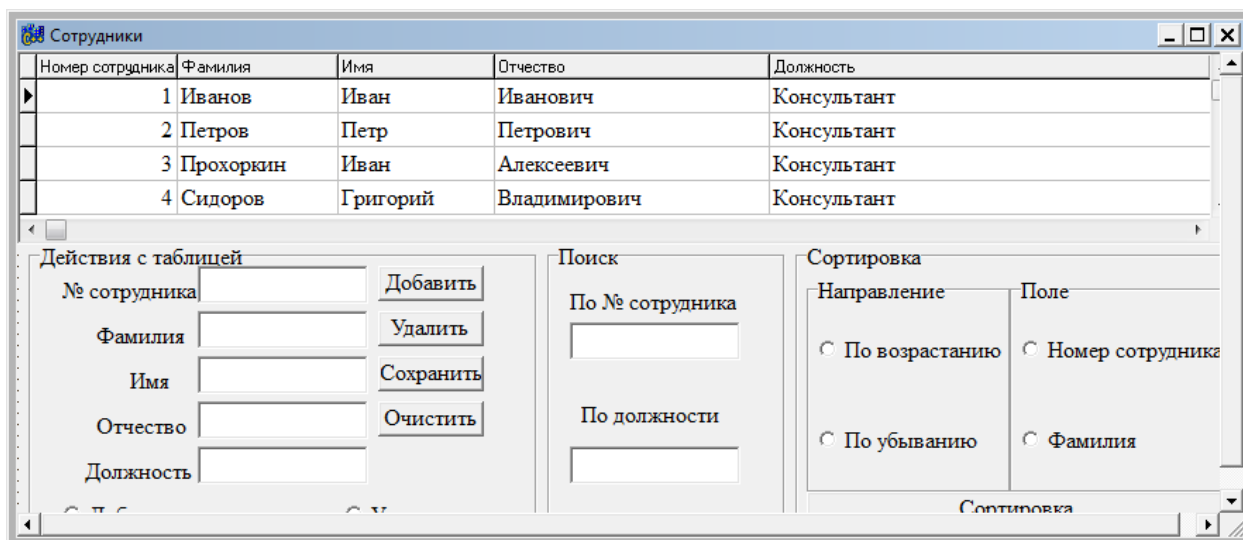


Рисунок 3.16 – Интерфейс формы «Сотрудники»

На рисунке 3.18 показан интерфейс формы «Тест». В форме «Тест» демонстрируется информация о номере теста и текст теста.

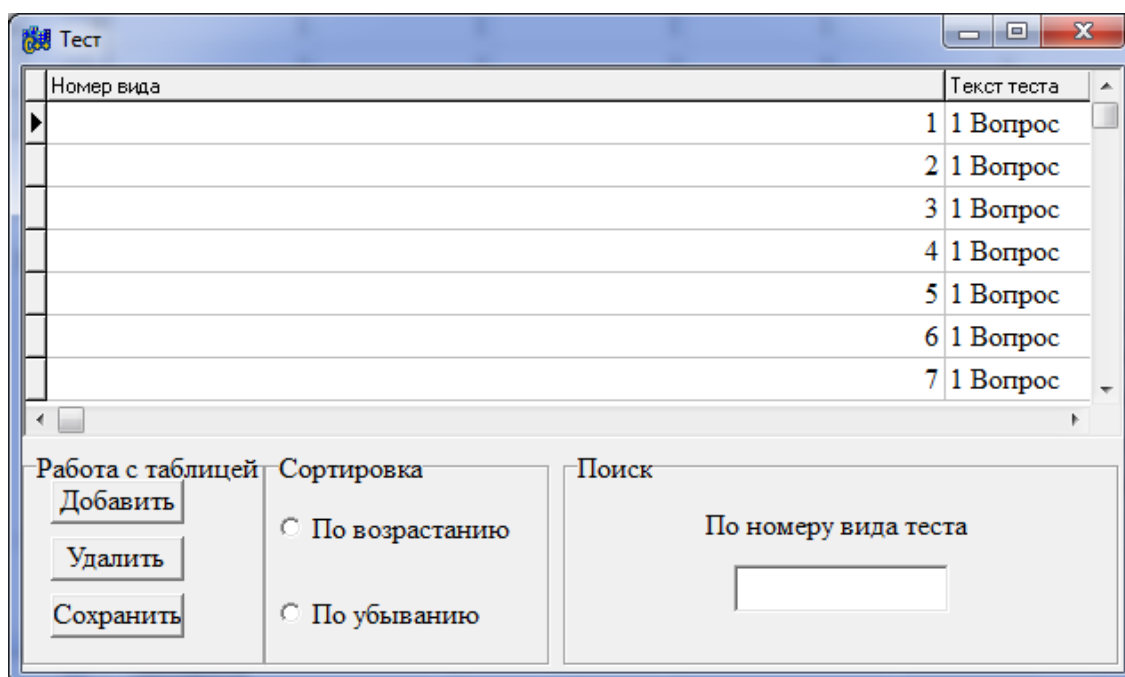


Рисунок 3.17 – Интерфейс формы «Тест»

На рисунке 3.18 показан интерфейс формы «Вакансии»

Форма «Вакансии» содержит информацию о названии вакансии, графике работы вакансии, возрасте соискателя требуемого по данной вакансии и размер заработной платы.

Номер вакансии	Название вакансии	Зарплата	Номер графика	Возраст соискателя	Опыт работы
4	Экономист	300000 рублей	4	35 лет	15 лет
1	Web-дизайнер	20000 рублей	2	30 лет	5 лет
2	Программист C++	25000 рублей	5	25 лет	3 года
3	Тракторист	50000 рублей	5	35 лет	10 лет

Работа с таблицей

Номер вакансии

Название вакансии

Зарплата

Номер графика

Возраст соискателя

Опыт работы

Добавить Удалить

Поиск

По вакансии

По заработной плате

По возрасту соискателя

Сортировка

Направление

По возрастанию

По убыванию

Поле

Номер вакансии

Название вакансии

Рисунок 3.18 – Интерфейс формы «Вакансии»

3.3 Тестирование программы

Для тестирования запустим программу. Выбрали из списка сотрудников фамилию, имя и отчество и ввели пароль, для каждого сотрудника он уникален, также реализована функция «Показать пароль» используется для проверки введенного пароля.

На рисунке 3.19 – меню запущенной программы.

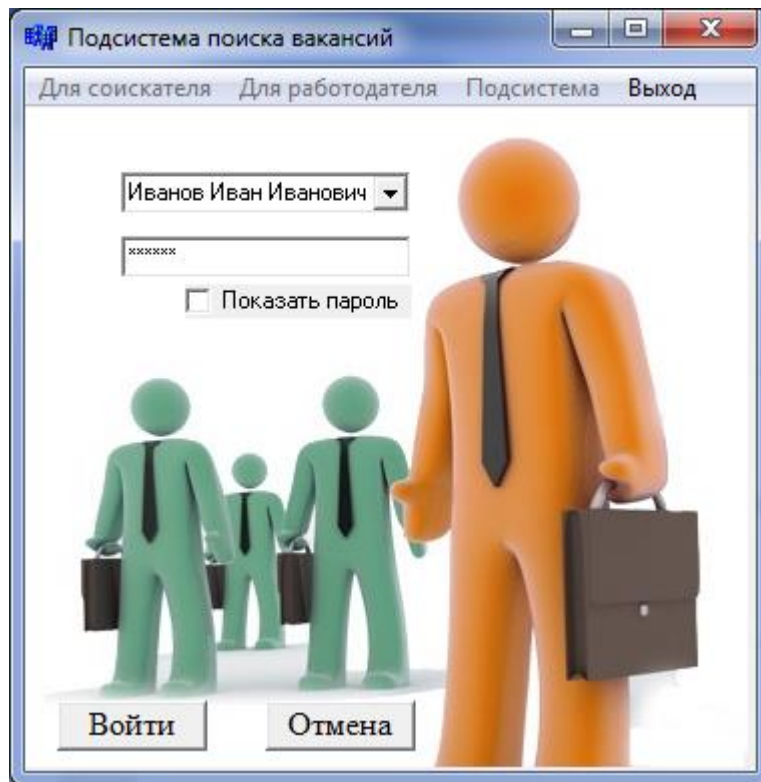


Рисунок 3.19 – меню запущенной программы

На рисунке 3.20 – сообщение при правильном вводе ФИО и пароле.

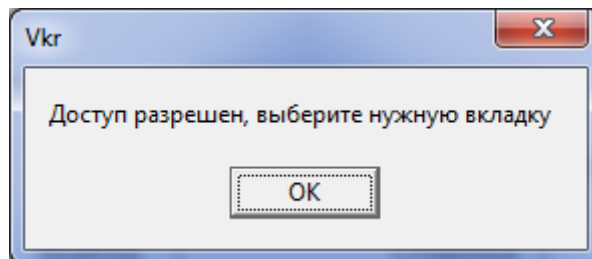


Рисунок 3.20 – сообщение при правильном вводе ФИО и пароле

Далее выбираем интересующую нас вкладку, к примеру, вкладку «Для работодателя» таблица «Соискатели».

На рисунке 3.21 – показана вкладка «Для работодателя» со ссылками на таблицы базы данных.

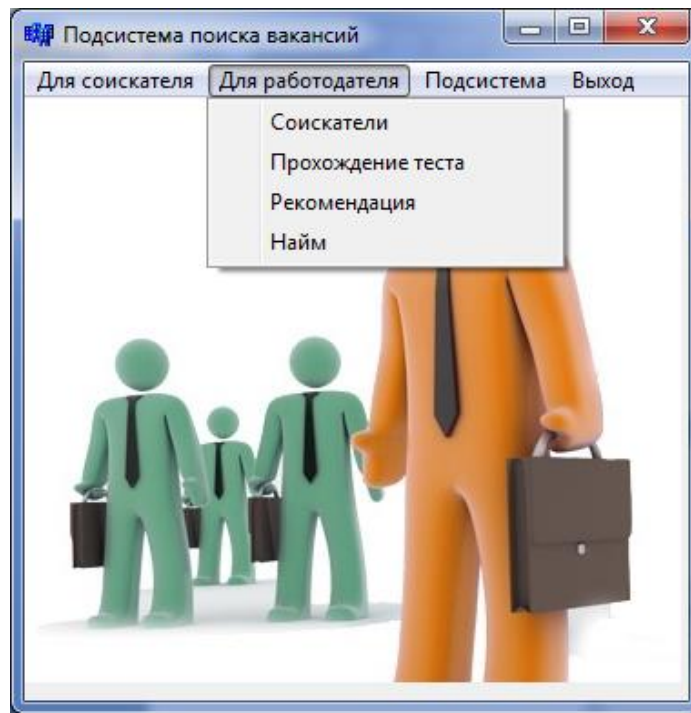


Рисунок 3.21 – Вкладка «Для работодателя» со ссылками на таблицы базы данных

Нажмём кнопку «Добавить» и введем данные для добавления.

На рисунке 3.22 – показана функция добавление в таблицу «Соискатели».

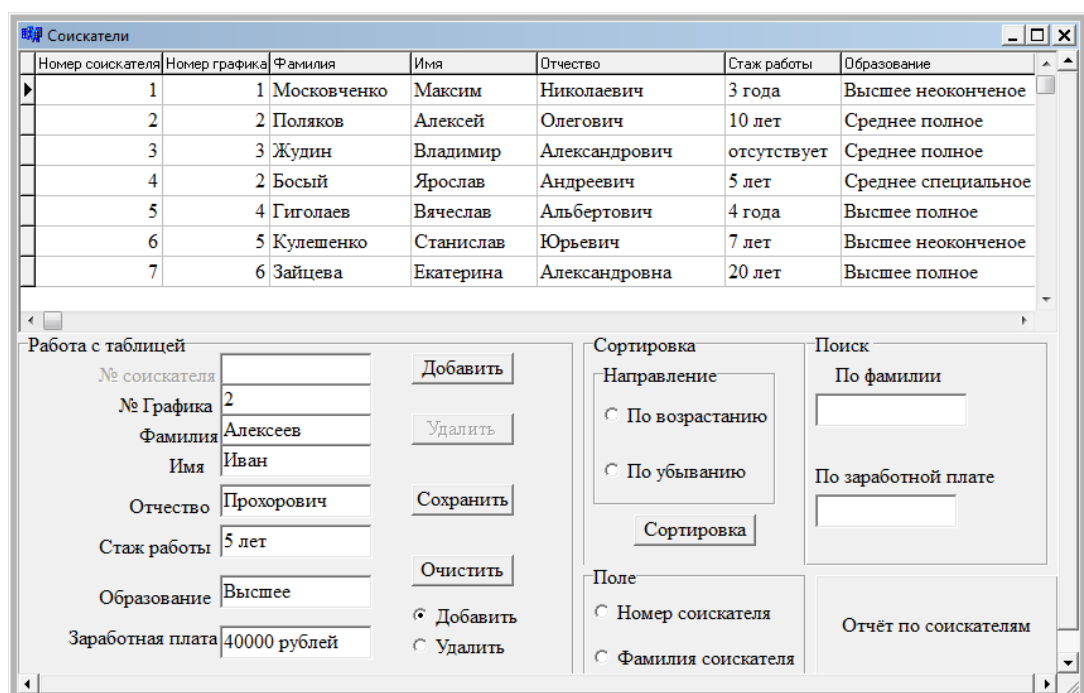


Рисунок 3.22 – Добавление в таблицу «Соискатели»

На рисунке 3.23 таблица «Соискатели» с добавленной записью.

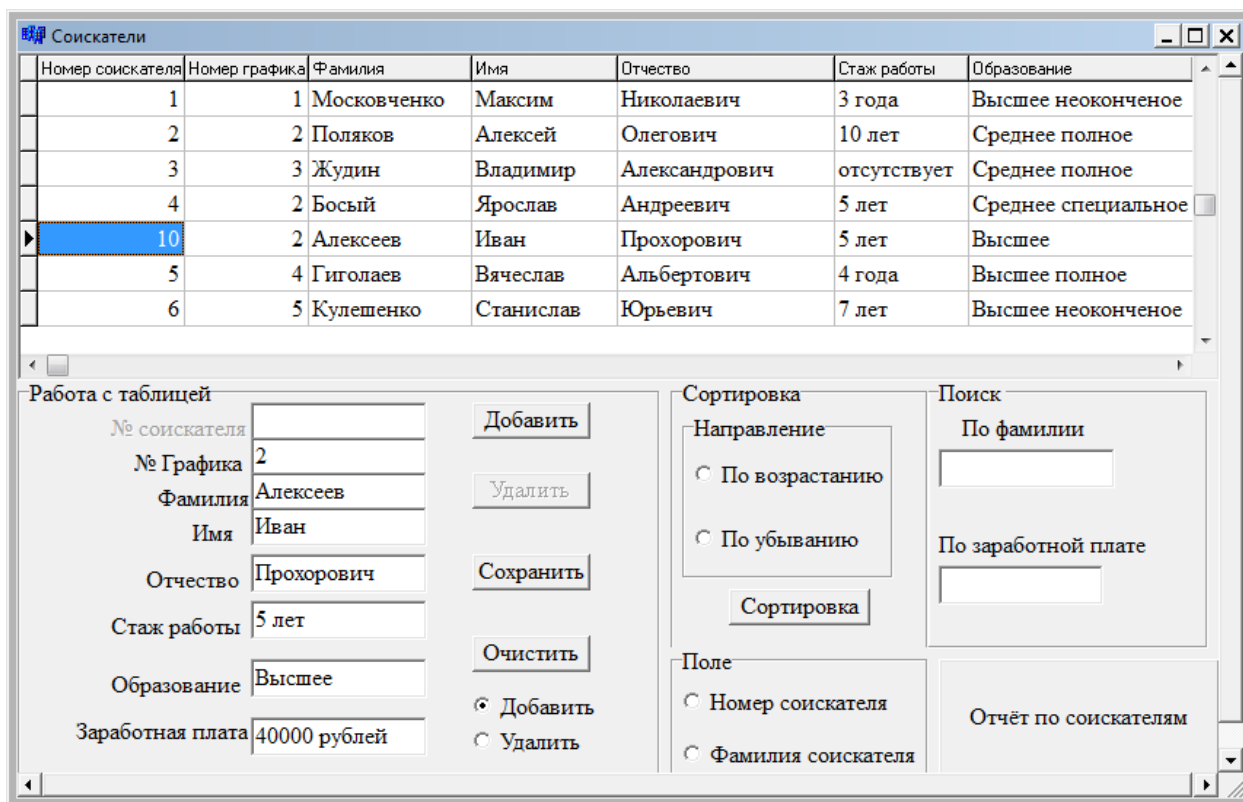


Рисунок 3.23 – Добавление в таблицу «Соискатели»

На рисунке 3.24 показан поиск по фамилии в таблице «Соискатели».

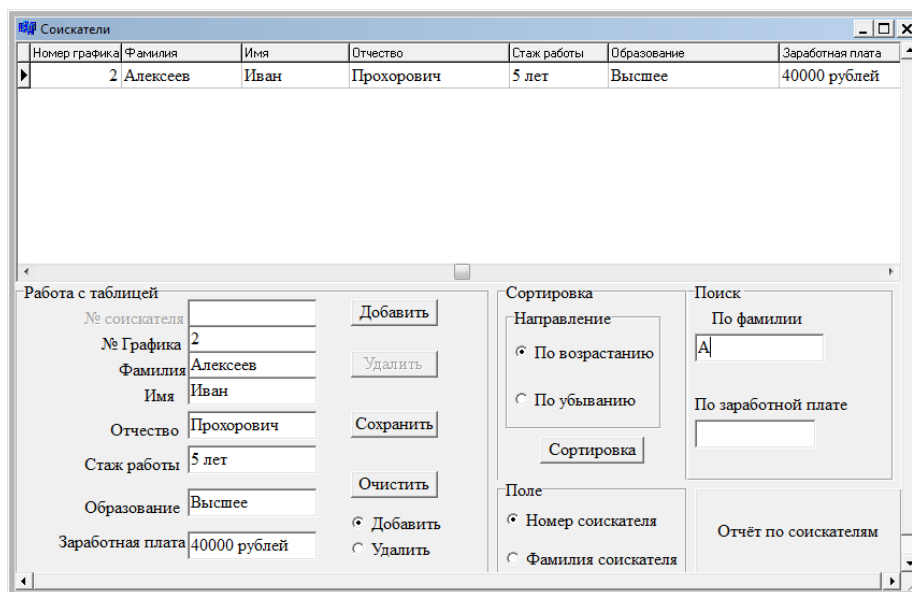


Рисунок 3.24 – Поиск по фамилии в таблице «Соискатели»

На рисунке 3.25 показан поиск по заработной плате в таблице «Соискатели».

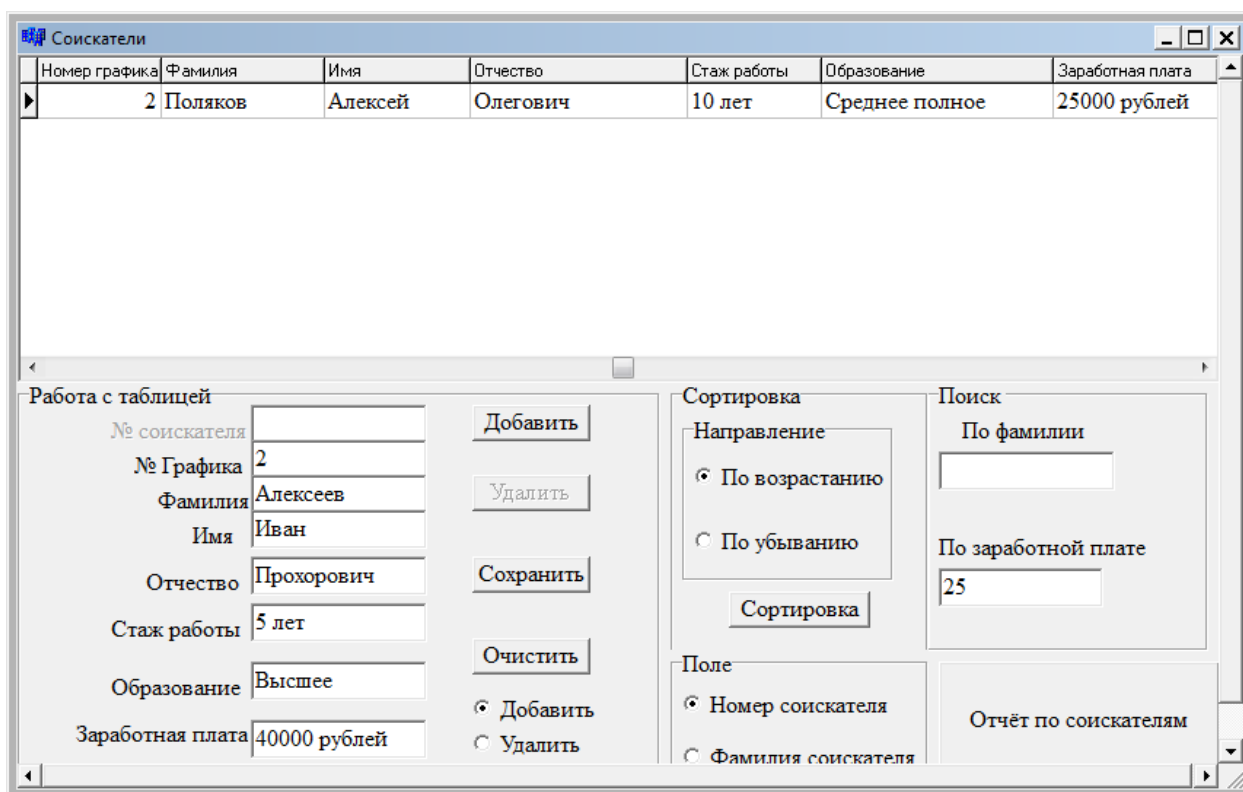


Рисунок 3.25 – Поиск по заработной плате в таблице «Соискатели»

На рисунке 3.26 продемонстрирована сортировка по возрастанию в таблице «Соискатели».

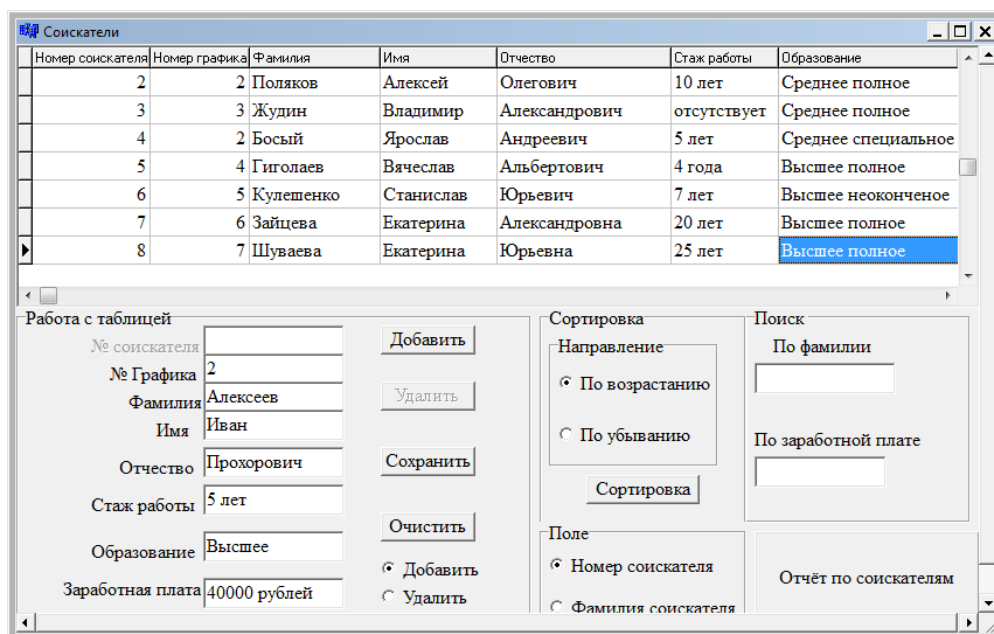


Рисунок 3.26 – Сортировка номера соискателя по возрастанию в таблице «Соискатели»

На рисунке 3.27 продемонстрирована сортировка по убыванию в таблице «Соискатели».

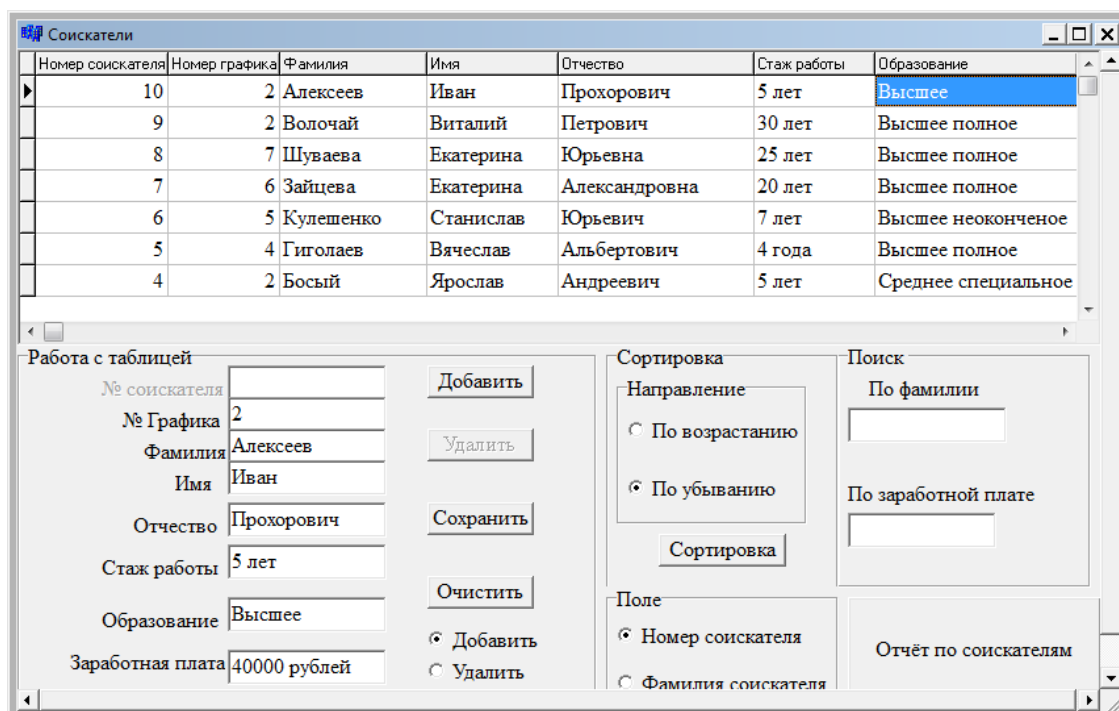


Рисунок 3.27 – Сортировка номера соискателя по убыванию в таблице «Соискатели»

На рисунке 3.28 – продемонстрирована сортировка фамилии по возрастанию в таблице «Соискатели».

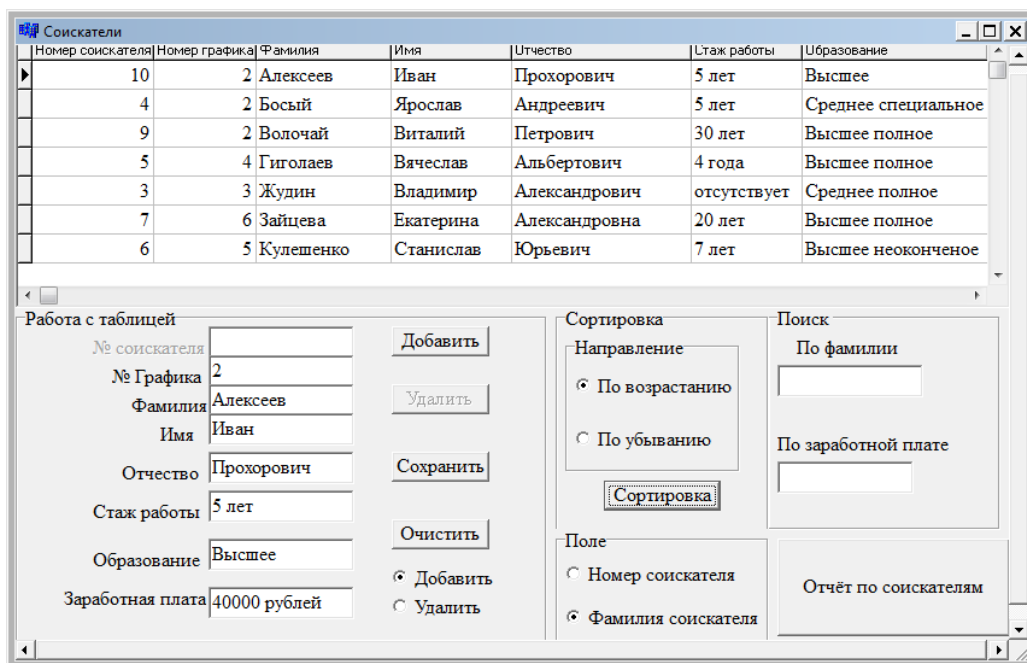


Рисунок 3.28 – Сортировка отчества по возрастанию в таблице «Соискатели»

На рисунке 3.29 показана сортировка отчества по убыванию в таблице «Соискатели».

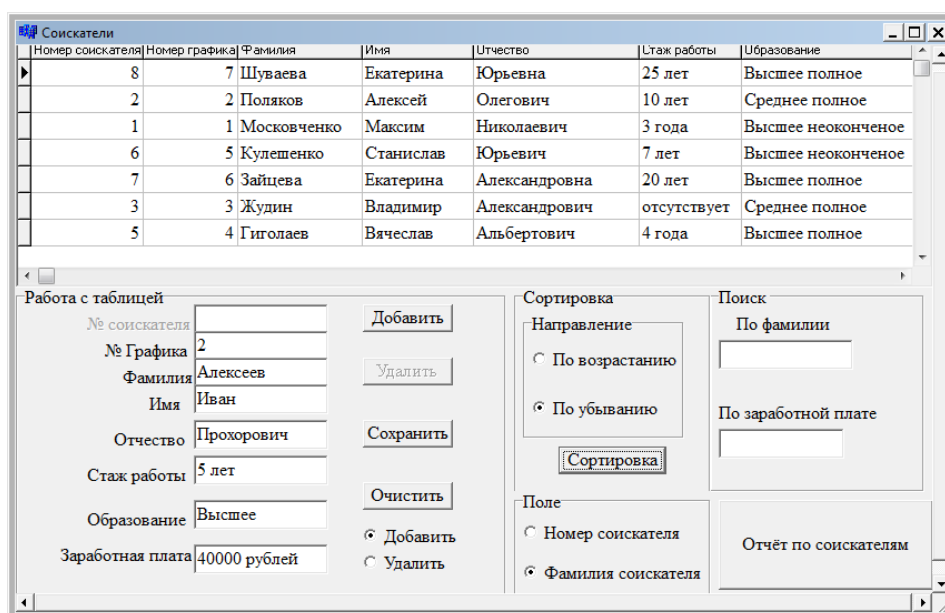
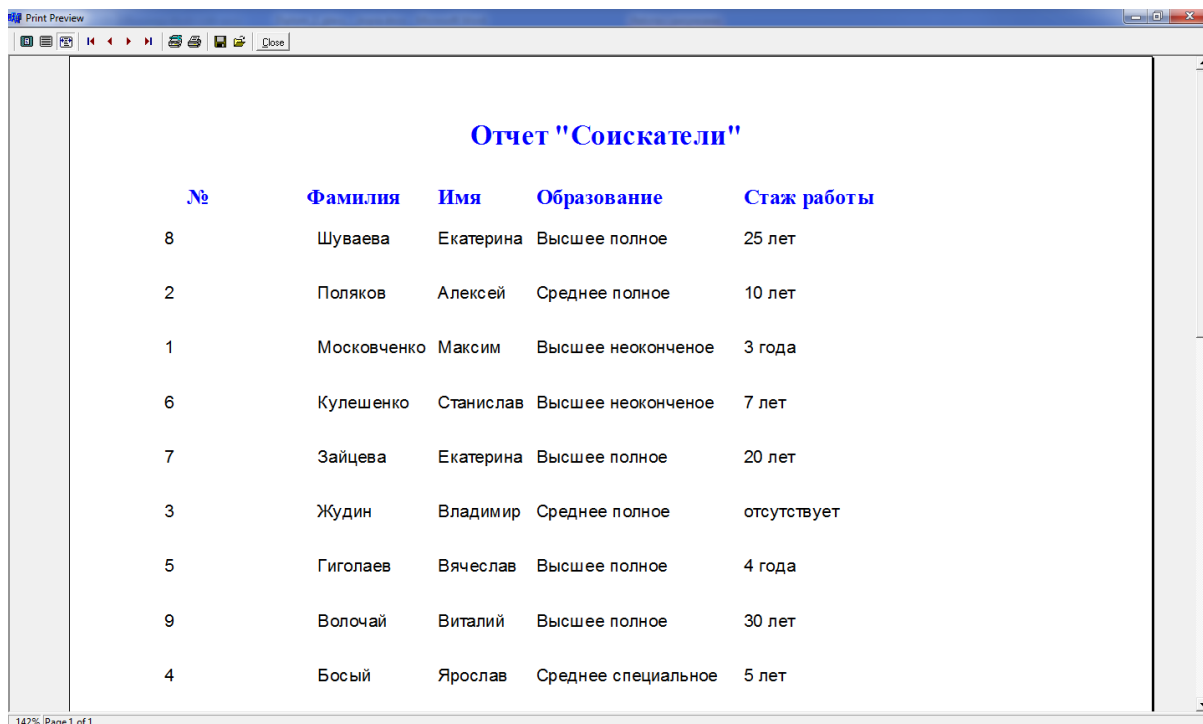


Рисунок 3.29 – Сортировка по отчеству в таблице «Соискатели»

На рисунке 3.30 показан отчёт по таблице «Соискатели»



№	Фамилия	Имя	Образование	Стаж работы
8	Шуваева	Екатерина	Высшее полное	25 лет
2	Поляков	Алексей	Среднее полное	10 лет
1	Московченко	Максим	Высшее неоконченное	3 года
6	Кулешенко	Станислав	Высшее неоконченное	7 лет
7	Зайцева	Екатерина	Высшее полное	20 лет
3	Жудин	Владимир	Среднее полное	отсутствует
5	Гиголаев	Вячеслав	Высшее полное	4 года
9	Волочай	Виталий	Высшее полное	30 лет
4	Босый	Ярослав	Среднее специальное	5 лет

Рисунок 3.30 – Отчёт по таблице «Соискатели»

На рисунке 3.31 показана статистика по вакансиям

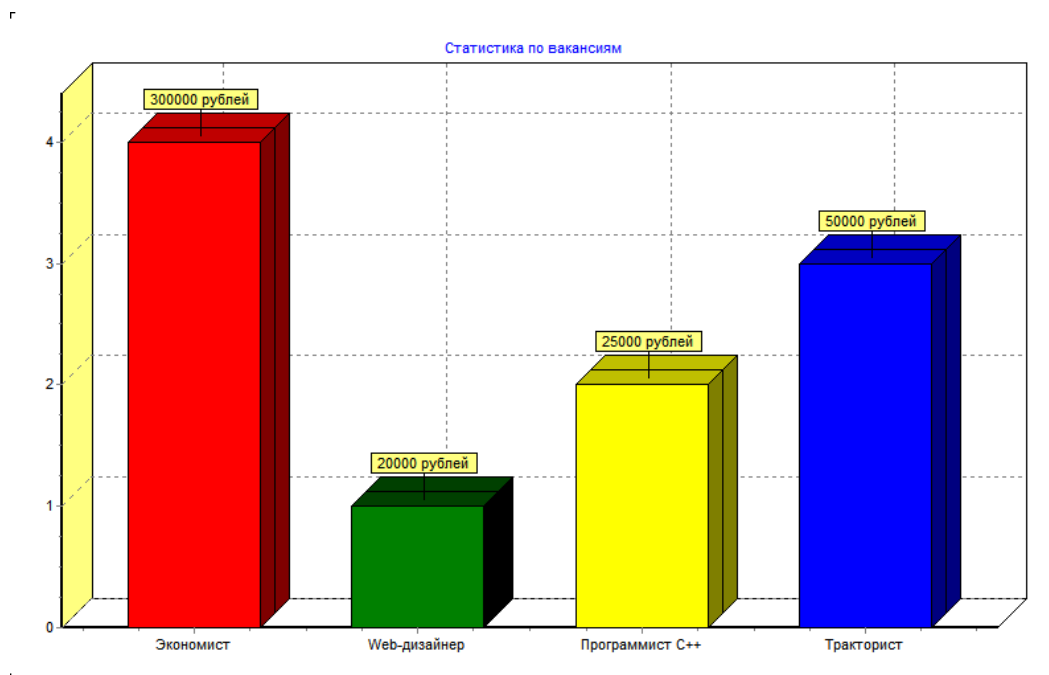


Рисунок 3.31 – Статистика по вакансиям

3.4 Оценка экономической эффективности внедрения подсистемы

Стоимость создания автоматизированной подсистемы определялась по следующим статьям калькуляции:

- основная заработная плата производственного персонала;
- отчисления на социальные нужды;
- затраты на электроэнергию;
- затраты на амортизацию и ремонт вычислительной техники;
- накладные расходы .

Оценка экономической эффективности проекта является важнейшей частью, при принятии решений о целесообразности вложения в него средств.

Показатель эффективности определяет положительный результат, который достигается при использовании программного продукта. Чистая прибыль от использования продукта за год определяется по формуле 3.1:

$$П = P_2 - Z_2, \quad (3.1)$$

где P_2 – стоимостная оценка результатов применения программного продукта в течение года;

Z_2 – стоимостная оценка затрат при использовании программного продукта.

Трудовые затраты на разработку составили 120 часов или 15 рабочих дней при восьмичасовом рабочем дне. Инженер в среднем за месяц тратит 160 часов, среднемесячная заработная плата 25000 рублей.

Расчет основной заработной платы ($Z_{осн}$) производилось по формуле 3.2.

$$Z_{осн} = \frac{Z_{ср}}{\Phi_{ср}} * Ч, \quad (3.2)$$

где $Z_{ср}$ – среднемесячная заработная плата;

$\Phi_{ср}$ – среднемесячный фонд рабочего времени;

$Ч$ – это количество отработанных часов.

В соответствии с формулой 3.2 основная заработная плата инженера составила 15000 рублей.

В соответствии с Федеральным законом от 24 июля 2009 года N 212-ФЗ "О страховых взносах в Пенсионный фонд РФ, Фонд социального страхования РФ, Федеральный фонд обязательного медицинского страхования и территориальные фонды обязательного медицинского страхования" (в редакции Федерального закона от 03.12.2011 № 379-ФЗ) страховой взнос составляет 30% от дохода, который вычисляется по формуле 3.3:

$$СВ = Z_{осн} * P_{св}, \quad (3.3)$$

где $Z_{осн}$ – основная заработная плата;

$P_{св}$ – размер страхового взноса на социальные нужды.

Итоговые отчисления на социальные нужды составили 7,500 рублей, а основная заработная плата разработчика с учетом отчислений – 17,500 руб.

Прежде чем рассчитать затраты на обработку информации, представим расчет экономии за счет увеличения производительности труда пользователя.

Расчет показателя повышения производительности труда произведен по формуле 3.4:

$$P = \left(\frac{\Delta T}{F - \Delta T} \right) * 100, \quad (3.4)$$

где F – время, которое планировалось пользователем для выполнения работы до внедрения программ;

ΔT – экономия времени после внедрения подсистемы.

Таблица 9 – Оценка времени работы пользователей

№ п/п	Вид работ	Среднее время на операцию в месяц на одного сотрудника, минут		Экономия времени в месяц, минут ΔT	Повышение производительности труда, % Р
		До автоматизации	После автоматизации		
1.	Сбор информации	660	330	330	50
2.	Анализ данных	440	220	220	50
3.	Ввод информации	880	440	440	50
4.	Поиск данных	440	220	220	50
5.	Проведение тестирования	1320	720	600	30
6.	Формирование отчетов	330	165	165	50
ИТОГО		4070	2095	1975	

При расчете сделаны следующие допущения:

- на шесть типов операций приведенных в таблице 9 , каждый пользователь тратит 60% рабочего времени;
- фонд рабочего времени в месяц составляет 5 280 минут;
- все сотрудники проводят одинаковое время при работе с операциями.

Рассчитали экономию времени работы пользователя:

$$P = \frac{2400}{3150} = 51,4\%$$

Стоимостная оценка затрат при использовании программного продукта рассчитали по формуле 3.5.

$$P_2 = (Z_{руч} - Z_{авт}), \quad (3.5)$$

где $Z_{руч}$ – затраты на ручную обработку информации, руб/год;

$Z_{авт}$ – затраты на автоматизированную обработку информации, руб/год.

Затраты на ручную обработку информации вычислялись по формуле 3.6:

$$Z_{руч} = V_p * C_{ч}, \quad (3.6)$$

где V_p – время, затрачиваемое на обработку информации вручную, ч/год;

$C_{ч}$ – цена 1 ч работы, руб/год.

Годовые затраты на 15 рабочих мест за год (12 месяцев) при ручной обработке информации составляют 47,3 ч, в месяц:

$$Z_{руч} = 47,3 * 12 * 73 * 8 = 331478,4 \text{ руб.}$$

Затраты на автоматизированную обработку информации вычислялись по формуле 3.7:

$$Z_{авт} = V_a * C_{ч}, \quad (3.7)$$

где V_a – затраты времени на автоматизированную обработку информации, руб/год.

При автоматизации затраты времени – 10,15 ч в месяц:

$$Z_{авт} = 37,83 * 12 * 84,08 * 8 = 71131,2 \text{ руб.}$$

Годовой результат от внедрения программного продукта рассчитали по формуле 3.5:

$$P_r = Z_{руч} - Z_{авт} = 331478,4 - 71131,2 = 260347,28 \text{ руб.}$$

Затраты на электроэнергию вычислялись по формуле 3.8:

$$C_{э\text{э}} = H_{ч} * Ч * T_{ч} \quad (3.8)$$

где H_q – норма потребления электроэнергии за час;

$Ч$ – количество рабочих часов электроприборов;

T_q – тарифный план за использование 1 КВт/ч.

Затраты на использование электроэнергии стационарным компьютером и монитором с мощностями 0,07 КВт/ч и 0,04 КВт/ч, тарифным планом – 3,53 рублей за 1 КВт за 170 часов работы составили 66,01 рублей.

Расчет амортизационных отчислений по компьютерной технике вычислялся исходя из нормы амортизации, установленной в зависимости от нормативного срока использования компьютерной техники. Для стационарного компьютера предполагаемый срок использования 5 лет, а процент амортизационных отчислений в год составил 20%. Первоначальная стоимость компьютера устанавливалась исходя из средней стоимости с учетом расходов на доставку, установку и монтаж (35200 рублей), а годовой фонд рабочего времени в часах, исходя из нормы рабочего времени на 2016 год (1974 час).

Расчет амортизационных отчислений вычислялся по формуле 3.9:

$$AO = \frac{C_n * П_{AO}}{\Phi_2} * \Phi_{cp}, \quad (3.9)$$

где C_n – первоначальная стоимость компьютера;

$П_{AO}$ – процент амортизационных отчислений в год;

Φ_2 – годовой фонд рабочего времени в часах за 2016 год;

Φ_{cp} – среднемесячный фонд рабочего времени.

Таким образом, амортизационные отчисления составили 739,28 рублей.

На основе произведенных расчетов по отдельным статьям калькуляции была составлена таблица 10.

Таблица 10 – Калькуляция себестоимости

№	Наименование статей затрат	Сумма (руб.)
1	Затраты на оплату труда работников	25000,00
2	Отчисления на социальные нужды	7500,00
3	Накладные расходы	35,200

Продолжение таблицы 10

4	Амортизационные отчисления	739,28
5	Электроэнергия	66,01
	Итого:	68505,29

В третьем разделе была описана программная реализация автоматизированной подсистемы поиска вакансий. Было описано создание базы данных для подсистемы. Описан процесс создания интерфейса для программного обеспечения. Проведено тестирование приложения. Дана оценка экономической эффективности от внедрения программного продукта в ОКУ «Белгородский центр занятости населения».

ЗАКЛЮЧЕНИЕ

В процессе написания выпускной квалификационной работы были достигнуты все поставленные цели и задачи. Цель исследования, является разработка и обоснование автоматизированной подсистемы поиска вакансий

Задачами в данной выпускной квалификационной работе были:

- анализ структуры и организации деятельности Белгородского центра занятости населения;
- выявление проблем информационной поддержки трудоустройства населения;
- оценка эффективности разработанной автоматизированной подсистемы поиска вакансий;

Было изучено и проанализировано использование информационных систем в центре занятости населения, продемонстрирована общая характеристика структуры и деятельности центра занятости населения, произведено исследование существующих автоматизированных систем обслуживания клиентов в центрах занятости, описано обоснование необходимости автоматизации и цели создания подсистемы поиска вакансий, также дано обоснование программных решений, описание технического и кадрового обеспечения. Разработано программное обеспечение для улучшения работы центра занятости с точки зрения затрат времени в работе с клиентами. Программа имеет удобный интерфейс и систему для входа пользователей. Была дана оценка экономической эффективности внедрения подсистемы поиска вакансий в центре занятости населения. С помощью case-инструментов описали работу центра занятости в сфере поиска вакансий.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- 1 Арсеньев, Б.П. Интеграция распределенных баз данных [Текст]/ Б.П. Арсеньев, С.А. Яковлев. . - СПб. Изд. «Лань», 2010 – 462с .
- 2 Архангельский, А. Я. С++Builder 6 Справочное пособие. Книга 2. Классы и компоненты. [Текст]/ А.Я. Архангельский.. - М.: Бином-Пресс, 2010. – 528 с.
- 3 Балдин, К.В., Уткин В.Б. Информационные системы в экономике. [Текст]/ К.В. Балдин, Уткин В.Б, Учебное пособие. – Дашков и К, 2011. – 395 с.
- 4 Беспалов, Р.С. Инструментарий разработчика бизнес-процессов. - М.: Акцион-Медиа, 2013. – 400 с.
- 5 Болтенков, В.И. Конфигурирование и настройка автоматизированных информационных систем: Учеб. пособие. В.И. Болтенков А.Л.Литвинов, Н.В. Лычёва. - Белгород: Издательство БелГУ, 2005.-25с.
- 6 Борри, Х. Firebird: руководство разработчика баз данных: Пер. с англ. –[Текст]/ Х. Борри. . - СПб.: БХВ-Петербург, 2010. - 1104 с.
- 7 Вендров, А.М. CASE-технологии. Современные методы и средства проектирования информационных систем. [Текст]/ А.М. Вендеров. - М.: Финансы и статистика, 2010. – 176 с.
- 8 Вендров, А.М. Современные методы и средства проектирования информационных систем / А.М. Вендров. - М.: Финансы и статистика, 2012. – 65 с.
- 9 Гахова, Н.Н. Инструментальные средства информационных систем: Учебно-методический комплекс [Электронный ресурс] / Н.Н. Гахова; НИУ БелГУ. - Белгород: НИУ БелГУ, 2012. - Режим доступа: <http://pegas.bsu.edu.ru/course/view.php?id=5188>
- 10 Гахов, Р.П. Методы и средства проектирования информационных систем и технологий: Учебно-методический комплекс

[Электронный ресурс]/ Р.П. Гахов; Белгород, 2013. - Режим доступа: <http://pegas.bsu.edu.ru/course/view.php>

11 Голицына, О.Л. Программное обеспечение [Текст]/ О. Л. Голицына, И. И. Попов, Т. Л. Партыка. - М.: Форум, 2013. – 448 с.

12 Дейт К. Дж. Введение в системы баз данных. - 6-е изд. - М., СПб., Киев, Изд. дом Вильяме, 2014. – 234 с.

13 Илюшечкин, В.М. Основы использования и проектирования баз данных [Текст]/ В.М. Илюшечкин. - М.: «Издательство Юрайт» 2010. -213с.

14 Ипатова, Э.Р. Методологии и технологии системного проектирования информационных систем [Текст]/ Э.Р. Ипатова, Ю.В. Ипатов. - М.: Флинта, 2012. – 256 с.

15 Карпова, Т.С. Базы данных. Модели, разработка, реализация (2-е изд.) [Текст]/ Т.С. Карпова. - М.: НОУ "Интуит", 2016. - 403с.

16 Козлов, А.С. Проектирование и исследование бизнес-процессов: Учебное пособие [Текст]/ А.С. Козлов. - Москва: Флинта, 2011. - 268 с.

17 Конноли, Т., Базы данных. Проектирование, реализация и сопровождение. Теория и практика. 3–е издание. [Текст]/Т. Конноли, К.Бегг. - М.: Издательский дом "Вильямс", 2011. - 1440 с.

18 Культин, Н. Самоучитель С++Builder [Текст]/ Н. Культин. - СПб.: БХВ-Перербург, 2011.-203 с.

19 Лавров, С.С. Программирование. Математические основы, средства, теория: учебное пособие. [Текст]/ С.С. Лавров - СПб.: БХВ-Петербург, 2011. - 320 с.

20 Ломакин, В.В. Программирование и программное обеспечение информационных технологий: Учебно-методический комплекс [Электронный ресурс] / В.В. Ломакин; НИУ БелГУ. - Белгород, 2014 Режим доступа: <http://pegas.bsu.edu.ru/course/view.php?id=4462>

21 Маклаков, С.В. ВРwin, ERwin. CASE-средства разработки информационных систем. [Текст]/ С.В. Маклаков. - М.: ДИАЛОГ-МИФИ, 2013. – 304 с.

22 Маторин, С.И. Теория систем и системный анализ: Учебно-методический комплекс [Электронный ресурс] / С.И. Маторин, О.А. Зимовец; НИУ БелГУ. - Белгород: НИУ БелГУ, 2012. - Режим доступа: <http://pegas.bsu.edu.ru/course/view.php?id=4733>

23 Мезенцев, К.Н. Автоматизированные информационные системы [Текст]/ К.Н. Мезенцев. - М.: Академия, 2012. – 174 с

24 Михелёв, В.М. Базы данных и СУБД: Учебное пособие. [Текст]/ В.М.Михелёв. - Белгород: Издательство БелГУ, 2012. – 200 с.

25 Мозговой, М.В., С++ мастер класс. 85 нетривиальных проектов, решений и задач. [Текст]/ М.В. Мозговой. - М.: Наука и техника, 2012 г. – 234 с.

26 Муромцев, В.В. Проектирование информационных систем: Учебное пособие для студентов вузов заочной формы обучения по спец. 010502 "Прикладная информатика в экономике" / Муромцев В.В.; Рец.: В.А. Ломазов, С.И. Маторин; Федеральное агентство по образованию; Фак. КНИТ каф. прикладной информатики БелГУ; БелГУ. - Белгород: БелГУ, 2012. - 160 с.

27 Основы проектирования реляционных баз данных. [Электронный ресурс] Режим доступа: http://www.intuit.ru/goods_store/ebooks/8322, свободный.

28 Паттерсон, Д. Архитектура компьютера и проектирование компьютерных систем [Текст]/ Д. Паттерсон, Дж. Хеннесси. - СПб.: Питер, 2012. – 784 с.

29 Пахомов, Б.И. С/C++ и Borland C++ Builder для начинающих. [Текст]/ Б.И. Пахомов. - СПб.: БХВ-Петербург, 2011. – 640 с.

30 Пахомов, Б.И. С\C++ и Borland C++ Builder для студента. [Текст]/ Б.И. Пахомов. - СПб.: БХВ-Петербург, 2013. – 448 с.

31 Петров, В.Н. Информационные системы. [Текст]/ В.Н. Петров. - СПб.: Питер, 2012. – 688 с.

- 32 Послед, Б.С. Borland C++ Builder 6. Разработка приложений баз данных [Текст]/ Б.С. Послед. - СПб.: ООО «ДиаСофтЮП», 2011. –320 с.
- 33 Репин, В. - Бизнес-процессы. Моделирование, внедрение, управление. [Текст]/ В. Репин. - Москва: Флинта, 2013. - 480 с.
- 34 Ресурсы информационных систем. [Электронный ресурс]. Режим доступа: <http://www.economica-upravlenie.ru/content/view/204/>, свободный.
- 35 Семенов, М.И. Автоматизированные информационные технологии в экономике. [Текст]/ М.И. Семенов. - М.: Финансы и статистика, 2012. – 432 с.
- 36 Смирнова, Г.Н. Проектирование экономических информационных систем. Учебное пособие. [Текст]/ Г.Н. Смирнова. - М.: Высшая школа, 2012. – 428 с.
- 37 Степанов, А. Н. Информатика: [Текст]/ А.Н. Степанов. - СПб: Питер, 2010. – 720 с.
- 38 Трояновский, В. М. Проектирование информационных систем. Курс лекций. [Текст]/ В.М.Трояновский. - М.: МИЭТ, 2014. – 234 с.
- 39 Титоренко, Г.А. Автоматизированные информационные технологии в экономике: учебное пособие. [Текст]/ Г.А. Титоренко. - М.:Атлас, 2013 г. – 245 с.
- 40 Устав ОКУ «Белгородский центр занятости населения».
- 41 Федоров, Н.В. Проектирование информационных систем на основе современных CASE-технологий: учебное пособие. [Текст]/ Н. В. Федоров. - МГИУ, 2010.-128 с.
- 42 Федорова, Е.Н. Теоретические основы программирования: учебное пособие. / Е. Н. Федорова. - МГИУ, 2012.-214 с.
- 43 Фельдман, Я.А. Создаем информационную систему [Текст]/ Я.А. Фельдман. - М.: Солон-Пресс, 2011. – 120 с.
- 44 Хомоненко, А.Д. Базы данных: учебник для высших учебных заведений, 4-е издание дополненное и переработанное. [Текст]/ А.Д. Хомоненко. - СПб.: Корона, 2012. – 736 с.

45 Хомоненко, Работа с базами данных в С++ Builder. [Текст]/ А.Д.
Хомоненко, А.Д., Ададулов С.Е .. - СПб.:БХВ-Петербург, 2011.-496 с.

ПРИЛОЖЕНИЕ

Листинг программы

DataModule

```
//-----  
  
#include <vcl.h>  
#pragma hdrstop  
  
#include "DataModule.h"  
#include "Main.h"  
#include "GRAFIK.h"  
#include "SOISKATEL.h"  
#include "VAKANSII.h"  
#include "TEST.h"  
#include "VID.h"  
#include "SOTRUDNIKI.h"  
#include "NAIM.h"  
#include "PROHOZH.h"  
#include "RABOTODATELI.h"  
#include "RABOTODATELI.h"  
#include "REKOMENDACIYA.h"  
#include "REKOMENDACIYA.h"  
#include "ReportGrafik.h"  
#include "Unit2.h"  
#include "Unit3.h"  
#include "Unit4.h"  
#include "Unit5.h"  
#include "Unit6.h"  
#include "Unit7.h"  
  
//-----  
#pragma package(smart_init)  
#pragma resource "*.dfm"  
TDM *DM;  
  
//-----  
__fastcall TDM::TDM(TComponent* Owner)  
    : TDataModule(Owner)  
{  
}  
  
//-----  
//-----  
Grafik  
#include <vcl.h>  
#pragma hdrstop  
  
#include "GRAFIK.h"  
#include "DataModule.h"  
#include "Main.h"  
#include "ReportGrafik.h"  
#include "Unit4.h"  
  
//-----
```



```

#pragma package(smart_init)
#pragma resource "*.dfm"
TFGRAFIK *FGRAFIK;
//-----
__fastcall TFGRAFIK::TFGRAFIK(TComponent* Owner)
    : TForm(Owner)
{
}
//-----
void __fastcall TFGRAFIK::Button1Click(TObject *Sender)
{
DM->GRAFIK->Insert();
DBGrid1->SetFocus();
}
//-----
void __fastcall TFGRAFIK::Button3Click(TObject *Sender)
{if (DM->GRAFIK->Modified) DM->GRAFIK->Post();}
//-----
void __fastcall TFGRAFIK::Button2Click(TObject *Sender)
{
if (Application->MessageBox("Действительно хотите удалить запись?", "Подтвердите
удаление записи",
MB_YESNO + MB_ICONEXCLAMATION) == IDYES)
DM->GRAFIK->Delete();
}
//-----

void __fastcall TFGRAFIK::RadioGroup1Click(TObject *Sender)
{
if (RadioGroup1->ItemIndex == 0) DM->GRAFIK->IndexName = "PK_GRAPHIK_RABOTY";
if (RadioGroup1->ItemIndex == 1) DM->GRAFIK->IndexName = "_IDX1";
}
//-----

void __fastcall TFGRAFIK::Edit1Change(TObject *Sender)
{
TLocateOptions SearchOptions;
Variant locvalues[] = {Edit1->Text};
DM->GRAFIK->Locate("VID_GRAPHIK", VarArrayOf(locvalues,1),
SearchOptions<<loPartialKey<<loCaseInsensitive);
}
//-----

void __fastcall TFGRAFIK::Edit2Change(TObject *Sender)
{
TLocateOptions SearchOptions;
Variant locvalues[] = {Edit2->Text};
DM->GRAFIK->Locate("VREMIA_RABOTY", VarArrayOf(locvalues,1),
SearchOptions<<loPartialKey<<loCaseInsensitive);
}
//-----

```

```

void __fastcall TFGRAFIK::Button4Click(TObject *Sender)
{
Form4->QuickRep1->Preview();
}
//-----
Main
//-----

#include <vcl.h>
#pragma hdrstop

#include "Main.h"
#include "DataModule.h"
#include "GRAFIK.h"
#include "SOISKATEL.h"
#include "VAKANSII.h"
#include "VID.h"
#include "TEST.h"
#include "SOTRUDNIKI.h"
#include "NAIM.h"
#include "PROHOZH.h"
#include "RABOTODATELI.h"
#include "RECOMENDACIYA.h"
#include "Unit6.h"
//-----
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//-----
__fastcall TForm1::TForm1(TComponent* Owner)
: TForm(Owner)
{
}
//-----
void __fastcall TForm1::N2Click(TObject *Sender)
{
FGRAFIK->Show();
}
//-----
void __fastcall TForm1::N7Click(TObject *Sender)
{
FVAKANSII->Show();
}
//-----
void __fastcall TForm1::N12Click(TObject *Sender)
{
FPROHOZH->Show();
}
//-----

void __fastcall TForm1::N10Click(TObject *Sender)

```

```

{
FGRAFIK->Show();
}
//-----

void __fastcall TForm1::N9Click(TObject *Sender)
{
FVID->Show();
}
//-----

void __fastcall TForm1::N11Click(TObject *Sender)
{
FSOISKATEL->Show();
}
//-----

void __fastcall TForm1::N8Click(TObject *Sender)
{
FTEST->Show();
}
//-----

void __fastcall TForm1::N3Click(TObject *Sender)
{
FNAIM->Show();
}
//-----

void __fastcall TForm1::N4Click(TObject *Sender)
{
FSOTRUDNIKI->Show();
}
//-----

void __fastcall TForm1::N5Click(TObject *Sender)
{
Close();
}
//-----

void __fastcall TForm1::N6Click(TObject *Sender)
{
FRABOTODATEL->Show();
}
//-----

void __fastcall TForm1::N13Click(TObject *Sender)
{
FRECOMENDACIYA->Show();
}
//-----

```

```

void __fastcall TForm1::N14Click(TObject *Sender)
{
FNAIM->Show();
}
//-----

void __fastcall TForm1::Button1Click(TObject *Sender)
{
if (((ComboBox1->ItemIndex==0 ) & (Edit1->Text=="147369"))
|| ((ComboBox1->ItemIndex==1 ) & (Edit1->Text=="000005"))
|| ((ComboBox1->ItemIndex==2 ) & (Edit1->Text=="123456"))
|| ((ComboBox1->ItemIndex==3 ) & (Edit1->Text=="мемо"))
|| ((ComboBox1->ItemIndex==4 ) & (Edit1->Text=="игра"))
|| ((ComboBox1->ItemIndex==5 ) & (Edit1->Text=="вкр"))
|| ((ComboBox1->ItemIndex==6 ) & (Edit1->Text=="родина"))
|| ((ComboBox1->ItemIndex==7 ) & (Edit1->Text=="эклипс"))
|| ((ComboBox1->ItemIndex==8 ) & (Edit1->Text=="пропасть"))
|| ((ComboBox1->ItemIndex==9 ) & (Edit1->Text=="шар")))
{
ShowMessage("Доступ разрешен, выберите нужную вкладку");
Form1->Show();
Panel1->Visible = false;
N1->Enabled = true;
N2->Enabled = true;
N3->Enabled = true;

}
else{
ShowMessage("Нет доступа. Введите правильный пароль.");
Button2Click (Button2);
}

}
//-----

void __fastcall TForm1::Button2Click(TObject *Sender)
{
ComboBox1->Text="Выберите пользователя";
Edit1->Clear();
}
//-----

void __fastcall TForm1::CheckBox1Click(TObject *Sender)
{
if (CheckBox1->Checked) Edit1->PasswordChar = 0;
else
Edit1->PasswordChar = '*';
}
//-----

void __fastcall TForm1::N15Click(TObject *Sender)
{

```

```

Form6->Show();
}
//-----

//-----
Naim
#include <vcl.h>
#pragma hdrstop
#include "NAIM.h"
#include "DataModule.h"
#include "GRAFIK.h"
#include "Main.h"
#include "SOISKATEL.h"
#include "SOTRUDNIKI.h"
#include "TEST.h"
#include "VAKANSII.h"
#include "VID.h"
//-----
#pragma package(smart_init)
#pragma resource "*.dfm"
TFNAIM *FNAIM;
//-----
__fastcall TFNAIM::TFNAIM(TComponent* Owner)
    : TForm(Owner)
{
}
//-----

void __fastcall TFNAIM::Button4Click(TObject *Sender)
{
Edit1->Clear();
Edit2->Clear();
Edit3->Clear();
Edit4->Clear();
Edit5->Clear();
}
//-----

void __fastcall TFNAIM::Button5Click(TObject *Sender)
{
if (DM->NAIM->Modified) DM->NAIM->Post();
}
//-----

void __fastcall TFNAIM::Button2Click(TObject *Sender)
{
DM->IBStoredProc7->StoredProcName="DEL_NAIM";
DM->IBStoredProc7->ParamByName("ID_RABOTODATEL")->AsString=Edit1->Text;
DM->IBStoredProc7->Prepare();
DM->IBStoredProc7->ExecProc();
DM->IBTransaction1->Commit();
DM->IBQuery2->Close();
}

```

```

DM->IBQuery8->Open();
DM->IBQuery2->Open();
DM->IBQuery3->Open();
DM->IBQuery4->Open();
DM->IBQuery5->Open();
DM->IBQuery6->Open();
DM->IBQuery7->Open();
}
//-----

void __fastcall TFNAIM::Button1Click(TObject *Sender)
{
DM->IBStoredProc6->StoredProcName="ADD_NAIM";
DM->IBStoredProc6->ParamByName("ID_SOTRUDNIKA")->AsString=Edit2->Text;
DM->IBStoredProc6->ParamByName("KOL_VAKANSII")->AsString=Edit3->Text;
DM->IBStoredProc6->ParamByName("ID_FIRMI")->AsString=Edit4->Text;
DM->IBStoredProc6->ParamByName("ID_VAKANSII")->AsString=Edit5->Text;
DM->IBStoredProc6->Prepare();
DM->IBStoredProc6->ExecProc();
DM->IBTransaction1->Commit();
DM->IBQuery2->Close();
DM->IBQuery8->Open();
DM->IBQuery2->Open();
DM->IBQuery3->Open();
DM->IBQuery4->Open();
DM->IBQuery5->Open();
DM->IBQuery6->Open();
DM->IBQuery7->Open();
}
//-----

void __fastcall TFNAIM::Edit6Change(TObject *Sender)
{
DM->IBQuery2->Close();
DM->IBQuery2->SQL->Clear();
DM->IBQuery2->SQL->Add("select * from SEARCH_NAIM (:skl)");
DM->IBQuery2->ParamByName("skl")->AsString=Edit6->Text;
DM->IBQuery2->Open();
}
//-----

void __fastcall TFNAIM::Edit7Change(TObject *Sender)
{
DM->IBQuery2->Close();
DM->IBQuery2->SQL->Clear();
DM->IBQuery2->SQL->Add("select * from SEARCH_NAIM1 (:skl)");
DM->IBQuery2->ParamByName("skl")->AsString=Edit7->Text;
DM->IBQuery2->Open();
}
//-----

void __fastcall TFNAIM::Edit8Change(TObject *Sender)

```

```

{
DM->IBQuery2->Close();
DM->IBQuery2->SQL->Clear();
DM->IBQuery2->SQL->Add("select * from SEARCH_NAIM2 (:skl)");
DM->IBQuery2->ParamByName("skl")->AsString=Edit8->Text;
DM->IBQuery2->Open();
}
//-----

void __fastcall TFNAIM::RadioGroup1Click(TObject *Sender)
{
if (RadioGroup1->ItemIndex == 0) DM->NAIM->IndexName = "SORT_NAIM";
if (RadioGroup1->ItemIndex == 1) DM->NAIM->IndexName = "SORT_NAIM2";
}
//-----

void __fastcall TFNAIM::RadioButton1Click(TObject *Sender)
{
Label1->Enabled = false;
Edit1->Enabled = false;
Label2->Enabled = true;
Edit2->Enabled = true;
Label3->Enabled = true;
Edit3->Enabled = true;
Label4->Enabled = true;
Edit4->Enabled = true;
Edit5->Enabled = true;
Label5->Enabled = true;
Button2->Enabled = false;
Button1->Enabled = true;
Button4->Enabled = true;
Button5->Enabled = true;
}
//-----

void __fastcall TFNAIM::RadioButton2Click(TObject *Sender)
{
Label1->Enabled = true;
Edit1->Enabled = true;
Edit2->Enabled = false;
Label2->Enabled = false;
Button2->Enabled = true;
Edit3->Enabled = false;
Label3->Enabled = false;
Edit3->Enabled = false;
Label4->Enabled = false;
Edit4->Enabled = false;
Label5->Enabled = false;
Edit5->Enabled = false;
Button4->Enabled = true;
Button5->Enabled = true;
}

```

```

Button1->Enabled = false;
}
//-----

void __fastcall TFNAIM::Button3Click(TObject *Sender)
{
AnsiString d;
switch (RadioGroup1->ItemIndex)
{
case 0: d = ""; break;
case 1: d = " DESC";
}
DM->IBQuery2->Close();
DM->IBQuery2->SQL->Clear();
DM->IBQuery2->SQL->Add("select*from NAIM");
switch (RadioGroup2->ItemIndex)
{
case 0: d = "ORDER BY KOL_VAKANSII " + d; break;
case 1: d = "ORDER BY ID_RABOTODATEL " + d;
}
DM->IBQuery2->SQL->Add(d);
DM->IBQuery2->Open();
}
//-----
PROHOZH
//-----

#include <vcl.h>
#pragma hdrstop

#include "PROHOZH.h"
#include "DataModule.h"
#include "GRAFIK.h"
#include "Main.h"
#include "NAIM.h"
#include "SOISKATEL.h"
#include "SOTRUDNIKI.h"
#include "TEST.h"
#include "VAKANSII.h"
#include "VID.h"
//-----
#pragma package(smart_init)
#pragma resource "*.dfm"
TFPROHOZH *FPROHOZH;
//-----
__fastcall TFPROHOZH::TFPROHOZH(TComponent* Owner)
: TForm(Owner)
{
}
//-----

void __fastcall TFPROHOZH::Button1Click(TObject *Sender)
{

```



```

DM->IBStoredProc11->StoredProcName="ADD_PROHOZH";
DM->IBStoredProc11->ParamByName("ID_SOISKATEL")->AsString=Edit2->Text;
DM->IBStoredProc11->ParamByName("ID_TESTA")->AsString=Edit6->Text;
DM->IBStoredProc11->Prepare();
DM->IBStoredProc11->ExecProc();
DM->IBTransaction1->Commit();
DM->IBQuery3->Close();
DM->IBQuery8->Open();
DM->IBQuery2->Open();
DM->IBQuery3->Open();
DM->IBQuery4->Open();
DM->IBQuery5->Open();
DM->IBQuery6->Open();
DM->IBQuery7->Open();
}
//-----

```

```

void __fastcall TFPROHOZH::Button3Click(TObject *Sender)
{
DM->IBStoredProc12->StoredProcName="DEL_PROHOZH";
DM->IBStoredProc12->ParamByName("ID_PROHOZH")->AsString=Edit1->Text;
DM->IBStoredProc12->Prepare();
DM->IBStoredProc12->ExecProc();
DM->IBTransaction1->Commit();
DM->IBQuery3->Close();
DM->IBQuery8->Open();
DM->IBQuery2->Open();
DM->IBQuery3->Open();
DM->IBQuery4->Open();
DM->IBQuery5->Open();
DM->IBQuery6->Open();
DM->IBQuery7->Open();
}
//-----

```

```

void __fastcall TFPROHOZH::Button5Click(TObject *Sender)
{
Edit1->Clear();
Edit2->Clear();
Edit6->Clear();
}
//-----

```

```

void __fastcall TFPROHOZH::Button4Click(TObject *Sender)
{
if (DM->PROHOZH->Modified) DM->PROHOZH->Post();
}
//-----

```

```

void __fastcall TFPROHOZH::Edit3Change(TObject *Sender)
{
DM->IBQuery3->Close();
}

```

```

DM->IBQuery3->SQL->Clear();
DM->IBQuery3->SQL->Add("select * from SEARCH_PROHOZH (:skl)");
DM->IBQuery3->ParamByName("skl")->AsString=Edit3->Text;
DM->IBQuery3->Open();
}
//-----

void __fastcall TFPROHOZH::Edit4Change(TObject *Sender)
{
DM->IBQuery3->Close();
DM->IBQuery3->SQL->Clear();
DM->IBQuery3->SQL->Add("select * from SEARCH_PROHOZH1 (:skl)");
DM->IBQuery3->ParamByName("skl")->AsString=Edit4->Text;
DM->IBQuery3->Open();
}
//-----

void __fastcall TFPROHOZH::Edit5Change(TObject *Sender)
{
DM->IBQuery3->Close();
DM->IBQuery3->SQL->Clear();
DM->IBQuery3->SQL->Add("select * from SEARCH_PROHOZH2 (:skl)");
DM->IBQuery3->ParamByName("skl")->AsString=Edit5->Text;
DM->IBQuery3->Open();
}
//-----

void __fastcall TFPROHOZH::RadioButton1Click(TObject *Sender)
{
Label1->Enabled = false;
Edit1->Enabled = false;
Label2->Enabled = true;
Edit2->Enabled = true;
Label3->Enabled = true;
Edit6->Enabled = true;
Button3->Enabled = false;
Button1->Enabled = true;
Button4->Enabled = true;
Button5->Enabled = true;
}
//-----

void __fastcall TFPROHOZH::RadioButton2Click(TObject *Sender)
{
Label1->Enabled = true;
Edit1->Enabled = true;
Label2->Enabled = false;

```

```

Edit2->Enabled = false;
Label3->Enabled = false;
Edit6->Enabled = false;
Button1->Enabled = false;
Button3->Enabled = true;
Button5->Enabled = true;
Button4->Enabled = true;

}
//-----

void __fastcall TFPROHOZH::Button2Click(TObject *Sender)
{
AnsiString d;
switch (RadioGroup2->ItemIndex)
{
case 0: d = ""; break;
case 1: d = " DESC";
}
DM->IBQuery3->Close();
DM->IBQuery3->SQL->Clear();
DM->IBQuery3->SQL->Add("select*from PROHOZH");
switch (RadioGroup3->ItemIndex)
{
case 0: d = "ORDER BY ID_PROHOZH" + d; break;
case 1: d = "ORDER BY ID_SOISKATEL" + d;
}
DM->IBQuery3->SQL->Add(d);
DM->IBQuery3->Open();
}
//-----

RABOTODATEL
//-----

#include <vcl.h>
#pragma hdrstop

#include "RABOTODATELI.h"
#include "DataModule.h"
#include "GRAFIK.h"
#include "Main.h"
#include "NAIM.h"
#include "PROHOZH.h"
#include "SOISKATEL.h"
#include "SOTRUDNIKI.h"
#include "TEST.h"
#include "VAKANSII.h"
#include "VID.h"
#include "Unit2.h"
//-----
#pragma package(smart_init)
#pragma resource "*.dfm"
TFRABOTODATEL *FRABOTODATEL;

```

```

//-----
__fastcall TFRABOTODATEL::TFRABOTODATEL(TComponent* Owner)
    : TForm(Owner)
{
}
//-----

void __fastcall TFRABOTODATEL::Button2Click(TObject *Sender)
{
DM->IBStoredProc17->StoredProcName="DEL_RABOTODATEL";
DM->IBStoredProc17->ParamByName("ID_FIRMI")->AsString=Edit7->Text;
DM->IBStoredProc17->Prepare();
DM->IBStoredProc17->ExecProc();
DM->IBTransaction1->Commit();
DM->IBQuery4->Close();
DM->IBQuery8->Open();
DM->IBQuery2->Open();
DM->IBQuery3->Open();
DM->IBQuery4->Open();
DM->IBQuery5->Open();
DM->IBQuery6->Open();
DM->IBQuery7->Open();
}
//-----

void __fastcall TFRABOTODATEL::Edit4Change(TObject *Sender)
{
DM->IBQuery4->Close();
DM->IBQuery4->SQL->Clear();
DM->IBQuery4->SQL->Add("select * from SEARCH_RABOTOD (:skl)");
DM->IBQuery4->ParamByName("skl")->AsString=Edit4->Text;
DM->IBQuery4->Open();
}
//-----

void __fastcall TFRABOTODATEL::Edit5Change(TObject *Sender)
{
DM->IBQuery4->Close();
DM->IBQuery4->SQL->Clear();
DM->IBQuery4->SQL->Add("select * from SEARCH_RABOTOD1 (:skl)");
DM->IBQuery4->ParamByName("skl")->AsString=Edit5->Text;
DM->IBQuery4->Open();}
//-----

void __fastcall TFRABOTODATEL::Edit6Change(TObject *Sender)
{
DM->IBQuery4->Close();
DM->IBQuery4->SQL->Clear();
DM->IBQuery4->SQL->Add("select * from SEARCH_RABOTOD2 (:skl)");
DM->IBQuery4->ParamByName("skl")->AsString=Edit6->Text;
DM->IBQuery4->Open();
}
//-----

```

```

void __fastcall TFRABOTODATEL::Button4Click(TObject *Sender)
{
Edit1->Clear();
Edit2->Clear();
Edit3->Clear();
Edit7->Clear();
}
//-----

void __fastcall TFRABOTODATEL::Button7Click(TObject *Sender)
{
if (DM->RABOTODATEL->Modified) DM->RABOTODATEL->Post();
}
//-----

void __fastcall TFRABOTODATEL::Button1Click(TObject *Sender)
{
DM->IBStoredProc16->StoredProcName="ADD_RABOTODATEL";
DM->IBStoredProc16->ParamByName("NAZVANIE_FIRMI")->AsString=Edit1->Text;
DM->IBStoredProc16->ParamByName("UR_ADRES_FIRMI")->AsString=Edit2->Text;
DM->IBStoredProc16->ParamByName("TELEFON")->AsString=Edit3->Text;
DM->IBStoredProc16->Prepare();
DM->IBStoredProc16->ExecProc();
DM->IBTransaction1->Commit();
DM->IBQuery4->Close();
DM->IBQuery8->Open();
DM->IBQuery2->Open();
DM->IBQuery3->Open();
DM->IBQuery4->Open();
DM->IBQuery5->Open();
DM->IBQuery6->Open();
DM->IBQuery7->Open();

}
//-----

void __fastcall TFRABOTODATEL::Button3Click(TObject *Sender)
{
Form2->QuickRep1->Preview();
}
//-----

void __fastcall TFRABOTODATEL::RadioButton1Click(TObject *Sender)
{
Label7->Enabled = false;
Edit7->Enabled = false;
Label1->Enabled = true;
Edit1->Enabled = true;
Label2->Enabled = true;
}

```

```

Edit2->Enabled = true;
Label3->Enabled = true;
Edit3->Enabled = true;
Button2->Enabled = false;
Button4->Enabled = true;
Button7->Enabled =true;
Button1->Enabled = true;
}
//-----

void __fastcall TFRABOTODATEL::RadioButton2Click(TObject *Sender)
{
Label7->Enabled = true;
Edit7->Enabled = true;
Label1->Enabled = false;
Edit1->Enabled = false;
Label2->Enabled = false;
Edit2->Enabled = false;
Label3->Enabled = false;
Edit3->Enabled = false;
Button2->Enabled = true;
Button4->Enabled = true;
Button7->Enabled =true;
Button1->Enabled = false;

}
//-----

void __fastcall TFRABOTODATEL::Button5Click(TObject *Sender)
{
AnsiString d;
switch (RadioGroup2->ItemIndex)
{
case 0: d = ""; break;
case 1: d = " DESC";
}
DM->IBQuery4->Close();
DM->IBQuery4->SQL->Clear();
DM->IBQuery4->SQL->Add("select*from RABOTODATEL");
switch (RadioGroup3->ItemIndex)
{
case 0: d = "ORDER BY ID_FIRMI" + d; break;
case 1: d = "ORDER BY NAZVANIE_FIRMI" + d;
}
DM->IBQuery4->SQL->Add(d);
DM->IBQuery4->Open();
}
//-----

REKOMENDACIYA
//-----

```

```

#include <vcl.h>
#pragma hdrstop

#include "RECOMENDACIYA.h"
#include "DataModule.h"
#include "GRAFIK.h"
#include "Main.h"
#include "NAIM.h"
#include "PROHOZH.h"
#include "RABOTODATELI.h"
#include "SOISKATEL.h"
#include "SOTRUDNIKI.h"
#include "TEST.h"
#include "VAKANSII.h"
#include "VID.h"
//-----
#pragma package(smart_init)
#pragma resource "*.dfm"
TFRECOMENDACIYA *FRECOMENDACIYA;
//-----
__fastcall TFRECOMENDACIYA::TFRECOMENDACIYA(TComponent* Owner)
    : TForm(Owner)
{
}
//-----

void __fastcall TFRECOMENDACIYA::Button5Click(TObject *Sender)
{
Edit1->Clear();
Edit2->Clear();
Edit3->Clear();
Edit4->Clear();
Edit5->Clear();
Edit6->Clear();
Edit7->Clear();
}
//-----

void __fastcall TFRECOMENDACIYA::Button4Click(TObject *Sender)
{
if (DM->RECOMENDACIYA->Modified) DM->RECOMENDACIYA->Post();
}
//-----

void __fastcall TFRECOMENDACIYA::Button2Click(TObject *Sender)
{
DM->IBStoredProc30->StoredProcName="DEL_REKOM";
}

```

```

DM->IBStoredProc30->ParamByName("ID_RECOMENDACII")->AsString=Edit1->Text;
DM->IBStoredProc30->Prepare();
DM->IBStoredProc30->ExecProc();
DM->IBTransaction1->Commit();
DM->IBQuery7->Close();
DM->IBQuery8->Open();
DM->IBQuery2->Open();
DM->IBQuery3->Open();
DM->IBQuery4->Open();
DM->IBQuery5->Open();
DM->IBQuery6->Open();
DM->IBQuery7->Open();
}
//-----

```

```

void __fastcall TFRECOMENDACIYA::Button1Click(TObject *Sender)
{
DM->IBStoredProc29->StoredProcName="ADD_REKOM";
DM->IBStoredProc29->ParamByName("ID_SOTRUDNIKA")->AsString=Edit2->Text;
DM->IBStoredProc29->ParamByName("ID_PROHOZH")->AsString=Edit3->Text;
DM->IBStoredProc29->ParamByName("ID_GRAPHIK")->AsString=Edit4->Text;
DM->IBStoredProc29->ParamByName("ID_SOISKATEL")->AsString=Edit5->Text;
DM->IBStoredProc29->ParamByName("ID_VAKANSII")->AsString=Edit6->Text;
DM->IBStoredProc29->ParamByName("REZULTAT_TESTA")->AsString=Edit7->Text;
DM->IBStoredProc29->Prepare();
DM->IBStoredProc29->ExecProc();
DM->IBTransaction1->Commit();
DM->IBQuery7->Close();
DM->IBQuery8->Open();
DM->IBQuery2->Open();
DM->IBQuery3->Open();
DM->IBQuery4->Open();
DM->IBQuery5->Open();
DM->IBQuery6->Open();
DM->IBQuery7->Open();

}
//-----

```

```

void __fastcall TFRECOMENDACIYA::Edit8Change(TObject *Sender)
{
DM->IBQuery7->Close();
DM->IBQuery7->SQL->Clear();
DM->IBQuery7->SQL->Add("select * from SEARCH_REKOM (:skl)");
DM->IBQuery7->ParamByName("skl")->AsString=Edit8->Text;
DM->IBQuery7->Open();
}
//-----

```

```

void __fastcall TFRECOMENDACIYA::Edit9Change(TObject *Sender)
{
DM->IBQuery7->Close();

```



```

DM->IBQuery7->SQL->Clear();
DM->IBQuery7->SQL->Add("select * from SEARCH_REKOM1 (:skl)");
DM->IBQuery7->ParamByName("skl")->AsString=Edit9->Text;
DM->IBQuery7->Open();
}
//-----

```

```

void __fastcall TFRECOMENDACIYA::Edit10Change(TObject *Sender)
{
DM->IBQuery7->Close();
DM->IBQuery7->SQL->Clear();
DM->IBQuery7->SQL->Add("select * from SEARCH_REKOM2 (:skl)");
DM->IBQuery7->ParamByName("skl")->AsString=Edit10->Text;
DM->IBQuery7->Open();
}
//-----

```

```

void __fastcall TFRECOMENDACIYA::RadioButton1Click(TObject *Sender)
{
Label1->Enabled = false;
Edit1->Enabled = false;
Button1->Enabled = true;
Label2->Enabled = true;
Edit2->Enabled = true;
Button2->Enabled = false;
Label3->Enabled = true;
Edit3->Enabled = true;
Button4->Enabled = true;
Label4->Enabled = true;
Edit4->Enabled = true;
Button5->Enabled = true;
Label5->Enabled = true;
Edit5->Enabled = true;
Label6->Enabled = true;
Edit6->Enabled = true;
Label7->Enabled = true;
Edit7->Enabled = true;
}
//-----

```

```

void __fastcall TFRECOMENDACIYA::RadioButton2Click(TObject *Sender)
{
Label1->Enabled = true;
Edit1->Enabled = true;
Button2->Enabled = true;
Label2->Enabled = false;
Edit2->Enabled = false;
Label3->Enabled = false;
Edit3->Enabled = false;
Button4->Enabled = true;
}

```

```

Label4->Enabled = false;
Edit4->Enabled = false;
Button5->Enabled = true;
Label5->Enabled = false;
Edit5->Enabled = false;
Label6->Enabled = false;
Edit6->Enabled = false;
Label7->Enabled = false;
Edit7->Enabled = false;
Button1->Enabled = false;

}
//-----

void __fastcall TFRECOMENDACIYA::Button3Click(TObject *Sender)
{
AnsiString d;
switch (RadioGroup2->ItemIndex)
{
case 0: d = ""; break;
case 1: d = " DESC";
}
DM->IBQuery7->Close();
DM->IBQuery7->SQL->Clear();
DM->IBQuery7->SQL->Add("select*from RECOMENDACIYA");
switch (RadioGroup3->ItemIndex)
{
case 0: d = "ORDER BY ID_RECOMENDACII" + d; break;
case 1: d = "ORDER BY REZULTAT_TESTA" + d;
}
DM->IBQuery7->SQL->Add(d);
DM->IBQuery7->Open();
}
//-----
REPORTGRAFIK
//-----

#include <vcl.h>
#pragma hdrstop

#include "ReportGrafik.h"
#include "DataModule.h"
#include "GRAFIK.h"
//-----
#pragma package(smart_init)
#pragma resource "*.dfm"
TRepGrafik *RepGrafik;
//-----
__fastcall TRepGrafik::TRepGrafik(TComponent* Owner)
: TForm(Owner)
{
}
//-----
SOISKIATEL

```

```

//-----

#include <vcl.h>
#pragma hdrstop

#include "SOISKATEL.h"
#include "DataModule.h"
#include "Main.h"
#include "Unit5.h"
#include "Unit6.h"
//-----

#pragma package(smart_init)
#pragma resource "*.dfm"
TFSOISKATEL *FSOISKATEL;
//-----
__fastcall TFSOISKATEL::TFSOISKATEL(TComponent* Owner)
    : TForm(Owner)
{
}
//-----

void __fastcall TFSOISKATEL::Button1Click(TObject *Sender)
{
DM->IBStoredProc21->StoredProcName="ADD_SOISK";
DM->IBStoredProc21->ParamByName("ID_GRAPHIK")->AsString=Edit2->Text;
DM->IBStoredProc21->ParamByName("FAMILIA")->AsString=Edit3->Text;
DM->IBStoredProc21->ParamByName("NAME")->AsString=Edit6->Text;
DM->IBStoredProc21->ParamByName("OTCHESTVO")->AsString=Edit7->Text;
DM->IBStoredProc21->ParamByName("STAZH_RABOTY")->AsString=Edit8->Text;
DM->IBStoredProc21->ParamByName("OBRAZOVANIE")->AsString=Edit4->Text;
DM->IBStoredProc21->ParamByName("ZARABOTNAYA_PLATA")->AsString=Edit5->Text;
DM->IBStoredProc21->Prepare();
DM->IBStoredProc21->ExecProc();
DM->IBTransaction1->Commit();
DM->IBQuery5->Close();
DM->IBQuery8->Open();
DM->IBQuery2->Open();
DM->IBQuery3->Open();
DM->IBQuery4->Open();
DM->IBQuery5->Open();
DM->IBQuery6->Open();
DM->IBQuery7->Open();
}
//-----

void __fastcall TFSOISKATEL::Button6Click(TObject *Sender)

```

```

{
Edit1->Clear();
Edit2->Clear();
Edit3->Clear();
Edit6->Clear();
Edit7->Clear();
Edit8->Clear();
Edit4->Clear();
Edit5->Clear();
}
//-----

void __fastcall TFSOISKATEL::Button2Click(TObject *Sender)
{
DM->IBStoredProc22->StoredProcName="DEL_SOISK";
DM->IBStoredProc22->ParamByName("ID_SOISKATEL")->AsString=Edit1->Text;
DM->IBStoredProc22->Prepare();
DM->IBStoredProc22->ExecProc();
DM->IBTransaction1->Commit();
DM->IBQuery5->Close();
DM->IBQuery8->Open();
DM->IBQuery2->Open();
DM->IBQuery3->Open();
DM->IBQuery4->Open();
DM->IBQuery5->Open();
DM->IBQuery6->Open();
DM->IBQuery7->Open();
}
//-----

void __fastcall TFSOISKATEL::Button4Click(TObject *Sender)
{
if (DM->SOISKATEL->Modified) DM->SOISKATEL->Post();
}
//-----

void __fastcall TFSOISKATEL::Edit9Change(TObject *Sender)
{
DM->IBQuery5->Close();
DM->IBQuery5->SQL->Clear();
DM->IBQuery5->SQL->Add("select * from SEARCH_SOISK (:skl)");
DM->IBQuery5->ParamByName("skl")->AsString=Edit9->Text;
DM->IBQuery5->Open();
}
//-----

void __fastcall TFSOISKATEL::Edit11Change(TObject *Sender)
{
DM->IBQuery5->Close();
DM->IBQuery5->SQL->Clear();
DM->IBQuery5->SQL->Add("select * from SEARCH_SOISK1 (:skl)");
}

```

```
DM->IBQuery5->ParamByName("sk1")->AsString=Edit11->Text;
DM->IBQuery5->Open();
}
//-----
```

```
void __fastcall TFSOISKATEL::Button5Click(TObject *Sender)
{
Form5->QuickRep1->Preview();
}
//-----
```

```
void __fastcall TFSOISKATEL::RadioButton1Click(TObject *Sender)
{
Label1->Enabled = false;
Edit1->Enabled = false;
Button1->Enabled = true;
Label2->Enabled = true;
Edit2->Enabled = true;
Label3->Enabled = true;
Edit3->Enabled = true;
Button2->Enabled = false;
Label6->Enabled = true;
Edit6->Enabled = true;
Button4->Enabled = false;
Label7->Enabled = true;
Edit7->Enabled = true;
Label8->Enabled = true;
Edit8->Enabled = true;
Label4->Enabled = true;
Edit4->Enabled = true;
Label5->Enabled = true;
Edit5->Enabled = true;
Button4->Enabled = true;
Button6->Enabled = true;
}
//-----
```

```
void __fastcall TFSOISKATEL::RadioButton2Click(TObject *Sender)
{
Label1->Enabled = true;
Edit1->Enabled = true;
Button1->Enabled = false;
Label2->Enabled = false;
Edit2->Enabled = false;
Label3->Enabled = false;
Edit3->Enabled = false;
Button2->Enabled = true;
Label6->Enabled = false;
Edit6->Enabled = false;
```

```

Button4->Enabled = false;
Label7->Enabled = false;
Edit7->Enabled = false;
Label8->Enabled = false;
Edit8->Enabled = false;
Label4->Enabled = false;
Edit4->Enabled = false;
Label5->Enabled = false;
Edit5->Enabled = false;
Button4->Enabled = true;
Button6->Enabled = true;
}
//-----

void __fastcall TFSOISKATEL::Button3Click(TObject *Sender)
{
AnsiString d;
switch (RadioGroup2->ItemIndex)
{
case 0: d = ""; break;
case 1: d = " DESC";
}
DM->IBQuery5->Close();
DM->IBQuery5->SQL->Clear();
DM->IBQuery5->SQL->Add("select*from SOISKATEL");
switch (RadioGroup3->ItemIndex)
{
case 0: d = "ORDER BY ID_SOISKATEL" + d; break;
case 1: d = "ORDER BY Familia" + d;
}
DM->IBQuery5->SQL->Add(d);
DM->IBQuery5->Open();
}
//-----
SOTRUDNIKI
//-----

#include <vcl.h>
#pragma hdrstop

#include "SOTRUDNIKI.h"
#include "DataModule.h"
#include "GRAFIK.h"
#include "Main.h"
#include "SOISKATEL.h"
#include "TEST.h"
#include "VAKANSII.h"
#include "VID.h"
//-----
#pragma package(smart_init)
#pragma resource "*.dfm"
TFSOTRUDNIKI *FSOTRUDNIKI;
//-----
__fastcall TFSOTRUDNIKI::TFSOTRUDNIKI(TComponent* Owner)

```

```

        : TForm(Owner)
    {
    }
//-----
void __fastcall TFSOTRUDNIKI::Button3Click(TObject *Sender)
{
DM->IBStoredProc26->StoredProcName="DEL_SOTR";
DM->IBStoredProc26->ParamByName("ID_SOTRUDNIKA")->AsString=Edit1->Text;
DM->IBStoredProc26->Prepare();
DM->IBStoredProc26->ExecProc();
DM->IBTransaction1->Commit();
DM->IBQuery6->Close();
DM->IBQuery8->Open();
DM->IBQuery2->Open();
DM->IBQuery3->Open();
DM->IBQuery4->Open();
DM->IBQuery5->Open();
DM->IBQuery6->Open();
DM->IBQuery7->Open();
}
//-----

void __fastcall TFSOTRUDNIKI::Button5Click(TObject *Sender)
{
Edit1->Clear();
Edit2->Clear();
Edit3->Clear();
Edit4->Clear();
Edit5->Clear();
}
//-----

void __fastcall TFSOTRUDNIKI::Button1Click(TObject *Sender)
{
DM->IBStoredProc25->StoredProcName="ADD_SOTR";
DM->IBStoredProc25->ParamByName("FAMILIA")->AsString=Edit2->Text;
DM->IBStoredProc25->ParamByName("NAME")->AsString=Edit3->Text;
DM->IBStoredProc25->ParamByName("OTCHESTVO")->AsString=Edit4->Text;
DM->IBStoredProc25->ParamByName("DOLZHNOST")->AsString=Edit5->Text;
DM->IBStoredProc25->Prepare();
DM->IBStoredProc25->ExecProc();
DM->IBTransaction1->Commit();
DM->IBQuery6->Close();
DM->IBQuery8->Open();
DM->IBQuery2->Open();
DM->IBQuery3->Open();
DM->IBQuery4->Open();
DM->IBQuery5->Open();
DM->IBQuery6->Open();
DM->IBQuery7->Open();
}

```

```

//-----

void __fastcall TFSOTRUDNIKI::Button4Click(TObject *Sender)
{
if (DM->SOTRUDNIKI->Modified) DM->SOTRUDNIKI->Post();
}
//-----

void __fastcall TFSOTRUDNIKI::Edit6Change(TObject *Sender)
{
DM->IBQuery6->Close();
DM->IBQuery6->SQL->Clear();
DM->IBQuery6->SQL->Add("select * from SEARCH_SOTR (:skl)");
DM->IBQuery6->ParamByName("skl")->AsString=Edit6->Text;
DM->IBQuery6->Open();
}
//-----

void __fastcall TFSOTRUDNIKI::Edit8Change(TObject *Sender)
{
DM->IBQuery6->Close();
DM->IBQuery6->SQL->Clear();
DM->IBQuery6->SQL->Add("select * from SEARCH_SOTR1 (:skl)");
DM->IBQuery6->ParamByName("skl")->AsString=Edit8->Text;
DM->IBQuery6->Open();
}
//-----

void __fastcall TFSOTRUDNIKI::RadioButton1Click(TObject *Sender)
{
Label1->Enabled = false;
Edit1->Enabled = false;
Button1->Enabled = true;
Label2->Enabled = true;
Edit2->Enabled = true;
Label3->Enabled = true;
Edit3->Enabled = true;
Label4->Enabled = true;
Edit4->Enabled = true;
Label5->Enabled = true;
Edit5->Enabled = true;
Button3->Enabled = false;
Button4->Enabled = true;
Button5->Enabled = true;

}
//-----

void __fastcall TFSOTRUDNIKI::RadioButton2Click(TObject *Sender)
{

```



```

Label1->Enabled = true;
Edit1->Enabled = true;
Button1->Enabled = false;
Label2->Enabled = false;
Edit2->Enabled = false;
Label3->Enabled = false;
Edit3->Enabled = false;
Label4->Enabled = false;
Edit4->Enabled = false;
Label5->Enabled = false;
Edit5->Enabled = false;
Button4->Enabled = true;
Button5->Enabled = true;
Button3->Enabled = true;

}
//-----

void __fastcall TFSOTRUDNIKI::Button2Click(TObject *Sender)
{
AnsiString d;
switch (RadioGroup2->ItemIndex)
{
case 0: d = ""; break;
case 1: d = " DESC";
}
DM->IBQuery6->Close();
DM->IBQuery6->SQL->Clear();
DM->IBQuery6->SQL->Add("select*from SOTRUDNIKI");
switch (RadioGroup3->ItemIndex)
{
case 0: d = "ORDER BY ID_SOTRUDNIKA" + d; break;
case 1: d = "ORDER BY FAMILIA" + d;
}
DM->IBQuery6->SQL->Add(d);
DM->IBQuery6->Open();
}
//-----
TEST
//-----

#include <vcl.h>
#pragma hdrstop

#include "TEST.h"
#include "DataModule.h"
#include "GRAFIK.h"
#include "Main.h"
#include "SOISKATEL.h"
#include "VAKANSII.h"
#include "VID.h"
//-----
#pragma package(smart_init)
#pragma resource "*.dfm"

```

```

TFTEST *FTEST;
//-----
__fastcall TFTEST::TFTEST(TComponent* Owner)
    : TForm(Owner)
{
}
//-----

void __fastcall TFTEST::Button1Click(TObject *Sender)
{
DM->TEST->Insert();
DBGrid1->SetFocus();
}
//-----

void __fastcall TFTEST::Button2Click(TObject *Sender)
{
if (Application->MessageBox("Действительно хотите удалить запись?", "Подтвердите
удаление записи",
MB_YESNO + MB_ICONEXCLAMATION) == IDYES)
DM->TEST->Delete();
}
//-----

void __fastcall TFTEST::Button4Click(TObject *Sender)
{
if (DM->TEST->Modified) DM->TEST->Post();
}
//-----

void __fastcall TFTEST::RadioGroup1Click(TObject *Sender)
{
if (RadioGroup1->ItemIndex == 0) DM->TEST->IndexName = "PK_TEST";
if (RadioGroup1->ItemIndex == 1) DM->TEST->IndexName = "PK_TEST2";
}
//-----

void __fastcall TFTEST::Edit4Change(TObject *Sender)
{
TLocateOptions SearchOptions;
Variant locvalues[] = {Edit4->Text};
DM->TEST->Locate("ID_VIDA", VarArrayOf(locvalues,1),
SearchOptions<<loPartialKey<<loCaseInsensitive);
}
//-----
Unit2
//-----

```

```

#include <vcl.h>
#pragma hdrstop

#include "Unit2.h"
#include "RABOTODATELI.h"
#include "DataModule.h"
#include "Main.h"
//-----
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm2 *Form2;
//-----
__fastcall TForm2::TForm2(TComponent* Owner)
    : TForm(Owner)
{
}
//-----
Unit3
//-----

#include <vcl.h>
#pragma hdrstop

#include "Unit3.h"
#include "VAKANSII.h"
#include "DataModule.h"
//-----
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm3 *Form3;
//-----
__fastcall TForm3::TForm3(TComponent* Owner)
    : TForm(Owner)
{
}
//-----
Unit4
//-----

#include <vcl.h>
#pragma hdrstop

#include "Unit4.h"
#include "DataModule.h"
#include "GRAFIK.h"
//-----
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm4 *Form4;
//-----
__fastcall TForm4::TForm4(TComponent* Owner)

```

```

        : TForm(Owner)
    {
    }
//-----
Unit5
//-----

#include <vcl.h>
#pragma hdrstop

#include "Unit5.h"
#include "DataModule.h"
#include "SOISKATEL.h"
//-----
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm5 *Form5;
//-----
__fastcall TForm5::TForm5(TComponent* Owner)
    : TForm(Owner)
{
}
//-----
Unit6
//-----

#include <vcl.h>
#pragma hdrstop

#include "Unit6.h"
#include "SOISKATEL.h"
#include "Main.h"
#include "DataModule.h"
//-----
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm6 *Form6;
//-----
__fastcall TForm6::TForm6(TComponent* Owner)
    : TForm(Owner)
{
}
//-----
Unit7
//-----

#include <vcl.h>
#pragma hdrstop

#include "Unit7.h"
#include "DataModule.h"
//-----

```

```

#pragma package(smart_init)
#pragma resource "*.dfm"
TForm7 *Form7;
//-----
__fastcall TForm7::TForm7(TComponent* Owner)
    : TForm(Owner)
{
}
//-----
VAKANSII
//-----

#include <vcl.h>
#pragma hdrstop

#include "VAKANSII.h"
#include "DataModule.h"
#include "GRAFIK.h"
#include "Main.h"
#include "SOISKATEL.h"
#include "VID.h"
#include "Unit3.h"
//-----
#pragma package(smart_init)
#pragma resource "*.dfm"
TFVAKANSII *FVAKANSII;
//-----
__fastcall TFVAKANSII::TFVAKANSII(TComponent* Owner)
    : TForm(Owner)
{
}
//-----
void __fastcall TFVAKANSII::Button1Click(TObject *Sender)
{
    DM->IBStoredProc1->StoredProcName="ADD_VAKANSII";
    DM->IBStoredProc1->ParamByName("NAZVANIE_VAKANSII")->AsString=Edit2->Text;
    DM->IBStoredProc1->ParamByName("RAZMER_ZP")->AsString=Edit3->Text;
    DM->IBStoredProc1->ParamByName("ID_GRAPHIK")->AsString=Edit4->Text;
    DM->IBStoredProc1->ParamByName("VOZRAST_SOISK")->AsString=Edit5->Text;
    DM->IBStoredProc1->ParamByName("OPIT_RABOTY")->AsString=Edit6->Text;
    DM->IBStoredProc1->Prepare();
    DM->IBStoredProc1->ExecProc();
    DM->IBTransaction1->Commit();
    DM->IBQuery8->Close();
    DM->IBQuery8->Open();
    DM->IBQuery2->Open();
    DM->IBQuery3->Open();
    DM->IBQuery4->Open();
    DM->IBQuery5->Open();
    DM->IBQuery6->Open();
    DM->IBQuery7->Open();
}
//-----

```

```

void __fastcall TFVAKANSII::Button5Click(TObject *Sender)
{
Edit2->Clear();
Edit3->Clear();
Edit4->Clear();
Edit5->Clear();
Edit6->Clear();
}
//-----

void __fastcall TFVAKANSII::Button4Click(TObject *Sender)
{
if (DM->VAK->Modified) DM->VAK->Post();
}
//-----

void __fastcall TFVAKANSII::Button3Click(TObject *Sender)
{
DM->IBStoredProc2->StoredProcName="DEL_VAKANSII";
DM->IBStoredProc2->ParamByName("ID_VAKANSII")->AsString=Edit1->Text;
DM->IBStoredProc2->Prepare();
DM->IBStoredProc2->ExecProc();
DM->IBTransaction1->Commit();
DM->IBQuery8->Close();
DM->IBQuery8->Open();
DM->IBQuery2->Open();
DM->IBQuery3->Open();
DM->IBQuery4->Open();
DM->IBQuery5->Open();
DM->IBQuery6->Open();
DM->IBQuery7->Open();
}
//-----

void __fastcall TFVAKANSII::Edit7Change(TObject *Sender)
{
DM->IBQuery8->Close();
DM->IBQuery8->SQL->Clear();
DM->IBQuery8->SQL->Add("select * from SEARCH_VAKAN (:skl)");
DM->IBQuery8->ParamByName("skl")->AsString=Edit7->Text;
DM->IBQuery8->Open();
}
//-----

void __fastcall TFVAKANSII::Edit8Change(TObject *Sender)
{
DM->IBQuery8->Close();
DM->IBQuery8->SQL->Clear();
DM->IBQuery8->SQL->Add("select * from SEARCH_VAKAN1 (:skl)");
DM->IBQuery8->ParamByName("skl")->AsString=Edit8->Text;
DM->IBQuery8->Open();
}

```

```

}
//-----

void __fastcall TFVAKANSII::Edit9Change(TObject *Sender)
{
DM->IBQuery8->Close();
DM->IBQuery8->SQL->Clear();
DM->IBQuery8->SQL->Add("select * from SEARCH_VAKAN2 (:sk1)");
DM->IBQuery8->ParamByName("sk1")->AsString=Edit9->Text;
DM->IBQuery8->Open();
}
//-----

void __fastcall TFVAKANSII::Button6Click(TObject *Sender)
{
Form3->QuickRep1->Preview();
}
//-----

void __fastcall TFVAKANSII::RadioButton1Click(TObject *Sender)
{
Label1->Enabled = false;
Edit1->Enabled = false;
Button1->Enabled = true;
Label2->Enabled = true;
Edit2->Enabled = true;
Label6->Enabled = true;
Edit3->Enabled = true;
Label4->Enabled = true;
Edit4->Enabled = true;
Label5->Enabled = true;
Edit5->Enabled = true;
Label3->Enabled = true;
Edit6->Enabled = true;
Button4->Enabled = true;
Button5->Enabled = true;
Button3->Enabled = false;
}
//-----

void __fastcall TFVAKANSII::RadioButton2Click(TObject *Sender)
{
Label1->Enabled = true;
Edit1->Enabled = true;
Button1->Enabled = false;
Label2->Enabled = false;
Edit2->Enabled = false;
Label6->Enabled = false;
Edit3->Enabled = false;
}

```

```

Label4->Enabled = false;
Edit4->Enabled = false;
Label5->Enabled = false;
Edit5->Enabled = false;
Label3->Enabled = false;
Edit6->Enabled = false;
Button4->Enabled = false;
Button5->Enabled = false;
Button3->Enabled = true;
}
//-----

void __fastcall TFVAKANSII::Button2Click(TObject *Sender)
{
AnsiString d;
switch (RadioGroup2->ItemIndex)
{
case 0: d = ""; break;
case 1: d = " DESC";
}
DM->IBQuery8->Close();
DM->IBQuery8->SQL->Clear();
DM->IBQuery8->SQL->Add("select*from VAKANSII");
switch (RadioGroup3->ItemIndex)
{
case 0: d = "ORDER BY ID_VAKANSII" + d; break;
case 1: d = "ORDER BY NAZVANIE_VAKANSII" + d;
}
DM->IBQuery8->SQL->Add(d);
DM->IBQuery8->Open();

}
//-----
VID
//-----

#include <vcl.h>
#pragma hdrstop

#include "VID.h"
#include "DataModule.h"
#include "Main.h"
//-----
#pragma package(smart_init)
#pragma resource "*.dfm"
TFVID *FVID;
//-----
__fastcall TFVID::TFVID(TComponent* Owner)
: TForm(Owner)
{
}
//-----

```



```

void __fastcall TFVID::Button1Click(TObject *Sender)
{
DM->VID->Insert();
DBGrid1->SetFocus();
}
//-----

void __fastcall TFVID::Button3Click(TObject *Sender)
{
if (Application->MessageBox("Действительно хотите удалить запись?","Подтвердите
удаление записи",
MB_YESNO + MB_ICONEXCLAMATION) == IDYES)
DM->VID->Delete();
}
//-----

void __fastcall TFVID::Button2Click(TObject *Sender)
{
if (DM->VID->Modified) DM->VID->Post();
}
//-----

void __fastcall TFVID::Edit3Change(TObject *Sender)
{
TLocateOptions SearchOptions;
Variant locvalues[] = {Edit3->Text};
DM->VID->Locate("NAIMENOVANIE", VarArrayOf(locvalues,1),
SearchOptions<<loPartialKey<<loCaseInsensitive);
}
//-----

void __fastcall TFVID::RadioGroup1Click(TObject *Sender)
{
if (RadioGroup1->ItemIndex == 0) DM->VID->IndexName = "PK_VID TESTA";
if (RadioGroup1->ItemIndex == 1) DM->VID->IndexName = "VID_TESTA_SORT";
}
//-----
VKR
//-----

#include <vcl.h>
#pragma hdrstop
//-----
USEFORM("Main.cpp", Form1);
USEFORM("DataModule.cpp", DM); /* TDataModule: File Type */
USEFORM("GRAFIK.cpp", FGRAFIK);
USEFORM("SOISKATEL.cpp", FSOISKATEL);
USEFORM("VID.cpp", FVID);
USEFORM("VAKANSII.cpp", FVAKANSII);
USEFORM("TEST.cpp", FTEST);
USEFORM("SOTRUDNIKI.cpp", FSOTRUDNIKI);

```

```

USEFORM("NAIM.cpp", FNAIM);
USEFORM("PROHOZH.cpp", FPROHOZH);
USEFORM("RABOTODATELI.cpp", FRABOTODATEL);
USEFORM("RECOMENDACIYA.cpp", FRECOMENDACIYA);
USEFORM("ReportGrafik.cpp", RepGrafik);
USEFORM("Unit2.cpp", Form2);
USEFORM("Unit3.cpp", Form3);
USEFORM("Unit4.cpp", Form4);
USEFORM("Unit5.cpp", Form5);
USEFORM("Unit6.cpp", Form6);
USEFORM("Unit7.cpp", Form7);
//-----
WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)
{
    try
    {
        Application->Initialize();
        Application->CreateForm(__classid(TForm1), &Form1);
        Application->CreateForm(__classid(TDM), &DM);
        Application->CreateForm(__classid(TFGRAFIK), &FGRAFIK);
        Application->CreateForm(__classid(TFSOISKATEL), &FSOISKATEL);
        Application->CreateForm(__classid(TFVID), &FVID);
        Application->CreateForm(__classid(TFVAKANSII), &FVAKANSII);
        Application->CreateForm(__classid(TFTEST), &FTEST);
        Application->CreateForm(__classid(TFSOTRUDNIKI), &FSOTRUDNIKI);
        Application->CreateForm(__classid(TFNAIM), &FNAIM);
        Application->CreateForm(__classid(TFPROHOZH), &FPROHOZH);
        Application->CreateForm(__classid(TFRABOTODATEL), &FRABOTODATEL);
        Application->CreateForm(__classid(TFRECOMENDACIYA),
&FRECOMENDACIYA);
        Application->CreateForm(__classid(TRepGrafik), &RepGrafik);
        Application->CreateForm(__classid(TForm2), &Form2);
        Application->CreateForm(__classid(TForm3), &Form3);
        Application->CreateForm(__classid(TForm4), &Form4);
        Application->CreateForm(__classid(TForm5), &Form5);
        Application->CreateForm(__classid(TForm6), &Form6);
        Application->CreateForm(__classid(TForm7), &Form7);
        Application->Run();
    }
    catch (Exception &exception)
    {
        Application->ShowException(&exception);
    }
    catch (...)
    {
        try
        {
            throw Exception("");
        }
        catch (Exception &exception)
        {
            Application->ShowException(&exception);
        }
    }
}

```

```
    }  
  }  
  return 0;  
}  
//-----
```