# Аннотация

Данный дипломный проект содержит 86 листов текста диплома, 18 рисунков, 4 таблицы, 5 приложений, 39 источников литературы.

ИНФОРМАЦИОННАЯ СИСТЕМА, РАБОТА С ДАННЫМИ, ИНТЕФЕЙС, ТРЕБОВАНИЯ, МОДЕЛЬ ЖИЗНЕННОГО ЦИКЛА, МОДЕЛИРОВАНИЕ, ПРОЕКТИРОВАНИЕ, РЕАЛИЗАЦИЯ, ЛОГИЧЕСКАЯ МОДЕЛЬ СТРУКТУРЫ ДАННЫХ, ФИЗИЧЕСКАЯ МОДЕЛЬ СТРУКТУРЫ ДАННЫХ, БАЗА ДАННЫХ, ТЕСТИРОВАНИЕ, ФИЗИЧЕСКОЕ ПРЕДСТАВЛЕНИЕ СИСТЕМЫ, ЛОГИЧЕСКОЕ ПРЕДСТАВЛЕНИЕ СИСТЕМЫ.

Темой дипломного проекта является «Информационная система учета призывников в администрации (на примере администрации с. Казинка)».

Источниками данных для данного дипломного проекта являлись книги, периодические издания, электронные ресурсы, используемые как теоретическая основа для рассматриваемой проблемы. Так же литературные источники использовались в качестве практических пособий при реализации проекта.

В проекте выполнен первоначальный процесс постановки на воинский учет в администрации, осуществлен сбор требований к информационной системе. Реализованная часть информационной системы внедрена в отделе воинского учета администрации с. Казинка.

СОДЕРЖАНИЕ

[Введение](#_Toc169061910)

[1. Разработка требований к программному обеспечению](#_Toc169061911)

[1.1 Анализ существующих решений по автоматизации предметной области](#_Toc169061912)

[1.3 Сбор требований](#_Toc169061914)

[1.4 Спецификация требований](#_Toc169061915)

[1.5 Аттестация требований](#_Toc169061916)

[1.6 Выбор методологии проектирования информационной системы](#_Toc169061917)

[2. Проектирование информационной системы](#_Toc169061919)

[2.1 Архитектурное проектирование](#_Toc169061920)

[2.2 Проектирование интерфейса информационной системы](#_Toc169061921)

[2.3 Проектирование баз данных](#_Toc169061922)

[2.4 Обоснование выбора платформы создания информационной системы](#_Toc169061923)

[2.5 Проектирование модулей (объектно-ориентированные модели)](#_Toc169061924)

[3. Реализация и аттестация информационной системы](#_Toc169061925)

[3.1 Реализация приложения](#_Toc169061926)

[3.2 Взаимодействие приложения с источниками данных](#_Toc169061927)

[3.3 Тестирование приложения](#_Toc169061928)

[3.4 Методика развертывания приложения](#_Toc169061929)

[4. Управление информационным проектом](#_Toc169061931)

[4.1 Выбор жизненного цикла разработки ПО](#_Toc169061932)

[4.2 Определение цели и области действия программного проекта](#_Toc169061933)

[4.3 Создание структуры пооперационного перечня работ](#_Toc169061934)

[4.4 Идентификация задач и действий](#_Toc169061935)

[4.5 Оценка размера и возможности повторного использования](#_Toc169061936)

[4.6 Оценка длительности и стоимости разработки проекта](#_Toc169061937)

[4.7 Распределение ресурсов проекта](#_Toc169061938)

[4.8 Оценка экономической эффективности проекта](#_Toc169061939)

[ЗАКЛЮЧЕНИЕ](#_Toc169061941)

[Список литературы](#_Toc169061942)

[ПРИЛОЖЕНИЕ А - Спецификация требований к программному обеспечению](#_Toc169061943)

[ПРИЛОЖЕНИЕ Б – Прототиты пользовательского интерфейса](#_Toc169061944)

[ПРИЛОЖЕНИЕ В – Атрибуты управляющих таблиц проектируемой ИС](#_Toc169061945)

[ПРИЛОЖЕНИЕ Г - Выбор модели жизненного цикла](#_Toc169061946)

[ПРИЛОЖЕНИЕ Д – Диаграмма Гантта](#_Toc169061947)

ВВЕДЕНИЕ

Информационные технологии все больше и больше затрагивает сферы деятельности человека. И сейчас под натиском информационных и телекоммуникационным технологиям необходимо введение информационных систем в те области где они не применяются или слабо развиты и которые помогут уменьшить затраты, время на обработку данных, и увеличить производительность труда.

Рассматриваемая информационная система в дипломной работе написана на базе и по заказу администрации муниципального образования Казинского сельсовета.

Далее описывается значимость ИС для данной сферы деятельности. Основные решаемые задачи.

Целью настоящего дипломного проекта является разработка и внедрение информационной системы отдела воинского учета администрации с. Казинка.

Программа должна обеспечивать:

* работу с входными данными (полная информация о лице подлежащему призыву на обязательную воинскую службу);
* получение выходных документов (структурированная информация содержащая вcе необходимые сведения о военнообязанных лицах);
* формирование отчетов (получение данных на бумажных носителях об отдельном лице, списке лиц).

Хранение всей информации в электронных базах данных, что позволит структурировать информацию, появится возможность быстрого поиска необходимых данных.

Для достижения вышеуказанной цели необходимо решить следующие задачи:

* провести анализ бизнес-процессов воинского учета;
* исследовать информационные потоки, возникающие в системе;
* разработать концептуальную и логическую модели данных;
* разработать программное обеспечение для АРМ воинского учета;
* провести оценку экономической эффективности информационной системы.

В данном дипломном проекте будет рассмотрено одно из административных подразделений отдел воинского учета населения. Этот отдел существует довольно долго. В нем хранится вся информация о лицах подлежащих воинскому учету и призванных на обязательную воинскую службу, лицах находящихся в запасе, непригодные к службе, а так же иная информация которая необходима для полного и достоверного ведения учета. Использование бумажных носителей, папок для распределения военнообязанных по категориям, предполагает затрату большого количества времени и сил при поиске по запросам, а так же требует значительного специально оборудованного места для хранения бумажных носителей, и папок с данной информацией. Поэтому необходима информационная система автоматизации отдела воинского учета, создать базу данных, где будет храниться вся информация о лицах подлежащих призыву, а так же иная необходимая информация связанная с воинской обязанностью граждан или снятых с учета по истечении определенного количества лет, или иная причина. Разработать удобные для работы формы, приятный интерфейс. Всё это позволит сократить время поиска информации о лицах подлежащих призыву, позволит хранить информацию о военнообязанных в базе данных, при необходимости которую можно распечатать, что значительно повысит эффективность работы отдела воинского учета и структурировать информацию в более удобный вид.

Все разрозненные сведения сводятся и хранятся в одном месте и программа для отдела воинского учета выдает только комплексные и выверенные данные, исключая их несогласованность, а значит, недостоверность.

Благодаря внедрению программы для отдела воинского учета возникает более четкое разделение функций, при котором специалист вводит в систему определенный вид записей достоверных записей. Таким образом, информационная система автоматизации учета военнообязанных сводит информацию с нескольких бумажных носителей в систему в одну базу данных, обобщает ее, и в итоге программа для отдела воинского учета структурирует разрозненную информацию и позволяет увидеть ее в удобном виде. Для выполнения данной задачи была изучена внешняя и внутренняя структура администрации, собрана необходимая информация для построения диаграммы функций организации и диаграммы потоков данных, рассмотрены основные процессы.

Программа для отдела воинского учета значительно ускорит работу администрации по учету военнообязанных лиц, а значит, повысит качество работы.

1. РАЗРАБОТКА ТРЕБОВАНИЙ К ПРОГРАММНОМУ ОБЕСПЕЧЕНИЮ

## Анализ существующих решений по автоматизации предметной области

При внедрении информационной системы перед администрацией стоит альтернатива выбора между программными продуктами, предлагаемыми на рынке информационных технологий или разработка собственной программы. При анализе существующих информационных систем необходимо учитывать особенности организации ее деятельности, а также рассматривать ряд факторов (например, простота использования и внедрения, стоимость, реализация выполняемых функций и т.д.).

Из существующих информационных систем можно выделить программную разработку "1С:Предприятие 8.0 Управление персоналом " компании 1С. предназначена для реализации кадровой политики компании по следующим направлениям: планирование потребностей в персонале, обеспечение бизнеса кадрами, кадровый учет, ведение регламентированного документооборота [1].

Возможность изменения и дополнения первоначальной (типовой) конфигурации программы позволяет настроить ее на требования любого предприятия и даже конкретного пользователя. В то же время первоначальная поставка позволяет сразу приступить к реализации процедур управления персоналом. Одной из основных функциональных возможностей программы является возможность использования конфигурации Воинский учет, которая включает в себя следующее:

* Ведение воинского учета в конфигурации организовано в соответствии со следующими нормативными документами:
* Федеральный закон от 26 февраля 1997 года № 31-ФЗ "О мобилизационной подготовке и мобилизации в Российской Федерации";
* Постановление Правительства РФ от 25 декабря 1998 года № 1541 "Об утверждении положения о воинском учете".

В конфигурацию включены следующие отчеты, которые представляются организациями по запросам военкоматов либо на регулярной основе:

* Данные для постановки на учет в военкомате;
* Численность граждан запаса (формы №6 и 11/МУ);
* Список граждан, подлежащих постановке на воинский учет;
* Список принятых и уволенных военнообязанных;
* Список юношей 15-16 лет;
* Список для первоначальной постановки юношей на воинский учет;
* Список принятых и уволенных призывников

Недостатками данной системы является:

* относительно высокая стоимость покупки лицензии на использование, а так же содержание системы после истечения срока бесплатного сервисного обслуживания данного продукта
* необходимость наличия специалиста для настройки системы под конкретные процессы отдела воинского учета.
* наличие функций, в которых нет необходимости использования.

К еще одной информационной системе на которую стоит обратить внимание, можно отнести программный продукт ООО Научно-производственного центра «КОСМОС-2», Автоматизированная информационная система (АИС) Сельского Административного образования (САО) сокращенно (АИС САО) [2]. Система служит для автоматизации хозяйственной деятельности, похозяйственной книги, операций, выполняемых работниками муниципальных сельских администраций:

* ведение похозяйственного учета (в объеме книги похозяйственного учета)
* учет наличного населения (как постоянного так и временно пребывающих), учет льготных категорий граждан, учет движения постоянно проживающих (прибытие - убытие), оформление различных списков населения,
* учет земельных участков и объектов недвижимости на территории, прав жителей на эти объекты,
* учет многоквартирных домов и квартир на территории сельского округа (с проживающими по квартирам жителями),
* подготовка данных для заполнения форм статистической отчетности, делопроизводство в Администрации,
* оформление и печать справок, выписок, постановлений,
* регистрации пребывания граждан с подготовкой пакета документов,
* воинский учет с подготовкой пакета документов (повестки, акты, анкета).

Система имеет многоуровневую структуру, объединяя 9 Автоматизированных Рабочих Мест (АРМ), в число которых входит АРМ «Данные о жителях»

АРМ «Данные о жителях» позволяет реализовать следующие функции на основании реестра населения, формируемого системой:

* Подготовка и печать произвольных списков жителей. Как частный случай – подготовка списка призывников.
* Получать данные о собственности как отдельных персоналий, так и всей семьи (т.е. для членов хозяйства как для сельских подворий, так и для подворий МКД).
* Подготовка списков юбиляров.
* Учет и оформление документов по регистрации граждан.

В том числе:

* Учет и оформление документов по воинскому учету.

К недостатками данной системы можно отнести:

* Относительно высокая стоимость продукта;
* Наличие специалиста для конфигурирования системы.

К последней информационной системе, которая будет рассмотрена в данном дипломном проекте, относится программный продукт ЗАО ИВЦ ИНСОФТ «Система Персонального Учета Населения» (СПУН). Система Персонального Учета Населения создана с целью формирования информационной базы для реализации процедур регистрации в рамках решаемых различными ведомствами задач Российской Федерации. В рамках данной системы имеется конфигурация для регистрация граждан, имеющих статус военнообязанных. Система обладает необходимым набором средств для учета и оформление документов по воинскому учету[3].

Недостатками данной информационной системы является:

* Высокая стоимость системы;
* Наличие высокоскоростного подключения в Интернет для режима постоянного доступа (on-line).

## Анализ предметной области

Анализ предметной области необходимо проводить для:

* Определения функциональной направленности предприятия;
* Определения перечня структурных звеньев предприятия.

Эти действия позволять более точно определить область деятельности предприятия, которые позволяют успешно решать возникшие проблемы.

Организационная структура сельской администрации представлена на рис. 1.1:

Рисунок 1.1 – Организационная структура администрации

Ниже перечислены функции подразделений администрации:

* бухгалтерия – начисление заработной платы, расчет платежами между организациями, ведение отчетности по налогам и сборам.
* отдел воинского учета – проведение на местах военно–организационных работ.
* земельный отдел – учет земельных участков, паев, организация налогового сбора, выдача справок о землевладении.
* отдел общего назначения – паспортный режим, прописка, выписка и т.д.

Поскольку задачей дипломного проектирования являлась разработка информационной системы автоматизации отдела воинского учета, а не всей администрации, то подробному анализу проведем только для процессов относящиеся к учету лиц подлежащих призыву.

Основными функциональными задачами отдела воинского учета являются:

* Постановка граждан на первоначальный воинский учет;
* Снятие с учета по причине смены места жительства;
* Снятие с учета по достижении возраста 50 лет;
* Ведение реестра граждан, подлежащих призыву;

Моделирование бизнес процессов отдела воинского учета проведено с использованием CASE-средства RanionalRose [5,6]. Разработанная диаграмма бизнес-вариантов использования представлена на рис. 1.2. Она отображает перечисленные выше функциональные задачи отдела и показывает основные бизнес-актеров данного процесса.

Рисунок 1.2 – Диаграмма бизнес-вариантов использования отдела воинского учета.

После завершения изучения предметной области следует перейти к процессу определения требований.

##

## Сбор требований

Сбор требований – это процесс, включающий мероприятия, необходимые для создания и утверждения документа, содержащего спецификацию системных требований [6].

В данном проекте для сбора требований была выбрана методика «Интервьюирование» которая рассматривает следующие этапы:

1. Разрабатываются вопросы
2. Производится выбор опрашиваемых пользователей.
3. Планируются контакты.
4. Проводится интервью.
5. Завершается встреча.
6. Определяются последующие действия.

А так же определены функциональные, системные требования и требования к интерфейсу системы:

В процессе формирования требований принимали участие следующие лица:

* Глава администрации;
* Инспектор ВУС (Военно-учетного стола отдела воинского учета);
* Специалист первой категории администрации.

На рисунке 1.3 представлена диаграмма вариантов использования ИС [5], представляющая процессы, происходящие в ИС отдела воинского учета.

Рисунок 1.3 – Диаграмма вариантов использования ИС отдела воинского учета.

## Спецификация требований

Определение корректных требований — ответственный этап программного проекта. Формат проекта должен соответствовать требованиям к ПО, собранным командой разработчиков или одним разработчиком, в противном случае эти требования не смогут быть поддержаны и представлены в программном продукте. Спецификация требований к ПО - Software Requirements Specification(SRS) имеет ключевое значение для всего жизненного цикла разработки программного продукта. Это не только производный документ, в котором определены спецификации программного проекта, но и основной документ, применяемый с целью проведения аттестационных и приемочных испытаний. Аттестация — это оценка качества работы менеджеров проекта. Она определяет степень соответствия программного продукта установленным требованиям. Спецификация SRS выступает в роли некоего механизма фиксирования системных требований, которые используются в качестве критериев при аттестации [7].

На основании SRS достигается согласие между заказчиками и производителями программного продукта. В спецификации SRS полностью описаны функции, которые должен выполнять разрабатываемый программный продукт. Позволяющая потенциальным пользователям определить степень соответствия продукта их потребностям, а также пути модификации продукта для того, чтобы он был максимально полезен в решении их задач.

Снижаются временные затраты на разработку. В подготовке спецификации SRS задействованы различные лица в организации заказчика. Они тщательно исследуют все требования еще до того, как начнется непосредственная разработка проекта. Это снижает вероятность последующей повторной разработки проекта, кодирования и тестирования.

При тщательном изучении требований, представленных в спецификации SRS, можно обнаружить недосмотры, недоразумения и противоречия еще на ранних этапах цикла разработки, когда проблемы устранять гораздо легче, чем на более поздних этапах.

Спецификация SRS становится основой для оценки стоимости и составления графика работ. Описание продукта — это реальный процесс необходимый для оценки стоимости проекта. В среде, где существует понятие формального предложения, SRS используют для утверждения оценки предложения или цены.

С помощью правильно составленных спецификаций SRS на уровне организации могут разрабатывать намного более продуктивные планы аттестаций и проверок. Являясь частью договора на разработку, SRS обеспечивает точку отсчета для оценки соответствия техническим условиям.

Благодаря спецификации SRS облегчается передача программного продукта новым пользователям, а также его установка на других компьютерах. Таким образом, заказчикам становится проще переносить программный продукт в другие подразделения организации, а разработчикам — передавать другим заказчикам.

В SRS документе рассматривается сам продукт, а не процесс разработки проекта, поэтому на ее основании можно производить расширение завершенного продукта.

Спецификация требований к реализуемому программному обеспечению представлена в ПРИЛОЖЕНИИ А.

После того как процесс определения и спецификации требований завершён, необходимо осуществить аттестацию требований.

## Аттестация требований

Аттестация должна продемонстрировать, что требования действительно определяют ту систему, которую хочет иметь заказчик. Проверка требований важна, так как ошибки в спецификации требований могут привести к переделке системы и большим затратам, если будут обнаружены во время разработки системы или после введения её в эксплуатацию [8-10].

Для аттестации требований будет использован метод прототипирования. В этом случае конечным пользователям и заказчику демонстрируется некий прототип системы.

Прототип основного меню данного модуля представлен на рисунке 1.4.

Рисунок 1.4 – Схема основного меню программы

Интерфейс этого модуля должен обеспечить следующие возможности пользователя:

* Создание новой записи о гражданине.
* Загрузки данных из базы данных «Призывник» и действия с данными;
* Загрузки данных из базы данных «Запасник» и действия с данными;
* Загрузки данных из базы данных «Снятые с учета» и действия с данными;
* Загрузки данных из базы данных «Служащие в армии» и действия с данными;
* Загрузки данных из базы данных «Непригодные к службе» и действия с данными.

Прототипы диалоговых окон программы представлены в Приложении Б.

## Выбор методологии проектирования информационной системы

Существует две базовых методологии проектирования информационных систем: структурный и объектно-ориентированный анализ.

Сущность структурного подхода к разработке информационной системы заключается в ее декомпозиции на автоматизируемые функции: система разбивается на функциональные подсистемы, которые в свою очередь делятся на подфункции, подразделяемые на задачи и так далее. Процесс разбиения продолжается вплоть до конкретных процедур. При этом автоматизируемая система сохраняет целостное представление, в котором все составляющие компоненты взаимоувязаны. При разработке системы «снизу-вверх» от отдельных задач ко всей системе целостность теряется, возникают проблемы при информационной стыковке отдельных компонентов [11,12].

Использование объектно-ориентированных методов позволяет создать описание (модель) предметной области в виде совокупности объектов – сущностей, объединяющих данные и методы обработки этих данных (процедуры). Каждый объект обладает своим собственным поведением и моделирует некоторый объект реального мира. С этой точки зрения объект является вполне осязаемой вещью, которая демонстрирует определенное поведение.

В объектном подходе акцент переносится на конкретные характеристики физической или абстрактной системы, являющейся предметом программного моделирования. Объекты обладают целостностью, которая не может быть нарушена. Таким образом, свойства, характеризующие объект и его поведение, остаются неизменными. Объект может только менять состояние, управляться или становиться в определенное отношение к другим объектам [13,14].

Наиболее подходящей методологией проектирования ИС отдела воинского учета является объектно-ориентированная.

## Выводы

В данном разделе проведен анализ требований по автоматизации предметной области, собраны необходимые требования к проектируемой информационной системе, пользовательского интерфейса, выполняемых функций. Выбрана объектно-ориентированная технология проектирования информационной системы.

2. ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ

## 2.1 Архитектурное проектирование

Архитектура представляет собой концептуальное видение структуры будущих функциональных процессов и технологий на системном уровне и во взаимосвязи системы. Сложные информационные системы организаций проектируются как композиция компонентов, взаимодействующих на высоком уровне, которые сами могут быть системами. Архитектура информационной системы организации делает понимание системы легче, определяя ее функциональность и структуру. Архитектура информационной системы организации представляет собой модель того, как информационная технология будет поддерживать основные цели и стратегию развития автоматизируемого объекта. Архитектура информационной системы описывает, как информационные системы, приложения и люди работают в пределах всей организации в единообразной объединенной манере.

При проектировании современных информационных систем организаций их архитектура должна разрабатываться с учетом многих заинтересованных сторон, она должна быть понятной пользователям, дать возможность разработчикам сделать план и графики системы, позволять определять ключевые интерфейсы, функции и технологии, а также позволять оценить график и бюджет исполнения проекта. При этом от архитекторов современных информационных систем требуется ответственность за создание удовлетворительной и осуществимой концепции системы на самом раннем этапе ее разработки, поддержки цельности этой концепции на протяжении разработки и определения пригодности результирующей системы для использования клиентом. Разработка архитектуры информационной системы – это процесс описания архитектур информационных систем в достаточной деталировке, чтобы сделать их более полезными для разработки информационных систем.

Архитектуры информационной системы требуется соблюдение следующих условий:

* соответствие с миссией организации;
* определенность в требованиях;
* направленность в разработке;
* возможность к адаптации;
* необходимость гибкости.

Соблюдение всех этих условий позволяет разработать архитектуру информационной системы организации более совершенной и эффективной [15,17].

Программные архитектуры, используемые в настоящее время:

* файл-серверная;
* клиент-серверная;
* многоуровневая.

Организация информационных систем на основе использования выделенных файл-серверов все еще является наиболее распространенной в связи с наличием большого количества персональных компьютеров разного уровня развитости и сравнительной дешевизны связывания PC в локальные сети. Такая организация привлекательна тем, что при опоре на файл-серверные архитектуры сохраняется автономность прикладного (и большей части системного) программного обеспечения, работающего на каждом компьютере сети. Фактически, компоненты информационной системы, выполняемые на разных компьютерах, взаимодействуют только за счет наличия общего хранилища файлов, которое располагается на файл-сервере. В классическом случае в каждом компьютере дублируются не только прикладные программы, но и средства управления базами данных. Файл-сервер представляет собой разделяемое всеми компьютерами комплекса расширение дисковой памяти. Возможно два варианта работы системы в файл-серверной архитектуре:

1. Многопользовательская работа с локальными таблицами на файл-сервере. В этом случае система Контур Стандарт и файл OLAP-приложения размещаются на клиентском компьютере, а таблицы исходных данных на файл-сервере. Это удобно, когда одни и те же исходные данные используются для выполнения различных видов анализа разными пользователями. Например, учетные данные, выгруженные из OLTP-системы в dbf-файл, анализируют бухгалтер и руководитель, причем каждый из них работает со своим аналитическим приложением.
2. Многопользовательская работа с OLAP-приложениями и исходными данными в виде локальных таблиц на файл-сервере. Тогда на компьютере пользователя устанавливается только Контур Стандарт с OLAP-машиной. Такая архитектура обеспечивает работу группы пользователей (например, бухгалтерии) с одним и тем же аналитическим приложением. Доступ пользователей группы к файлу OLAP-приложения регламентируется средствами операционной системы [4].

Архитектура "клиент-сервер" сегодня является доминирующей концепцией в создании распределенных сетевых приложений и предусматривает взаимодействие и обмен данными между ними. Она предусматривает такие основные компоненты:

* набор серверов, предоставляющих информацию или другие услуги программам, которые обращаются к ним;
* набор клиентов, использующих сервисы, которые предоставляются серверами;
* сеть, которая обеспечивает взаимодействие между клиентами и серверами

На рисунке 2.1 представлена схема клиент-серверной архитектуры.

Рисунок 2.1 – клиент-серверная архитектура.

Серверы являются независимыми друг от друга. Клиенты также функционируют параллельно и независимо друг от друга. Отсутствует жесткая привязка клиентов к серверам. Более чем типичной является ситуация, если один сервер одновременно обрабатывает запросы от разных клиентов; с другой стороны, клиент может обращаться то к одному серверу, то к другому. Клиенты должны знать о доступных серверах, но могут не иметь представления о существовании других клиентов. Общепринятым является соглашение, что клиенты и серверы - это, прежде всего программные модули. Чаще всего они находятся на разных компьютерах, но бывают ситуации, если обе программы - и клиентская, и серверная, физически размещаются на одной машине; в такой ситуации сервер часто называется локальным. Модель клиент-серверного взаимодействия определяется, прежде всего, распределением обязанностей между клиентом и сервером. Логически можно выделить три различные операции:

* уровень представления данных, который, по сути, является интерфейсом пользователя и отвечает за представление данных пользователю и введение от него управляющих команд;
* прикладной уровень, который реализует основную логику приложения и на котором осуществляется необходимая обработка информации;
* уровень управления данными, которые обеспечивает сохранение данных и доступ к ним.

Двухуровневая клиент-серверная архитектура предусматривает взаимодействие двух программных модулей - клиентского и серверного. В зависимости от того, как между ними распределяются приведенные выше функции, различают:

* модель тонкого клиента, в рамках которой вся логика приложения и управление данными сосредоточена на сервере. Клиентская программа обеспечивает только функции уровня представления;
* модель толстого клиента, в которой сервер только управляет данными, а обработка информации и интерфейс пользователя сосредоточены на стороне клиента.

Архитектура приложения, разделяющая пользовательские и прикладные сервисы и сервисы данных. Другое название - трехуровневая архитектура, однако термин «многоуровневая» корректнее, поскольку он предполагает, что при логическом проектировании может возникнуть более трех уровней сервисов. МНОГОУРОВНЕВАЯ АРХИТЕКТУРА ПРИЛОЖЕНИЯ (multi-tiered architecture), способ организации взаимодействия программ или компонентов программы. Как правило, Многоуровневая архитектура приложения используется в распределенных приложениях, компонеты которых выполняются на разных компьютерах. Частным случаем Многоуровневая архитектура приложения является архитектура клиент-сервер. В последнее время в информационных системах получила распространение архитектура, в которой распределенное приложение состоит из компонентов трех уровней:

* компонент, ответственный за управление данными, выполняется на сервере баз данных;
* компонент, выполняющий обработку данных, выполняется на сервере приложений;
* компонент, реализующий интерфейс с пользователем, исполняется на рабочей станции [4].

В данном проекте выбрана клиент-серверная архитектура, т.к. информационная система будет использовать одну базу данных на нескольких рабочих станциях. Сеть модели “клиент-сервер” уменьшает потребность Компьютеров-клиентов в оперативной памяти, поскольку вся работа с файлами выполняется на сервере. Серверы в клиент-серверных системах способны хранить большое количество данных. Благодаря этому на компьютерах-клиентах освобождается значительный объем дискового пространства для других приложений. Наконец, управление всей системой, включая контроль за ее безопасностью, становится намного проще, так как все файлы и данные централизованно размещаются на сервере или на небольшом числе серверов. Упрощается также резервное копирование.

Диаграмма компонентов позволяет определить архитектуру разрабатываемой системы, установив зависимости между программными компонентами.

Диаграмма компонентов разрабатывается для следующих целей:

* визуализации общей структуры исходного кода программной системы;
* спецификации исполнимого варианта программной системы;
* обеспечения многократного использования отдельных фрагментов программного кода;
* представления концептуальной и физической схем баз данных.

Диаграмма компонентов системы представлена на рисунке 2.2.

Рисунок 2.2 – диаграмма компонентов информационной системы.

Физическое представление программной системы не может быть полным, если отсутствует информация о том, на какой платформе и на каких вычислительных средствах она реализована. Второй формой физического представления является диаграмма развёртывания. Она применяется для представления общей конфигурации и топологии распределённой программной системы и содержит распределение компонентов по отдельным узлам системы. Диаграмма развёртывания предназначена для визуализации элементов и компонентов программы, существующих лишь на этапе её исполнения. При этом представляются только компоненты-экземпляры программы, являющиеся исполнимыми файлами или динамическими библиотеками. Те компоненты, которые не используются на этапе исполнения, на диаграмме развёртывания не показываются.

Диаграмма размещения представлена ниже на рисунке 2.3.

Рисунок 2.3 – диаграмма размещения информационной системы.

## 2.2 Проектирование интерфейса информационной системы

Разработка пользовательского интерфейса включает те же основные этапы, что и разработка программного обеспечения:

1. Определение типа интерфейса и общих требований к нему.
2. Определение сценариев использования.
3. Определение пользовательской модели интерфейса.
4. Программирование и тестирование программных интерфейсов.

Первым этапом в разработке пользовательского интерфейса является прототипирование, которое выполняется во время сбора требований к системе. При необходимости для каждого отдельного процесса системы создается частичный прототип включающий: экранную форму, диалог или отчет. Затем определяются требования разграничения доступа к данным.

После детального рассмотрения процессов определяется количество функциональных элементов разрабатываемой системы. Это позволяет разделить информационную систему на ряд подсистем, каждая из которых реализуется одной командой разработчиков за приемлемое для RAD-проектов время (порядка полутора месяцев).

Пользователи часто судят о качестве системы в целом, исходя из качества ее интерфейса. Более того, от качества интерфейса зависит эффективность использования системы.

Пользовательский интерфейс представляет собой совокупность программных и аппаратных средств, обеспечивающих взаимодействие пользователя с компьютером. Основу такого взаимодействия составляют диалоги. Под диалогом в данном случае понимают регламентированный обмен информацией между человеком и компьютером, осуществляемый в реальном масштабе времени и направленный на совместное решение конкретной задачи: обмен информацией и координация действий. Каждый диалог состоит из отдельных процессов ввода-вывода, которые физически обеспечивают связь пользователя и компьютера.

Пользовательский интерфейс проектируемой информационной системы имеет следующую структуру: после запуска приложения открывается главная форма, которая содержит основное меню, состоящее из 3-ти пунктов меню: Файл, Данные, Помощь. Прототип пользовательского интерфейса ПРИЛОЖЕНИИ Б.

## 2.3 Проектирование баз данных

Основными целями проектирования базы данных являются:

* представление данных и связей между ними, необходимых для всех основных областей применения данного приложения и любых существующих групп его пользователей;
* создание модели данных, способной поддерживать выполнение любых требуемых транзакций обработки данных;
* разработка предварительного варианта проекта, структура которого позволяет удовлетворить все основные требования, предъявляемые к производительности системы — например, ко времени реакции системы [16].

В основу проектирования БД должны быть положены представления конечных пользователей конкретной организации – концептуальные требования к системе. Именно конечный пользователь в своей работе принимает решения с учетом получаемой в результате доступа к базе данных информации. От оперативности и качества этой информации будет зависеть эффективность работы организации. Данные, помещаемые в базу данных, также предоставляет конечный пользователь.

При рассмотрении требований конечных пользователей необходимо принимать во внимание следующее:

База данных должна удовлетворять актуальным информационным потребностям организации. Получаемая информация должна по структуре и содержанию соответствовать решаемым задачам [14,16].

При проектировании базы данных создаются два уровня модели – логический и физический. Логический уровень – это абстрактный взгляд на данные, на нем данные представляются так, как выглядят в реальном мире, и могут называться так, как они называются в реальном мире. Объекты модели, представляемые на логическом уровне, называются сущностями. Логический уровень модели данных может быть построен на основе другой модели, например, концептуальной модели данных (смотри рисунок 2.4). Логический уровень модели данных является универсальным и никак не связан с конкретной реализацией системы управления базами данных.

Реализация логической модели начинается с определения концептуальной модели, определяющей основные сущности, сохраняемые в виде таблиц реляционной базы данных. ПРИЛОЖЕНИЕ В.

На рисунке 2.4 пример концептуальной модели, на базе анализа сущностей

Рисунок 2.4 – Концептуальная модель данных ИС.

Доработка этой концептуальной модели с учетом атрибутов таблиц позволяет перейти непосредственно к логической модели БД.

На рисунке 2.5 пример логической модели, на базе анализа сущностей

Рисунок 2.5 – Логическая модель данных ИС.

Физический уровень модели данных, напротив, зависит от конкретной системы управления базами данных, фактически являясь отображением системного каталога. В физическом уровне модели содержится информация о всех объектах базы данных. Поскольку стандартов на объекты базы данных не существует (например, нет стандарта на типы данных), физический уровень модели зависит от конкретной реализации системы управления базами данных. Следовательно, одному и тому же логическому уровню модели могут соответствовать несколько разных физических уровней различных моделей. Если на логическом уровне модели не имеет большого значения, какой конкретно тип данных у атрибута (хотя и поддерживаются абстрактные типы данных), то на физическом уровне модели важно описать всю информацию о конкретных физических объектах – таблицах, колонках, индексах, процедурах и т. д.

Физической СУБД для ИС отдела воинского учета выбрана СУБД Microsoft SQL Server 2005 . Этот выбор осуществлен потому, что приложение будет функционировать на нескольких рабочих станциях используя базу данных одновременно по локальной сети. Также следует отметить, что СУБД MS SQL Server положительно зарекомендовала себя в процессе эксплуатации.

СУБД MS SQL Server компании Microsoft начала разрабатываться в 1988 и изначально предназначалась для платформы OS/2 [19,20]. Последующие версии этого сервера баз данных предназначались для платформы Windows NT и со временем были тесно интегрированы с этой операционной системой. Для других платформ версии этого сервера не выпускаются и не выпускались.

Удобный пользовательский интерфейс утилит администрирования в сочетании с достаточно высокой производительностью и относительно невысокой стоимостью эксплуатации сделал эту серверную СУБД второй по популярности - после Oracle.

Клиентские приложения для Microsoft SQL Server и MSDE можно создавать как с помощью средств разработки Microsoft - Visual Basic, Visual C++, С#, Access и Visual FoxPro, так и с помощью средств разработки других производителей. Для этой цели имеются ODBC-драйвер и OLE DB-провайдер, а также содержащий их набор библиотек Microsoft Data Access Components (MDAC), позволяющий использовать в средствах разработки объекты ActiveX Data Objects (ADO) - COM-объекты для доступа к данным. MDAC является составной частью Windows XP, а для пользователей других Windows-платформ доступен отдельно на Web-сайте Microsoft.

В отличие от Oracle, Microsoft не производит средств разработки, использующих тот же самый язык программирования, что и язык для создания кода триггеров и хранимых процедур, однако производит средства отладки серверного кода (например, SQL Server Debugger входит в состав Visual Basic и Visual C++).

В настоящее время наиболее широко используемой является версия MS SQL Server 2005. В состав Microsoft SQL Server 2005 входят простые утилиты администрирования (Enterprise Manager), сервисы преобразования данных (Data Transformation Services), облегчающие перенос данных в SQL Server из других типов СУБД, поддержка распределенных запросов и транзакций, OLAP-сервер и утилиты для создания хранилищ данных (в том числе данных из других серверных СУБД.

* масштабируемость и надежность. SQL Server 2005 обеспечивает практически неограниченный рост объемов данных за счет увеличения надежности и масштабируемости системы, используя все преимущества мультипроцессорной обработки данных. SQL Server 2005 Enterprise Edition обеспечивает параллельность обработки данных
* высокая скорость построения решений. SQL Server 2005 уменьшает время создания, развертывания и выхода на рынок современных приложений для задач бизнеса, электронной коммерции, использует встроенный отладчик T-SQL. Совершенствует и ускоряет процесс поиска данных, упрощает управление, позволяет использовать создаваемые пользователем функции в других приложениях [22,26].

На рисунке 2.6 представлена физическая модель данных.

Рисунок 2.6 – Схема физической модели данных для ИС отдела воинского учета

## 2.4 Обоснование выбора платформы создания информационной системы

В настоящее время обязательной возможностью считается визуальное проектирование, когда программист строит свои приложения, используя готовые модули. Примером могут служить все современные пакеты для разработчиков – Borland Delphi, ,Microsoft Visual Studio 2005 и т.д.

Чтобы средства разработки и технологии отвечали требованиям разработчиков, в корпорации Майкрософт была создана совершенно новая модель программирования для доступа к данным, основанная на .NET Framework. Построение на основе .NET Framework гарантирует единообразие доступа к данным: компоненты используют систему общих типов, общие шаблоны разработки и соглашения о пространствах имен [24,25].

В .NET Framework поддерживается прямая и обратная совместимость. В контексте .NET Framework обратная совместимость означает, что любое приложение, созданное в .NET Framework более ранней версии, будет выполняться и в более поздней версии[29]. Прямая совместимость означает возможность выполнения приложения, созданного в более поздней версии .NET Framework SDK v 2.0, в .NET Framework более ранней версии.

Классы ADO.NET были разработаны для поддержки возможностей новой модели программирования: интеграции с XML, единого представления данных с возможностью комбинирования данных из различных источников, а также средств оптимизации взаимодействия с базой данных, представленных в .NET Framework [29].

Структура ADO.NET создана для решения задач современной модели разработки приложений. В то же время модель программирования по возможности приближена к ADO, что упрощает переход разработчиков ADO к новой среде. ADO.NET является неотъемлемой частью .NET Framework, оставаясь понятной программистам ADO [28].

.NET представляет собой совершенно новый способ создания распределенных настольных и встроенных приложений. Для типов .NET не нужны ни фабрики классов, ни поддержка IUnknown, ни регистрация в системном реестре. Эти основные элементы СОМ не скрыты – их просто больше нет.

Специально для новой платформы Microsoft разработала новый язык программирования – С#, который впитал в себя многое из того лучшего, что есть в самых разных языках программирования, и так же является составной частью Microsoft Visual Studio 2005.

Платформа .NET является полностью независимой от используемых языков программирования. Можно использовать несколько .NET-совместимых языков программирования даже в рамках одного проекта.

Основные возможности .NET следующие:

* Полные возможности взаимодействия с существующим кодом;
* Полное и абсолютное межъязыковое взаимодействие, межъязыковая обработка исключений и межъязыковая отладка;
* Общая среда выполнения для любых приложений .NET, вне зависимости от того, на каких языках они были созданы. Один из важных моментов при этом – то, что для всех языков используется один и тот же набор встроенных типов данных;
* Библиотека базовых классов, которая обеспечивает сокрытие всех сложностей, связанных с непосредственным использованием вызовов API, предлагает целостную объектную модель для всех языков программирования, поддерживающих .NET;
* Отсутствует сложность СОМ;
* Действительное упрощение процесса развертывания приложения.

В .NET нет необходимости регистрировать двойные типы в системном реестре. .NET позволяет разным версиям одного и того же модуля DLL мирно сосуществовать на одном компьютере.

Microsoft Visual Studio 2005 продолжает поддерживать технологии Microsoft .NET Framework уже в версии Microsoft .NET Framework SDK v2.0, которые предоставляют общеязыковую среду выполнения и унифицированные классы программирования. Также в Visual Studio включена библиотека MSDN, содержащая документацию по данным инструментам разработки.

Платформа Microsoft.NET для отображения данных на компьютере конечного пользователя и его интерактивного взаимодействия с системой. предоставляет класс System.Windows.Forms.Form и большое разнообразие классов элементов управления, дочерних от класса Control. Функциональность уровня представления во многом определяется составом элементов управления, входящих в коллекцию Controls для конкретной формы [29].

Уровень бизнес-логики отражает логику предметной области и реализует основные функции информационной системы. К таким функциям относятся вычисления на основе вводимых и хранимых данных, проверка элементов данных и обработка команд, поступающих от слоя представления, а также передача информации слою источника данных. Возможности, предоставляемые технологией Microsoft.NET, позволяют достаточно эффективно решать вопросы корректности ввода пользователем данных и поэтому часть функций проверки элементов данных может быть решена на уровне представления.

Уровень бизнес-логики получает на вход информацию от уровня представления, проводит необходимые проверки и вычисления, сохраняет в информацию базе данных и возвращает уровню представления определенные данные.

Бизнес-логика описывается набором методов, реализующих бизнес-транзакцию. Для платформы Microsoft.NET это типовое решение сценарий транзакций использует прямой доступ к базе данных и базируется на использовании объектов классов DataCommand и DataReader технологии ADO.NET, а так же используя bindingSource, TableAdapter, DataSet . Класс, реализующий сценарий транзакций, обеспечивает прямой доступ к источнику данных и необходимую функциональность бизнес-логики. Для данного типового решения все обязанности по реализации бизнес-логики возлагаются на методы сценария транзакций.

Для разрабатываемой информационной системы выбрана платформа Microsoft Visual Studio 2005. В качестве языка реализации приложения выбран C# [23,26].

## 2.5 Проектирование модулей (объектно-ориентированные модели)

Процесс проектирования начинается с того, что модель анализа и выбранная архитектура принимается в качестве основной входной информации. Далее, в процессе проектирования используются нефункциональные требования к системе и ограничения налагаемые на архитектуру, в результате чего модель анализа трансформируется в новую форму – модель проектирования, которая затем может быть напрямую реализована в виде программного кода. Проектирование ИС предполагает решение следующих вопросов:

 Выбор архитектуры и определение средств дальнейшей физической реализации полученной в конце модели проектирования.

 Уточнение модели анализа путём построения диаграмм взаимодействий и детализации диаграммы классов [33-35]. Внесение необходимых изменений и поправок в имеющуюся модель анализа, если необходимо.

 Построение диаграммы состояний системы. Внесение необходимых изменений и поправок в имеющуюся модель анализа, если необходимо.

Определение с учётом ограничений налагаемые на архитектуру компонентов проектируемой системы, построение диаграммы компонентов.

Модель проектирования представляется в виде UML-диаграмм, схем, рисунков и описаний.

Рисунок 2.7 - Диаграммы Состояний (Statechart) Информационной системы

Рисунок 2.8 - Диаграммы Компонентов (Component) Информационной системы

Выводы к разделу

Во втором разделе рассмотрены архитектурное проектирование информационной системы, определяется пользовательский интерфейс системы. Проводится проектирование баз данных. Выбирается база данных которая будет удовлетворять требованиям разрабатываемой информационной системы. Определяются таблицы и тип хранимых данных в них. Определяется структура базы данных. Выбирается платформа для создания информационной системы. Для разрабатываемой информационной системы была выбрана платформа Microsoft Visual Studio 2005. В качестве языка реализации приложения выбран C#.

3. РЕАЛИЗАЦИЯ И АТТЕСТАЦИЯ ИНФОРМАЦИОННОЙ СИСТЕМЫ

## 3.1 Реализация приложения

Итогом реализации приложения является работоспособная информационная система. Разрабатываемая информационная система будет являться приложением клиент серверного типа. Т.к. информационная система будет взаимодействовать с серверной базой данных. Т.о. информационная система будет состоять из двух частей. Первая часть приложения реализующая интерфейс пользователя и находящаяся на клиентской рабочей станции. А вторая часть приложения будет отвечать за хранение и обработка данных осуществляется на стороне сервера.

Для реализации взаимодействия клиента и сервера необходимо реализовать функции загрузки и отображения данных из базы данных, и пересылка данных в базу данных с последующим сохранением данных в базе. Для работы с данными, а так же с базами данных, используется следующее пространство имен

using System.Data

using System.Data.SqlClient;

using System.Data.OleDb;

входящий в состав Microsoft .NET Framework SDK v2.0. В данном проекте использовалось следующее пространственное имя для поключения к базе данных:

using System.Data

Загрузку данных из базы данных осуществляет следующая функция:

this.prizivnikTableAdapter.Fill(this.dataSet1.Prizivnik).

Отображение данных из базы данных на форме информационной системы реализуется через функцию, часть кода которой представлен ниже:

private void toolStripButton2\_Click(object sender, EventArgs e)

 {

 foreach(DataRow row in dataSet1.Prizivnik)

 {

listBox1.Items.Add(row["Familia"].ToString() + " " + row["Imya"].ToString() + " " + row["Othestvo"].ToString());

 }

 }

Для изменения информации отображаемой на форме, т.е. переключение между данными осуществляется через функцию, часть кода которой представлен ниже:

private void listBox1\_SelectedIndexChanged(object sender, EventArgs e)

 {

 int pos = -1;

 pos = listBox1.SelectedIndex;

 textBox1.Text = (dataSet2.Grazdanin[pos].LastName);

 textBox2.Text = (dataSet2.Grazdanin[pos].Name);

 textBox3.Text = (dataSet2.Grazdanin[pos].SecondName);

 textBox9.Text = (dataSet2.Grazdanin[pos].DataRozd.ToString());

 textBox4.Text = (dataSet2.Grazdanin[pos].Rozd\_Oblast);

 textBox5.Text = (dataSet2.Grazdanin[pos].Rozd\_Raion);

 textBox6.Text = (dataSet2.Grazdanin[pos].Rozd\_Gorod);

 textBox10.Text = (dataSet2.Passport\_Data[pos].Pas\_Ser\_1pole);

 textBox11.Text = (dataSet2.Passport\_Data[pos].Pas\_Ser\_2pole);

 textBox12.Text = (dataSet2.Passport\_Data[pos].Nomer);

 textBox13.Text = (dataSet2.Passport\_Data[pos].Vidan);

 textBox16.Text = (dataSet2.Passport\_Data[pos].DataVidathi);

 textBox17.Text = (dataSet2.Passport\_Data[pos].Kod\_1pole);

 textBox18.Text = (dataSet2.Passport\_Data[pos].Kod\_2pole);

 }

Здесь представлен код отображения данных на форме из таблиц «Гражданин» и «Паспортные данные». За пересылку новых данных в базу, которые были введены на форме, отвечает функция, краткий код которой представлен ниже:

private void toolStripButton7\_Click(object sender, EventArgs e)

 {

DataRow NewPrizivnik = dataSet2.Grazdanin.NewRow();

 NewPrizivnik["SecondName"] = textBox1.Text;

 NewPrizivnik["Name"] = textBox2.Text;

 NewPrizivnik["LastName"] = textBox3.Text;

 NewPrizivnik["CreateDay"] = dateTimePicker1.Text;

 NewPrizivnik["Rozd\_Oblast"] = textBox4.Text;

 NewPrizivnik["Rozd\_Raion"] = textBox5.Text;

 NewPrizivnik["Rozd\_Gorod"] = textBox6.Text;

 NewPrizivnik["DataRozd"] = textBox9.Text;

 NewPrizivnik["Foto"] = pictureBox1.Image;

 dataSet2.Grazdanin.Rows.Add(NewPrizivnik);

}

Код показывает, как происходит запись только в одну таблицу «Гражданин», но помимо таблицы «Гражданин» существует еще ряд таблиц, в которые происходит запись новых данных, но они здесь не рассматриваются.

После передачи данных в базу данных их необходимо сохранить, за процедуру сохранения отвечает функция:

this.prizivnikTableAdapter.Update(dataSet1.Prizivnik);

В случаях когда данные не нужны в базе данных, либо эти данные являются недоставерными их необходимо удалить, за удаление данных из базы отвечает следующая функция :

private void toolStripButton4\_Click(object sender, EventArgs e)

 {

 int pos = -1;

 pos = listBox1.SelectedIndex;

 dataSet2.Grazdanin.Rows[pos].Delete();

 dataSet2.Passport\_Data.Rows[pos].Delete();

 dataSet2.Prizivnik.Rows[pos].Delete();

 dataSet2.Negoden.Rows[pos].Delete();

 dataSet2.Med\_Card.Rows[pos].Delete();

 dataSet2.In\_Army.Rows[pos].Delete();

 dataSet2.Factual\_Plase\_Residence.Rows[pos].Delete();

 dataSet2.Get\_Out.Rows[pos].Delete();

 dataSet2.Officcer.Rows[pos].Delete();

 dataSet2.Sniat\_50.Rows[pos].Delete();

 dataSet2.Soldat.Rows[pos].Delete();

 dataSet2.Zapasnik.Rows[pos].Delete();

 }


##

## 3.2 Взаимодействие приложения с источниками данных

Взаимодействие приложения с источником данных осуществляется при помощи запросов языка SQL. SQL (Structured Query Language) является инструментом для выборки и обработки информации, содержащейся в базе данных. SQL является языком программирования, который применяется для организации взаимодействия пользователя с базой данных [21]. Если пользователю необходимо получить информацию из базы данных, он запрашивает её у СУБД при помощи SQL. СУБД обрабатывает запрос, находит требуемые данные и посылает их пользователю [16]. Процесс запрашивания данных и получения результата называется запросом к базе данных. SQL используется для реализации всех функциональных возможностей, которые СУБД предоставляют пользователю:

* организация данных;
* выборка данных;
* обработка данных;
* совместное использование данных;
* управление доступом;
* целостность данных.
* Так же можно использовать хранимые процедуры, обладающие следующими преимуществами перед прямыми запросами:
* последовательная, безопасная модификация данных;
* уменьшение сетевого трафика;
* поддержка автоматического выполнения при запуске системы;
* перестройка и повторное использование схемы выполнения;
* автоматическая настройка параметра запроса;
* обеспечение модульной структуры приложения;
* совместное использование в нескольких приложениях;
* авторизованный и единообразный доступ к объектам базы данных;

## 3.3 Тестирование приложения

Полностью избежать ошибок при программировании невозможно, даже профессионалы время от времени допускают ошибки, а так же недоработки системы проявляющиеся во время тестирования, а иногда и эксплуатации.

Тестирование – это проверка работы программ с данными, подобным реальным, которые будут обрабатываться в процессе эксплуатации системы. Процесс тестирования программного обеспечения осуществляется на основе фактических или смоделированных входных данных (как стандартных, так и не стандартных) при определённых контролируемых условиях.

Если программный продукт не соответствует сформулированным требованиям, идентифицируются значительные отличия между ожидаемыми и фактическими результатами.

На разных этапах процесса разработки программного обеспечения применяют различные виды тестирования:

* тестирование дефектов обусловленных ошибками в программе(синтаксические ошибки, ошибки периода выполнения, логические ошибки);
* статическое тестирование оценивает производительность и надёжность программ, а также работу системы в различных режимах эксплуатации.

Неформальное тестирование осуществляется разработчиками для того, чтобы оценить выполняемый процесс разработки программного обеспечения. В данном случае термин «неформальный» вовсе не означает, что тестированию не придаётся серьёзного значения. Это подразумевает, что заказчик официально не принимает участия в процессе тестирования, а основная цель - обнаружение различного рода ошибок.

Тестирование приложения проводилось с помощью стандартных инструментов предоставляемых Microsoft Visual Studio 2005.

Режим пошагового исполнения кода позволяет построчно анализировать программу для диагностики и исправления ошибок. Visual Studio 2005 предоставляет несколько вариантов пошагового исполнения:

* Step Into позволяет построчно просматривать код с заходом в вызываемые функции;
* Step Over позволяет построчно просматривать код без захода в вызываемые функции;
* Step Out исполняет текущую функцию до конца и останавливается (если возможно) на следующей строке функции, из которой была вызвана текущая процедура;
* Run To Cursor позволяет установить курсор в некоторую строку и исполнить весь код до этой строки;
* Set Next Statement (назначить следующий оператор) позволяет назначить следующий оператор для исполнения, при этом все строки до этого оператора будут пропущены.

Точки прерывания — это строки кода, назначенные во время отладки, по достижении которых исполнение останавливается, а приложение переходит в режим пошагового исполнения. Для точек прерывания можно назначать дополнительные условия, определяющие обстоятельства, при которых эти точки активируются. Для управления точками прерывания предназначено окно Breakpoints,позволяющее создавать, отключать и удалять их [25,26,29].

Visual Studio 2005 предоставляет ряд инструментов для наблюдения за исполнением программы. Окна Locals, Autos и Watch позволяют отслеживать значения переменных программы во время ее исполнения; окно Command позволяет исполнять код и получать значения переменных и свойств. Имеются также дополнительные окна для наблюдения за самыми разными данными приложения [26].

Тестирование и отладка — это два различных и в тоже время взаимосвязанных мероприятия. Под отладкой понимают непосредственный поиск ошибок в коде и их исправление, а тестированием называют процесс, позволяющий выявить эти ошибки. Обычно тестированию подвергают каждый метод приложения, заставляя его обработать всевозможные параметры в разных условиях. Такой подход называют блочным тестированием, поскольку при этом выполняется тестирование отдельных компонентов приложения. Однако приложения, как правило, слишком сложны, чтобы проверить все возможные параметры и условия исполнения [29].

Успех тестирования целиком определяется качественным выбором контрольных примеров. Если их слишком мало, тестирование не даст результата и в окончательной версии программы непременно окажется много ошибок, а если их чрезвычайно много, вы потеряете время, деньги и превысите сроки, отведенные на разработку. Сбалансированным планом тестирования приложения считается тот, где достаточно полно представлены функции приложения и проверяются наиболее типичные сценарии его использования.

Проверка функциональности программы при обработке всех вариантов однотипных данных — хорошая отправная точка для составления плана тестирования. Но только тестирование обработки разных типов данных позволяет убедиться в надежности программы. Приложение должно не только нормально работать и генерировать ожидаемые результаты на основе аргументов, на работу с которыми оно рассчитано, но и корректно обрабатывать значения, которые выходят за пределы допустимого диапазона. Тестирование считается законченным не раньше, чем будет установлено, что оно корректно обрабатывает различные данные, в том числе и значения, которые больше или меньше допустимых. Далее описаны приемы составления тестовых данных, которые использовались при создании контрольных примеров для тестирования приложения.

* Проверка типичных значений аргументов;
* Проверка обработки минимальных и максимальных значений аргументов;
* Использование заведомо недопустимых аргументов;
* Комбинированные примеры.

## 3.4 Методика развертывания приложения

Чтобы приложение оценили по достоинству, оно прежде всего должно оказаться на компьютерах своей целевой аудитории. Microsoft Visual Studio 2005 поддерживает самые разные способы развертывания проектов: от простейшего с использованием команды XCOPY до самого сложного и гибкого с применением программы Microsoft Windows Installer.

Простые приложения .NET Framework позволяет развертывать, просто копируя их каталоги на клиентские компьютеры. Для развертывания более сложных приложений в Visual Studio 2005 предусмотрена технология Windows Installer, позволяющая создавать проекты установочных программ с широкими возможностями настройки [24,26,34].

Но для того чтобы приложение работало на компьютере клиента должна быть установлена требуемая версия общеязыковой среды исполнения Frameworks, в данном случае под данную информационную систему необходимо установить Microsoft .NET Framework SDK v2.0. В данном наборе библиотек классов есть все классы которые потребуются разработанной информационной системе. Для развертывания приложения были использованы инструменты с использованием команды XCOPY и установки на компьютер клиентам Microsoft .NET Framework SDK v2.0 [26].

## Выводы к разделу

В данном разделе дипломного проекта рассмотрены реализация информационной системы. Описаны способы нахождения ошибок в приложении, так же описаны методики тестирования, которые использовались при тестировании приложения. Описана методика развертывания данной информационной системы, с указанием версии общеязыковой среды исполнения Frameworks необходимой для работы данной системы. Рассмотрена методика взаимодействия информационной системы с СУБД MS SQL Server 2005.

4. УПРАВЛЕНИЕ ИНФОРМАЦИОННЫМ ПРОЕКТОМ

## 4.1 Выбор жизненного цикла разработки ПО

Жизненный цикл – совокупность стадий и этапов, которые проходит информационная система в своём развитии от момента принятия решения о создании системы до момента прекращения её функционирования [35, 39].

Процесс создания программного обеспечения – это множество взаимосвязанных процессов и результатов их выполнения, которые ведут к созданию программного продукта. Несмотря на то, что наблюдается огромное многообразие подходов, методов и технологий создания программного обеспечения, существуют фундаментальные базовые процессы, без реализации которых не может обойтись ни одна технология разработки программных продуктов: разработка спецификации ПО; проектирование и реализация ПО; аттестация ПО; эволюция ПО.

Выбор приемлемой модели жизненного цикла разработки программного обеспечения для проекта может осуществляться в ходе использования процесса создания.

Данный процесс заключается в том, что необходимо проанализировать следующие отличительные категории проекта: требования, команда разработчиков, коллектив пользователей, тип проекта и риски. Далее, следует ответить на вопросы по каждой категории и проанализировать полученные данные. На основе этого результата будет определена наиболее приемлемая модель модели жизненного цикла информационной системы.

Таблицы с вопросами, ответы на которые будут определять оптимальную модель жизненного цикла для информационной системы приведено в ПРИЛОЖЕНИИ Г.

В таблице 4.1 представлены итоговые результаты выбора модели жизненного цикла.

Таблица 4.1 – Определение оптимальной модели жизненного цикла в баллах.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Характеристика | Каскадная | V-образная | Прототипирование | Спиральная | RAD | Инкрементная |
| Требования  | 5 | 4 | 2 | 2 | 4 | 2 |
| Участники команды разработчиков | 6 | 5 | 4 | 5 | 7 | 6 |
| Коллектив пользователей | 4 | 5 | 0 | 3 | 3 | 2 |
| Типы проектов и рисков | 8 | 6 | 6 | 4 | 7 | 6 |
| Итого  | 23 | 20 | 12 | 14 | 21 | 16 |

По результатам, приведенным в таблице, наиболее подходящая модель жизненного цикла информационной системы для данного небольшого проекта является модель RAD [15].

Модель жизненного цикла информационной системы есть некоторая структура, определяющая последовательность осуществления процессов, действий и задач, выполняемых на протяжении жизненного цикла информационной системы, а также взаимосвязи между этими процессами, действиями и задачами.

Методология разработки информационных систем, основанная на использовании средств быстрой разработки приложений, получила в последнее время широкое распространение и приобрела название методологии быстрой разработки приложений — RAD (Rapid Application Development). Данная методология охватывает все этапы жизненного цикла современных информационных систем.

RAD — это комплекс специальных инструментальных средств быстрой разработки прикладных информационных систем, позволяющих оперировать с определенным набором графических объектов, функционально отображающих отдельные информационные компоненты приложений.

В последнее время широкое распространение методология быстрой разработки приложений RAD (Rapid Application Development). Под этим термином обычно понимается процесс разработки ПО, содержащий 3 элемента:

* небольшую команду программистов (от 2 до 10 человек);
* короткий, но тщательно проработанный производственный график (от 2 до 6 мес.);
* повторяющийся цикл, при котором разработчики, по мере того, как приложение начинает обретать форму, запрашивают и реализуют в продукте требования, полученные через взаимодействие с заказчиком.

Команда разработчиков должна представлять из себя группу профессионалов, имеющих опыт в анализе, проектировании, генерации кода и тестировании ПО с использованием CASE-средств. Члены коллектива должны также уметь трансформировать в рабочие прототипы предложения конечных пользователей.

Жизненный цикл ПО по методологии RAD состоит из четырех фаз:

* фаза анализа и планирования требований;
* фаза проектирования;
* фаза построения;
* фаза внедрения.

На фазе анализа и планирования требований пользователи системы определяют функции, которые она должна выполнять, выделяют наиболее приоритетные из них, требующие проработки в первую очередь, описывают информационные потребности. Определение требований выполняется в основном силами пользователей под руководством специалистов-разработчиков. Ограничивается масштаб проекта, определяются временные рамки для каждой из последующих фаз. Кроме того, определяется сама возможность реализации данного проекта в установленных рамках финансирования, на данных аппаратных средствах и т.п. Результатом данной фазы должны быть список и приоритетность функций будущей ИС, предварительные функциональные и информационные модели ИС.

На фазе проектирования часть пользователей принимает участие в техническом проектировании системы под руководством специалистов-разработчиков. CASE-средства используются для быстрого получения работающих прототипов приложений. Пользователи, непосредственно взаимодействуя с ними, уточняют и дополняют требования к системе, которые не были выявлены на предыдущей фазе. Более подробно рассматриваются процессы системы. Анализируется и, при необходимости, корректируется функциональная модель. Каждый процесс рассматривается детально. При необходимости для каждого элементарного процесса создается частичный прототип: экран, диалог, отчет, устраняющий неясности или неоднозначности. Определяются требования разграничения доступа к данным. На этой же фазе происходит определение набора необходимой документации.

После детального определения состава процессов оценивается количество функциональных элементов разрабатываемой системы и принимается решение о разделении ИС на подсистемы, поддающиеся реализации одной командой разработчиков за приемлемое для RAD-проектов время - порядка 60 - 90 дней. С использованием CASE-средств проект распределяется между различными командами (делится функциональная модель). Результатом данной фазы должны быть:

* общая информационная модель системы;
* функциональные модели системы в целом и подсистем, реализуемых отдельными командами разработчиков;
* точно определенные с помощью CASE-средства интерфейсы между автономно разрабатываемыми подсистемами;
* построенные прототипы экранов, отчетов, диалогов.

Все модели и прототипы должны быть получены с применением тех CASE-средств, которые будут использоваться в дальнейшем при построении системы. Данное требование вызвано тем, что в традиционном подходе при передаче информации о проекте с этапа на этап может произойти фактически неконтролируемое искажение данных. Применение единой среды хранения информации о проекте позволяет избежать этой опасности.

В отличие от традиционного подхода, при котором использовались специфические средства прототипирования, не предназначенные для построения реальных приложений, а прототипы выбрасывались после того, как выполняли задачу устранения неясностей в проекте, в подходе RAD каждый прототип развивается в часть будущей системы. Таким образом, на следующую фазу передается более полная и полезная информация.

На фазе построения выполняется непосредственно сама быстрая разработка приложения. На данной фазе разработчики производят итеративное построение реальной системы на основе полученных в предыдущей фазе моделей, а также требований нефункционального характера. Программный код частично формируется при помощи автоматических генераторов, получающих информацию непосредственно из репозитория CASE-средств. Конечные пользователи на этой фазе оценивают получаемые результаты и вносят коррективы, если в процессе разработки система перестает удовлетворять определенным ранее требованиям. Тестирование системы осуществляется непосредственно в процессе разработки.

После окончания работ каждой отдельной команды разработчиков производится постепенная интеграция данной части системы с остальными, формируется полный программный код, выполняется тестирование совместной работы данной части приложения с остальными, а затем тестирование системы в целом. Завершается физическое проектирование системы:

* определяется необходимость распределения данных;
* производится анализ использования данных;
* производится физическое проектирование базы данных;
* определяются требования к аппаратным ресурсам;
* определяются способы увеличения производительности;
* завершается разработка документации проекта.

Результатом фазы является готовая система, удовлетворяющая всем согласованным требованиям.

На фазе внедрения производится обучение пользователей, организационные изменения и параллельно с внедрением новой системы осуществляется работа с существующей системой (до полного внедрения новой). Так как фаза построения достаточно непродолжительна, планирование и подготовка к внедрению должны начинаться заранее, как правило, на этапе проектирования системы. Приведенная схема разработки ИС не является абсолютной. Возможны различные варианты, зависящие, например, от начальных условий, в которых ведется разработка: разрабатывается совершенно новая система; уже было проведено обследование предприятия и существует модель его деятельности; на предприятии уже существует некоторая ИС, которая может быть использована в качестве начального прототипа или должна быть интегрирована с разрабатываемой.

Следует, однако, отметить, что методология RAD, как и любая другая, не может претендовать на универсальность, она хороша в первую очередь для относительно небольших проектов, разрабатываемых для конкретного заказчика. Если же разрабатывается типовая система, которая не является законченным продуктом, а представляет собой комплекс типовых компонент, централизованно сопровождаемых, адаптируемых к программно-техническим платформам, СУБД, средствам телекоммуникации, организационно-экономическим особенностям объектов внедрения и интегрируемых с существующими разработками, на первый план выступают такие показатели проекта, как управляемость и качество, которые могут войти в противоречие с простотой и скоростью разработки. Для таких проектов необходимы высокий уровень планирования и жесткая дисциплина проектирования, строгое следование заранее разработанным протоколам и интерфейсам, что снижает скорость разработки.

Методология RAD неприменима для построения сложных расчетных программ, операционных систем или программ управления космическими кораблями, т.е. программ, требующих написания большого объема (сотни тысяч строк) уникального кода.

Оценка размера приложений производится на основе так называемых функциональных элементов (экраны, сообщения, отчеты, файлы и т.п.) Подобная метрика не зависит от языка программирования, на котором ведется разработка. Размер приложения, которое может быть выполнено по методологии RAD, для хорошо отлаженной среды разработки ИС с максимальным повторным использованием программных компонентов, метод определения которых представлен в 4.2.

Таблица 4.2 – определение размера приложения по методологии RAD

|  |  |
| --- | --- |
| < 1000 функциональных элементов | один человек |
| 1000-4000 функциональных элементов | одна команда разработчиков |
| > 4000 функциональных элементов | 4000 функциональных элементов на одну команду разработчиков |

В качестве итога перечислим основные принципы методологии RAD:

* разработка приложений итерациями;
* необязательность полного завершения работ на каждом из этапов жизненного цикла;
* обязательное вовлечение пользователей в процесс разработки ИС;
* необходимое применение CASE-средств, обеспечивающих целостность проекта;
* применение средств управления конфигурацией, облегчающих внесение изменений в проект и сопровождение готовой системы;
* необходимое использование генераторов кода;
* использование прототипирования, позволяющее полнее выяснить и удовлетворить потребности конечного пользователя;
* тестирование и развитие проекта, осуществляемые одновременно с разработкой;
* ведение разработки немногочисленной хорошо управляемой командой профессионалов;
* грамотное руководство разработкой системы, четкое планирование и контроль выполнения работ [39].

## 4.2 Определение цели и области действия программного проекта

Целью программного проекта является разработать и внедрить информационную систему первоначального учета граждан в поселении администрации для отдела воинского учета. Разработанная информационная система будет представлена в виде взаимосвязанных модулей, которые обеспечивают эффективный ввод, обработку и передачу информации с сохранением в серверной базе данных.

К модулям относятся:

* Индивидуальная карточка – которая содержит необходимые сведения для постановки гражданина на первоначальный учет;
* Отчеты, создаваемые по введенным критериям;

Область действия данной информационной системы не выходит за рамки отдела Воинского учета администрации села. Информационная система не будет применяться в операционных, отличных от Windows.

## 4.3 Создание структуры пооперационного перечня работ

Процесс создания Информационной системы для отдела Воинского учета представляется в виде перечня работ, реализованном в прикладном пакете Microsoft Office Project 2003 и включающий следующие этапы:

* анализ требований;
* изучение бизнес-цели проекта;
* постановка задач;
* создание технического задания;
* определение спецификаций проекта;
* оценка стоимости проекта;
* проектирование;
* определение логических объектов;
* построение схем взаимодействия объектов;
* определение функций системы;
* определение вариантов использования;
* проектирование уровня данных;
* проектирование уровня бизнес-логики;
* проектирование пользовательского интерфейса
* реализация;
* разработка уровня данных;
* разработка уровня бизнес-логики;
* разработка пользовательского интерфейса;
* интеграция системы;
* тестирование;
* отладка.

Разработка руководства по использованию системы, краткая характеристика и требования системы;

## 4.4 Идентификация задач и действий

Все работы можно классифицировать по своим характеристикам: длительности, трудозатратам и количеству людских ресурсов. Данные параметры связаны друг с другом: трудозатраты задачи равны произведению длительности на количество людских ресурсов. Задачи в плане проекта могут быть трех типов: с фиксированными длительностью, трудозатратами и количеством ресурсов. Для того чтобы реализовать ту или иную задачу необходимы ресурсы. Рассмотрим подобно все задачи и ресурсы с помощью которых реализуются эти задачи.

Таблица 4.3 – Ресурсы проекта

|  |  |
| --- | --- |
| Задачи | Ресурсы |
| Анализ требований | Разработчик  |
| Создание черновой версии спецификации проекта | Разработчик |
| Создание предварительного бюджета | Разработчик |
| Обсуждение спецификаций программного продукта | Руководитель проекта |
| Разработка графика сдачи | Разработчик |
| Получение разрешений на продолжение | Разработчик |
| Разработка общей информационной модели системы | Разработчик |
| Разработка функциональных моделей системы в целом, а также подсистем | Разработчик |
| Точное определение интерфейсов | Разработчик |
| Построение прототипов экранных форм, диалогов, отчетов | Разработчик |
| Определение параметров модульной и уровневой архитектуры | Разработчик |
| Назначение персонала для разработки | Руководитель проекта |
| Разработка кода | Разработчик |
| Разработка планов тестирования модулей с использованием спецификации продукта | Инженер-испытатель |
| Тестирование модулей | Инженер-испытатель |
| Тестирование интеграции | Инженер-испытатель |
| Тестирование интеграции модулей | Инженер-испытатель |
| Выявление ошибок и недоработок | Инженер-испытатель |
| Изменение кода | Разработчик |
| Повторное тестирование измененного кода | Инженер-испытатель |

## 4.5 Оценка размера и возможности повторного использования

Для оценки размера описываемого приложения был применен метод оценки количества строк кода LOC(Lines of Code). Данный метод заключается в подсчете созданных классов, а так же определенных методах в классах приложения. В данной разрабатываемой системе количество созданных классов будет равняться 10, а количество методов приходящихся на каждый класс будет рано в среднем 12. Так же необходимо определить какое количество строк кода(LOC) приходится на один метод. В зависимости от функций выполняемых определенным методом, количество строк кода будет колебаться от 10 – до 30. В общем же количество строк кода разрабатываемой системы равно 1700. Данные цифры являются усредненными, т.к. невозможно подсчитать точную цифру на этапе проектирования или разработки информационной системы.

Код данной информационной системы или отдельные классы могут повторно использоваться как в рамках данного приложения, так и в других приложениях. При необходимости можно провести модификацию кода или отдельных классов информационной системы, например классы взаимодействующие с базой данных. В зависимости от модификаций будет меняться и размер разрабатываемой информационной системы [24,25].

## 4.6 Оценка длительности и стоимости разработки проекта

Проект – это уникальный процесс, в ходе выполнения которого получают уникальный продукт. Управление проектом – это процесс планирования, организации и контроль состояния задач и ресурсов проекта, направленный на своевременное достижение цели проекта [7].

В ходе управления проектом должно быть обеспечено решение следующих задач:

* соблюдение директивных сроков завершения проекта;
* рациональное распределение материальных ресурсов и исполнителей между задачами проекта, а также во времени;
* своевременная коррекция исходного плана в соответствии с реальным положением дел.

Для автоматизации управление проекта использовался пакет Microsoft Project 2003. Автоматизированное средство Microsoft Project является широко распространённой системой управления проектами. К тому же данная система проста в применении. Проект состоит из серии взаимосвязанных задач, составляющих основу плана работ. Задача должна представлять определённую часть работы с ясной постановкой, но в тоже время должна быть достаточно короткой, чтобы иметь возможность регулярно отслеживать её исполнение и идентифицировать проблемы на ранних этапах [7].

Управление проектами в пакете Microsoft Project 2003 первым шагом подразумевает построение календарного графика работ. Календарный график строится на основе диаграммы Гантта. Диаграмма Гантта – это линейный график, задающий сроки начала и окончания взаимосвязанных работ, с указанием ресурсов, используемых для их выполнения [7]. ПРИЛОЖЕНИЕ Д.

Диаграмма Гантта не единственный элемент управления проектом который формируется при создании проекта разрабатываемой системы, помимо диаграммы для формирования календарного графика используется структурное планирование. Основная цель структурного планирования заключается в описании состава и взаимосвязи технологических операций, которые требуется выполнить для реализации проекта. Результатом структурного планирования является сетевой график проекта. Фрагмент сетевого графика представлена ниже на рисунке 4.1.

Рисунок 4.1 – Фрагмент сетевого графика

Так же необходимо рассчитать и длительность данного проекта для создаваемой информационной системы. Для этого необходимо рассчитать разницу между датой окончания проекта (Finish Date) и датой начала проекта (Start Date), определенных в свойствах проекта и представленных на рисунке 4.2.

Рисунок 4.2 – Свойства проекта

Разрабатываемая ИС не направлена на непосредственное увеличение прибыли администрации, для которого она реализуется. Поэтому эффективность внедрения ИС может оцениваться лишь по критериям улучшением производительности труда в отделе воинского учета.

Для оценки эффективности воспользуемся методом экспертных оценок. Метод расчета в данном случае состоит из нескольких этапов:

1. Выделить цели работы системы.
2. Определить наборы показателей, характеризующих определенную цель.
3. Определить уровень достижения показателя.
4. Рассчитать степень достижения каждой цели по выдвинутым показателям.
5. Определить весовые коэффициенты целей.
6. Рассчитать общий показатель эффективности разрабатываемой информационной системы.

Степень достижения цели рассчитывается как средняя величина достижения частных показателей. Формула расчета имеет следующий вид:

, (4.1)

где u(gi) – степень достижения цели, баллы;

– значение показателя, баллы;

K – количество показателей.

Весовой коэффициент вычисляется по формуле:

, (4.2)

где  – весовой коэффициент, баллы;

Vi – оценка, баллы.

Расчет оценки ведется по формуле:

 (4.3)

где Vi – оценка, баллы;

Rmin – минимальное значение ранга, баллы;

Ri – сумма рангов, баллы.

Для расчета суммы рангов воспользуемся формулой:

, (4.4)

где Ri – сумма рангов, баллы;

ri – значение, выставленное экспертом, баллы;

n – количество экспертов.

При этом проверяется согласованность мнений экспертов путем расчета значения известного коэффициента q Кендала (конкордации), для оценок данных экспертами.

Общий показатель эффективности рассчитывается как:

, (4.5)

где Em – показатель эффективности, баллы;

wi – весовой коэффициент, баллы;

u(gi) – степень достижения цели, баллы.

Теперь, рассмотрев общие положения методики оценки информационной системы перейдем к расчету конкретного показателя эффективности работы ИС.

Для начала, определим цели и показатели работы системы, а так же укажем уровень достижения показателей при создании прототипа.

Все это сведем в таблицу (таблица ).

Таблица – Цели, показатели и уровень достижения работы ИС

|  |  |  |
| --- | --- | --- |
| Цель | Показатель | Уровень достижения, баллы |
| g1 – технический уровень | y11 – минимизация количества ошибок при выполнении лабораторных исследований  | 0,9 |
| y12 – автоматизированный ввод результатов ручных методов исследований | 1,0 |
| y13 – автоматизация получения заказов, выдачи результатов и отчетов | 0,85 |
| g2 – коммуникация | y21 – оперативность | 0,75 |
| y22 – удобство использования | 0,89 |
| g3 – социальные цели | y31 – улучшение условий труда | 0,80 |
| y32 – внедрение групповых форм работы | 0,75 |
| y33 – уменьшение времени выполнения исследований | 0,95 |
| g4 – получение отчетности | y41 – автоматическое получение отчетов | 0,87 |
| y42 – уменьшение объема рутинной работы персонала лаборатории | 1,0 |
| g5 – простота использования | y51 – легко понимаемый интерфейс пользователя | 0,91 |
| y52 – возможность поиска | 0,75 |
| y53 – возможность сохранения, извлечения и редактирования документов | 0,87 |

На основе формулы 4.1 рассчитаем степень достижения целей. Расчет:

.

.

.

.

.


## 4.7 Распределение ресурсов проекта

В MS Project 2003 под ресурсами подразумевается все то, что необходимо для выполнения работ по реализации проекта. Это могут быть:

* Люди;
* Техника;
* Механизмы;
* Различные расходные материалы.

На рисунке 4.3 представлен список ресурсов, необходимых для выполнения задач.

Рисунок 4.3 – Список ресурсов

Для эффективного управления проектом, необходимо назначать каждой задаче ресурсы, требуемые для ее исполнения. Процесс назначения ресурсов задачам проекта, а также связанное с ним редактирование предварительного варианта календарного графика называют ресурсным планированием проекта. Ресурсное планирование позволяет:

* оценить потребность в ресурсах;
* спланировать рациональное распределение потребности в ресурсах во времени;
* определить участки проекта, являющиеся критическими с точки зрения потребностей в ресурсах;
* контролировать расходование ресурсов при реализации проекта.

Ресурсное планирование проекта так же связанно и с анализом его временных параметров, так как время также может рассматриваться как специфический ресурс, избыточное количество которого способно компенсировать недостаток каких-либо других видов ресурсов. Назначение ресурсов задачам позволяет определить, какое количество ресурсов, необходимо для выполнения той или иной задачи. Объём назначений – это общее количество единиц конкретного ресурса, назначенных на выполнение данной задачи. Объём назначений может быть выражен как в абсолютных единицах, так и в процентах.

Пример назначения ресурса задачи можно видеть из рисунка 4.4.

Из рисунка видно, что для реализации задачи «Построение физической модели данных системы» назначен ресурс Разработчик.

Рисунок .4 – Ресурс для задачи «Построение физической модели данных системы»

При необходимости можно просмотреть таблицу использования ресурсов в проекте, отображающая объем работ, т.е. общее количество «трудового участия» ресурса, необходимое для выполнения каждой задачи проекта. [7].

На рисунке 4.5 Представлен пример таблицы использования ресурсов.

Рисунок .5 – Таблица использования ресурса «Инженер испытатель»

MS Project 2003 предоставляет возможность просмотра календаря ресурса. Календарь ресурса – это распределение рабочего и нерабочего времени для конкретного ресурса. Используя диалоговое окно «Календарь ресурса», менеджер может задавать различные параметры, отражающие характеристики Рабочего времени ресурсов. Так для каждого ресурса определяется тип календаря, количество рабочих и выходных дней, ежедневный график работы. На рисунке 4.6 представлен пример календаря для ресурса «Инженер испытатель».

Рисунок 4.6 – Календарь ресурса « Инженер испытатель»

## 4.8 Оценка экономической эффективности проекта

Разрабатываемая информационная система не несет в себе экономической выгоды. Поэтому экономические расчеты здесь не нужны. Целью создания информационной системы является информатизация процесса учета призывников в администрации с. Казинка, с реализацией последующих функций.

Программа должна обеспечивать:

* работу с входными данными (полная информация о лице подлежащему призыву на обязательную воинскую службу);
* получение выходных документов (структурированная информация содержащая вcе необходимые сведения о военнообязанных лицах);
* формирование отчетов (получение данных на бумажных носителях об отдельном лице, списке лиц).

Хранение всей информации в электронных базах данных, что позволит структурировать информацию, появится быстрый поиск необходимых данных.

Использование бумажных носителей, папок для распределения военнообязанных по категориям, предполагает затрату большого количества времени при поиске по запросам, а так же требует значительного специально оборудованного места для хранения бумажных носителей, и папок с данной информацией. Поэтому необходима информационная система автоматизации отдела воинского учета, создать базу данных, где будет храниться вся информация о лицах подлежащих призыву, разработать удобные для работы формы, приятный интерфейс. Всё это позволит сократить время поиска информации о лицах подлежащих призыву, позволит хранить информацию о военнообязанных в базе данных, при необходимости которую можно распечатать, что значительно повысит эффективность работы отдела воинского учета и структурировать информацию в более удобный вид.

Все разрозненные сведения сводятся и хранятся в одном месте и программа для отдела воинского учета выдает только комплексные и выверенные данные, исключая их несогласованность, а значит, недостоверность.

Благодаря внедрению программы для отдела воинского учета возникает более четкое разделение функций, при котором специалист вводит в систему определенный вид записей достоверных записей. Таким образом, информационная система автоматизации учета военнообязанных сводит информацию с нескольких бумажных носителей в систему в одну базу данных, обобщает ее, и в итоге программа для отдела воинского учета структурирует разрозненную информацию и позволяет увидеть ее в удобном виде. Для выполнения данной задачи была изучена внешняя и внутренняя структура администрации, собрана необходимая информация для построения диаграммы функций организации и диаграммы потоков данных, рассмотрены основные процессы.

Программа для отдела воинского учета значительно ускорит работу администрации по учету военнообязанных лиц, а значит, повысит качество работы.

## Выводы к разделу

В четвертом разделе дипломного проекта проводился выбор модели жизненного цикла информационной системы. И по результатам, наиболее подходящей моделью жизненного цикла информационной системы для данного небольшого проекта является модель RAD. Составлена структура пооперационного перечня работ и на её основе построен график выполнения работ, также определены ресурсы и распределены на каждую задачу, приведена диаграмма Гантта. Проведена оценка размера программного продукта и возможность повторного использования. Выбраны методы тестирования программного продукта. Проведен анализ экономической эффективности проекта.

ЗАКЛЮЧЕНИЕ

Целью дипломного проекта являлась разработка информационной системы для отдела воинского учета администрации с. Казинка.

Первым этапом дипломного проекта являлась определение цели и задач дипломного проекта. Был проведен анализ существующих систем.

В первом разделе выполнено бизнес моделирование бизнес-моделирование процессов организации. Построена диаграмма бизнес-вариантов использования отдела воинского учета и построена диаграмма вариантов использования информационной системы.

Проведен анализ требований, предъявляемых пользователями к информационной системе. В процессе формирования требований принимали участие следующие лица: Глава администрации, Инспектор ВУС (Военно-учетного стола отдела воинского учета), Специалист первой категории администрации. Результаты моделирования требований представлены в виде диаграмм прецедентов. Осуществлён процесс специфицирования требований. Итогом данного этапа стало выполнение аттестации требований посредством прототипирования.

Во втором разделе дипломного проекта выполнено проектирование информационной системы. На данном этапе были построены модели логического и физического представления системы.

Логическое представление системы включает в себя динамическую и статическую модели системы. Здесь значительное внимание было уделено проектированию изолированных классов системы, их иерархий и композиций. При построении объектной модели информационной системы был разработан объектный шаблон приложения, который послужил основой для проектирования и реализации системы отдела воинского учета. Классы были разделены на две группы: классы контролеры и классы взаимодействия с базой данных.

Физическое представление системы заключалось в построении диаграммы компонентов системы и диаграммы развертывания. Программа была спроектирована как трехзвенное приложение состоящее из: модулей Systems.Windows.Forms, сервера и сервера базы данных MS SQL Server 2005.

В ходе осуществления процесса проектирования выполнено моделирование структуры данных (логическая и физическая модели). Программное средство используемое для создания CASE-средства использовался программный продукт Rational Rose 2000 Enterprise Edition. Был рассмотрен использованный программный инструментарий. В качестве среды разработки программного обеспечения была использована Microsoft Visual Studio 2005 и язык программирования C#. Для управления контроля версиями программного продукта использовалось стандартное средство из свойств проекта Publish Version.

В третьем разделе дипломного проекта рассмотрена реализация программного продукта и вопросы связанные с реализацией. Реализованы функции и классы взаимодействия с базой данных. Приведены методы и свойства классов. Продемонстрирован фрагментарно исходный код. Так же рассмотрена и продемонстрирована методика взаимодействия приложения с СУБД MS SQL Server 2005. Проведено тестирование программного кода с использованием стандартных средств предоставляемых Microsoft Visual Studio 2005.

В процессе дипломного проектирования осуществлялась деятельность по управления проектом разработки информационной системы. Часть вопросов менеджмента информационного проекта были рассмотрены в четвертом разделе.

В четвертом разделе дипломного проекта определялась цель и область действия программного продукта. Осуществлен выбор модели жизненного цикла процесса разработки по результатам, представленным в таблице 4.1 – «Определение оптимальной модели жизненного цикла в баллах». Составлена структура пооперационного перечня работ с использованием пакета управления проектами Microsoft Project 2003, на её основе построен график выполнения работ, приведена диаграмма Гантта.

Описаны необходимые ресурсы и затраты проекта, планирование на этапах тестирования и развертывания приложения.

Разработанная информационная система внедрена в отделе воинского учета администрации с. Казинка. Справка об использовании программного продукта приведена в приложении.

СПИСОК ЛИТЕРАТУРЫ

1. 1C: Предприятие [Электронный документ] (<http://v8.1c.ru>.) Проверено 07.06.2007
2. ООО Научно-производственный центр «КОСМОС-2» [Электронный документ] (<http://www.cosmos2.aaanet.ru/>) Проверено 07.06.2007
3. ЗАО ИВЦ ИНСОФТ [Электронный документ] ([http://projects.economy.gov.ru/](http://projects.economy.gov.ru/pms/DownloadFile.aspx/%D0%9F%D0%BE%D1%8F%D1%81%D0%BD%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F%20%D0%B7%D0%B0%D0%BF%D0%B8%D1%81%D0%BA%D0%B0_%D0%BA%20%D0%9F%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D1%83%20%D0%A0%D0%B0%D1%81%D0%BF%D0%BE%D1%80%D1%8F%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F%20%D0%BA%20%D0%9A%D0%BE%D0%BD%D1%86%D0%B5%D0%BF%D1%86%D0%B8%D0%B8%20%D0%A1%D0%9F%D0%A3%D0%9D_%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D1%8F%201.doc?workproductid=2e133887-0ae0-4a3a-a3f0-080523cdf0b1)) Проверено 07.06.2007
4. Визуальный словарь [Электронный документ] (<http://i.viwo.ru/> ) Проверено 07.06.2007
5. Боггс У., Боггс М. UML и Rational Rose. 2002. / Боггс У., Боггс М. – М.: ЛОРИ. - 2002. - 582 с.
6. Вендров, А.М. CASE технологии Современные методы и средства проектирования информационных систем. / А.М. Вендров.- М.: Финансы и статистика, 1998. – 193 с.
7. Вигерс, К. Разработка требований к программному обеспечению.: Пер. с англ. / К. Вигерс.:– М.: Издательско-торговый дом «Русская редакция», 2004
8. Фатрелл, Т. Управление программными проектами: достижение оптимального качества при минимуме затрат.: Пер. с англ. / Р.Т. Фатрелл, Д.Ф. Шафер, Л.И. Шафер. – М.: Издательский дом "Вильямс", 2003.
9. Вигерс, К. Разработка требований к программному обеспечению.: Пер. с англ. / К. Вигерс.:– М.: Издательско-торговый дом «Русская редакция», 2004.
10. Мацяшек, Л.А. Анализ требований и проектирование систем. Разработка информационных систем с использованием UML. ./ Л. А. Мацяшек. Пер. с англ. – М.: Издательский дом «Вильямс». – 2002.
11. Атре, Ш. Структурный подход к организации баз данных / Ш. Атре.: Пер. с англ. А.А. Александрова и В.Ш. Будзко; под ред В.И. Будзко. – М.: Финансы и статистика. 1983. – 468 с.
12. Алехина, Г.В. Информационные технологии в экономике и управлении. / Г.В. Алехина.– М.:МЭСИ, 2002. – 635 с.
13. Буч Г. Объектно-ориентированный анализ и проектирование. / Буч Г.: Пер. с англ. – М: «Издательство Бином», 1999.
14. Буч Г., Рамбо Д., Джекобсон А. UML – руководство пользователя. / Буч Г., Рамбо Д., Джекобсон А.: Пер с англ. – М: «ДМК», 2001
15. Избачков, Ю.С. Информационные системы. Учебник для вузов / Ю.С. Избачков, В.Н. Петров. 2-е изд. – СПб.: Питер, 2005. – 739 с.
16. Microsoft Corporation Проектирование и реализация баз данных Microsoft SQL Server 2000. Учебный курс MCAD/MCSE, MCDBA/Пер. с англ. — 2-е изд., испр. — М.: Издательско-торговый дом Русская Редакция,
17. 2003. - 512стр.
18. Конноли Т., Бегг К. Базы данных. Проектирование, реализация и сопровождение. Теория и практика. / Конноли Т., Бегг К.: Пер. с англ. – М.: Изд. Дом «Вильямс», 2001. – 1120 с.
19. Дейт, К. Дж. Введение в системы баз данных. / К. Дж. Дейт.: Пер. с англ. – М.: Изд. Дом «Вильямс», 2002. – 1072с.
20. Диго, С.М. Проектирование и использование баз данных. / С.М. Диго. - М.: Финансы и статистика. 1995. – 216с.
21. Когаловский, М.Р. Энциклопедия технологий баз данных / М.Р. Когаловский. – М.: Финансы и статистика, 2002. – 800с.
22. Грофф, Дж. Р. Энциклопедия SQL. / Дж. Р. Грофф, П. Н. Вайнберг.: Пер. с англ. – СПб: «Питер», 2003. – 896 с.
23. Страуструп Б. Язык программирования С++, 3-е изд./Пер. с англ. – М.: «Издательство Бином», СПб: «Невский диалект», 1999. – 991 с.
24. Лабор В.В., Си Шарп: Создание приложений для Windows/ В. В. Лабор.— Мн.: Харвест, 2003. -384 с.
25. Джесс Либерти. Microsoft Visual C# Создание Net приложений: Пер. с англ. – К.: ВЕК+, М.: ЭНТРОП, 2002. – 704 с.
26. Visual Studio Documentation. MSDN Library –2005.
27. А. Горев, С. Макашарипов, Р. Ахаян. Эффективная работа с СУБД - 2004
28. Сеппа Д. Microsoft ADO.NET/Пер. с англ. — М.: Издательско-торговый домРусская Редакция, 2003- — 640 стр.
29. Троелсен. Э. С# и платформа .NET. Библиотека программиста. — СПб.: Питер, 2004. —796 с.:
30. Трельсен Э. Модель COM и применение ATL 3.0. / Пер. с англ. – СПб: BHV. 2000. – 926 с.
31. Армстронг Т. ActiveX: создание Web-приложений. / Пер. с англ. – К.: BHV. 1998. – 592 с.
32. Справочник по Microsoft OLE DB 1.1. / Пер. с англ. – М.: Издательский отдел «Русская редакция» ТОО «Channel Trading Ltd». 1997. – 624 с.
33. Ван Тассел Д. Стиль, разработка, эффективность, отладка и испытание программ. – М.: Мир. – 1981
34. Коллинз Г. Структурные методы разработки систем: от стратегического планирования до тестирования. / Коллинз Г., Блей Дж. Пер. с англ. – М.: Финансы и статистика, 1986. 264 с.
35. Богдатских, В.А. Экономика, разработка и использование программного обеспечения ЭВМ: Учебник. / В.А. Богдатских. - М.: Финансы и статистика, 1995. – 288 с.
36. Корнеев, И.К. Информационные технологии в управлении / И.К Корнеев, В.А. Машурцев . – М.:ИНФРА – М, 2001. – 651 с.
37. Хубаев, Г.Н. Экономическая оценка потребительского качества программных средств: Текст лекций / Г.Н. Хубаев. - РГЭА.: Ростов-на-Дону, 1997. – 94 с.
38. Хубаев, Г.Н. Экономика проектирования и применения банков данных / Г.Н. Хубаев. - Ростов-на-Дону: Изд-во РИСХМ, 1989. – 69 с
39. RAD материал из Википедия – свободная энциклопедия [Электронный документ] ([http://ru.wikipedia.org/](http://projects.economy.gov.ru/pms/DownloadFile.aspx/%D0%9F%D0%BE%D1%8F%D1%81%D0%BD%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F%20%D0%B7%D0%B0%D0%BF%D0%B8%D1%81%D0%BA%D0%B0_%D0%BA%20%D0%9F%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D1%83%20%D0%A0%D0%B0%D1%81%D0%BF%D0%BE%D1%80%D1%8F%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F%20%D0%BA%20%D0%9A%D0%BE%D0%BD%D1%86%D0%B5%D0%BF%D1%86%D0%B8%D0%B8%20%D0%A1%D0%9F%D0%A3%D0%9D_%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D1%8F%201.doc?workproductid=2e133887-0ae0-4a3a-a3f0-080523cdf0b1)) Проверено 07.06.2007

ПРИЛОЖЕНИЯ

Приложение А - Спецификация требований к программному обеспечению

Введение

Назначение

Cпецификация требований к ПО описывает функциональные и нефункциональные требования к информационной системе отдела воинского учета. Этот документ предназначен для команды, которая будет реализовывать и проверять корректность работы системы.

Объем проекта и функции продукта

Продукт позволит осуществить:

* Отказаться от бумажного процесса постановки на первоначальный учет:
* Структурировать хранящиеся данные;
* Уменьшить площадь хранимой информации за счет использования информационных технологий;

Программа будет использоваться для доступа к базе данных, хранящей информацию о :

* .Гражданине подлежащего воинскому призыву;
* Родителях гражданина;
* Ближайших родственниках;
* Паспортные данные;
* Месте рождения;
* Месте регистрации и фактическом месте проживания;
* Категориях граждан;
* Гражданах снятых с учета по причине смены места жительства;
* Гражданах снятых с учета по достижении 50 лет.

Общее описание

Общий взгляд на продукт

Информационная система позволит отказаться от бумажного процесса постановки на первоначальный воинский учет и перейти к электронной версии постановки на учет. Продукт позволит:

* Минимизировать сроки создания индивидуальной карточки гражданина;
* Формировать необходимых отчетов по заданным критериям;
* Получать необходимые сведения по конкретному лицу;
* Ускорить поиск данных;
* Вести реестр категорий граждан;
* Получать сведения о лицах снятых с учета по достижении 50 лет или сменивших место жительства;
* Получать сведения о лицах проходящих действительную воинскую службу;
* Получать сведения о лицах находящихся в запасе, с указанием их звания и профессии.

Классы и характеристики пользователей

В таблице приведены основные категории пользователей, использующих проектируемую ИС.

|  |  |
| --- | --- |
| Класс пользователей | Описание |
| 1. Инспектор ВУС
 | Персонал отдела воинского учета занимающийся воинским учетом граждан в поселении |
| 1. Начальник отдела
 |  |

Операционная среда

Операционная среда-1. Минимальные требования к операционной системе - Windows XP.

Ограничения дизайна и реализации

Ограничения дизайна и реализации-1. Приложение должно быть написано на высокоуровневом языке C#.

Ограничения дизайна и реализации-2. База данных должна быть спроектирована в MS SQL Server 2005

Ограничения дизайна и реализации-3. Приложение должно быть реализовано как клиент-серверная система, в которой модули, управляющие внешними устройствами, являются серверами автоматизации.

Документация для пользователей

Документация для пользователей-1. Разрабатывается руководство пользователя.

Функции системы

Функциональные требования

| Функция | Требование |
| --- | --- |
| 1. Создать карточку
 | Сохранить в базе данных сведения о гражданине, внесенные с помощью формы |
| 1. Получить список всех призывников
 | Получить список всех граждан подлежащих воинскому призыву |
| 1. Найти запись
 | Найти необходимые сведения по введенным критериям предлагаемых ИС |
| 1. Сформировать отчет
 | Сформировать отчет по конкретному гражданину |
| 1. Снять с учета
 | Снять с учета гражданина по причине смены места жительства |
| 1. Изменить запись
 | Провести корректировку уже сохраненной записи в базе данных |

Требования к внешнему интерфейсу

Интерфейсы пользователя

Интерфейсы пользователя-1. Экраны вывода должны соответствовать общепринятым стандартам.

Интерфейсы пользователя-2. Система должна обеспечивать ссылку на справку на каждой форме, объясняющую, как пользоваться этой формой.

Интерфейсы пользователя-3. Формы должны предоставлять полную возможность навигации и выбор при помощи клавиатуры и мыши.

Программные интерфейсы

Интерфейсы передачи информации

Интерфейсы передачи информации-1. В базе данных ИС должна сохраняться информация о гражданах, их родителях, ближайших родственниках, паспортные данные гражданина, его текущий статус, категория годности, звание если таковое есть.

Требования к производительности

Требования к производительности-1. Загрузка данных из БД не должна превышать более чем 5 сек

Требования к охране труда

Требования к охране труда не определены.

Требования к безопасности

Требования к безопасности не определены.

Атрибуты качества ПО

Доступность-1. Система должна быть доступна в рабочее время с 08.00 до 17.00 для инженера разработчика и во время проведения испытаний

Надежность-1. Система не должна нарушать целостность данных.

Приложение Б. Прототиты пользовательского интерфейса

Рисунок Б.1 – Внешний вид основного окна программы.

Рисунок Б.2 – Окно медицинской карты призывника.

Приложение В. Атрибуты управляющих таблиц проектируемой ИС

Таблица В.1 – Спецификация таблиц базы данных

|  |  |  |
| --- | --- | --- |
| Имя поля | Тип | Значение |
| Атрибуты таблицы «Grazdanin» -Гражданин |
| ID | int | Уникальный номер гражданина |
| Name | text | Имя гражданина |
| SecondName | text | Фамилия |
| LastName | text | Отчество |
| GerlsName | text | Девичья фамилия  |
| Rozd\_Oblast | text | Название региона |
| Rozd\_Raion | text | Название района |
| Rozd\_Gorod | text | Название города/села |
| DataRozd | datetime | Дата рождения |
| CreateDay | datetime | Дата записи |
| ID\_Relatives | int | Уникальный номер гражданина |
| ID\_Father | int | Уникальный номер гражданина |
| ID\_Mother | int | Уникальный номер гражданина |
| ID\_Adress\_Pasport | int | Уникальный номер гражданина |
| ID\_Adress\_Fuctual | int | Уникальный номер гражданина |
| Foto | image | Хранимая фотография гражданина |

Таблица В.2 – Спецификация таблиц базы данных

|  |  |  |
| --- | --- | --- |
| Имя поля | Тип | Значение |
| Атрибуты таблицы «Passport\_Data» -Паспортные данные |
| ID | int | Уникальный номер гражданина |
| Pas\_Ser\_1pole | nchar(10) | Серия паспорта |
| Pas\_Ser\_2pole | nchar(10) | Серия паспорта |
| Nomer | nchar(10) | Номер паспорта |
| Vidan | text | Подразделение выдавшее паспорт |
| DateVidathi | datetime | Дата выдачи |
| Kod\_1pole | nchar(10) | Код подразделения выдавшего паспорт |
| Kod\_2pole | nchar(10) | Код подразделения выдавшего паспорт |
| Атрибуты таблицы «Fuctual\_Place\_Residence» -Фактическое место проживания |
| ID | int | Уникальный номер гражданина |
| Endex | nchar(10) | Индекс |
| Oblast | text | Область фактического проживания |
| Raion | text | Район фактического проживания |
| Gorod | text | Город фактического проживания |
| Street | text | Улица фактического проживания |
| House | nchar(10) | Дом фактического проживания |
| ZaregDate | datetime | День регистрации места жительства по паспорту |
|  |  |  |
| Kvartira | nchar(10) | Квартира фактического проживания |
| HomeTel | nchar(10) | Домашний телефон |
| WorkTel | nchar(10) | Рабочий телефон |

Таблица В.4 – Спецификация таблиц базы данных

|  |  |  |
| --- | --- | --- |
| Имя поля | Тип | Значение |
| Атрибуты таблицы «Get\_Out» -Сменившие место проживания |
| ID | int | Уникальный номер гражданина |

Таблица В.5 – Спецификация таблиц базы данных

|  |  |  |
| --- | --- | --- |
| Имя поля | Тип | Значение |
| Атрибуты таблицы «In\_Army» -Проходящие действительную воинскую службу  |
| ID | int | Уникальный номер гражданина |
| ID\_MedCard | int | ID Мед карты гражданина |

Таблица В.6 – Спецификация таблиц базы данных

|  |  |  |
| --- | --- | --- |
| Имя поля | Тип | Значение |
| Атрибуты таблицы «Med\_card» -Медицинская карта гражданина  |
| ID | int | Уникальный номер гражданина |
| KategorGodn | text | Категория годности гражданина к службе в армии |

Таблица В.7 – Спецификация таблиц базы данных

|  |  |  |
| --- | --- | --- |
| Имя поля | Тип | Значение |
| Атрибуты таблицы «Negoden» - Граждане, не подлежащие призыву в армию по состоянию здоровья  |
| ID | int | Уникальный номер гражданина |
| ID\_MedCard | int | ID Мед карты гражданина |

Таблица В.8 – Спецификация таблиц базы данных

|  |  |  |
| --- | --- | --- |
| Имя поля | Тип | Значение |
| Атрибуты таблицы «Officcer» -Офицер запаса  |
| ID | int | Счетчик |
| ID\_Zapas | int | ID Запасника |
| Zvanie | text | Звание гражданина находящегося в запасе |
| Uthebnoe\_Zavidenie | text | Название учебного учреждения |

Таблица В.9 – Спецификация таблиц базы данных

|  |  |  |
| --- | --- | --- |
| Имя поля | Тип | Значение |
| Атрибуты таблицы «Sniat\_50» -Граждане, снятые с учета по достижении 50 лет  |
| ID | int | Уникальный номер гражданина |
| Zvanie | text | Звание гражданина на момент снятия с учета |

Таблица В.10 – Спецификация таблиц базы данных

|  |  |  |
| --- | --- | --- |
| Имя поля | Тип | Значение |
| Атрибуты таблицы «Soldat» -Солдат запаса  |
| ID | int | Счетчик |
| ID\_Zapas | int | ID Запасника |
| Professia | text | Профессия полученная во время прохождения действительной воинской службы |

Таблица В.11 – Спецификация таблиц базы данных

|  |  |  |
| --- | --- | --- |
| Имя поля | Тип | Значение |
| Атрибуты таблицы «Zapasnik» -Граждане находящиеся в запасе |
| ID | int | Уникальный номер гражданина |

Таблица В.12 – Спецификация таблиц базы данных

|  |  |  |
| --- | --- | --- |
| Имя поля | Тип | Значение |
| Атрибуты таблицы «Prizivnik» - Гражданин подлежащий призыву на действительную воинскую службу  |
| ID | int | Уникальный номер гражданина |
| ID\_MedCard | int | ID Мед карты гражданина |

Приложение Г - Выбор модели жизненного цикла

(обязательное)

Таблица Г.1 – Выбор модели ЖЦ на основе характеристик требований

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Требования | Каскадная | V-образная | Прототипирование | Спиральная | RAD | Инкрементная |
| Являются ли требования легко определимыми и/или хорошо известными | Да | Да | Нет | Нет | Да | Нет |
| Могут ли требования заранее определятся в цикле | Да | Да | Нет | Нет | Да | Да |
| Часто ли изменяются требования в цикле | Нет | Нет | Да | Да | Нет | Нет |
| Нужно ли демонстрировать требования с целью определения | Нет | Нет | Да | Да | Да | Нет |
| Требуется ли демонстрация возможностей проверка концепции  | Нет | Нет | Да | Да | Да | Нет |
| Будут ли требования отражать сложность системы | Нет | Нет | Да | Да | Нет | Да |
| Обладает ли требование функциональными свойствами на раннем этапе | Нет | Нет | Да | Да | Да | Да |

Таблица Г.2 – Выбор модели ЖЦ на основе характеристик команды разработчиков

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Команда разработчиков проекта | Каскадная | V-образная | Прототипирование | Спиральная | RAD | Инкрементная |
| Являются ли проблемы предметной области проекта новыми для большинства разработчиков | Нет | Нет | Да | Да | Нет | Нет |
| Является ли технология предметной области проекта новой для большинства разработчиков  | Да | Да | Нет | Да | Да | Да |
| Являются ли инструменты, используемые проектом, новыми для большинства разработчиков | Да | Да | Нет | Да | Нет | Нет |
| Изменяются ли роли участников проекта во время ЖЦ | Нет | Нет | Да | Да | Нет | Да |
| Могут ли разработчики проекта пройти обучение | Нет | Да | Нет | Нет | Да | Да |
| Является ли структура более значимой для разработчиков, чем гибкость | Да | Да | Нет | Нет | Да | Да |
| Будет ли менеджер проекта строго отслеживать прогресс проекта | Да | Да | Нет | Да | Да | Да |
| Важна легкость распределения ресурсов | Да | Да | Нет | Нет | Да | Да |
| Приемлет ли команда равноправные обзоры инспекций | Да | Да | Да | Да | Да | Да |

Таблица Г.3 – Выбор модели ЖЦ на основе характеристик типа проектов и рисков

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Тип проекта и риски | Каскадная | V-образная | Прототипирование | Спиральная | RAD | Инкрементная |
| Будет ли проект идентифицировать новое направление продукта для организации | Нет | Нет | Да | Да | Нет | Да |
| Будет ли проект иметь тип системной интеграции | Нет | Да | Да | Да | Да | Да |
| Будет ли проект являться расширением существующей системы | Нет | Да | Нет | Нет | Да | Да |
| Будет ли финансирование проекта стабильным на всем протяжении ЖЦ | Да | Да | Да | Нет | Да | Нет |
| Ожидается ли длительная эксплуатация продукта в организации | Да | Да | Нет | Да | Нет | Да |
| Должна ли быть высокая степень надежности | Нет | Да | Нет | Да | Нет | Да |
| Будет ли система изменяться, возможно, с применением непредвиденных методов, на этапе сопровождения | Нет | Нет | Да | Да | Нет | Да |
| Является ли график ограниченным | Нет | Нет | Да | Да | Да | Да |
| Являются ли «прозрачными» интерфейсные модули | Да | Да | Нет | Нет | Нет | Да |
| Доступны ли повторноиспользуемые компоненты | Нет | Нет | Да | Да | Да | Нет |
| Являются ли достаточными ресурсы (время, деньги, инструменты, персонал) | Нет | Нет | Да | Да | Нет | Нет |

Таблица Г.4 – Выбор модели ЖЦ на основе характеристик коллектива пользователей

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Коллектив пользователей | Каскадная | V-образная | Прототипирование | Спиральная | RAD | Инкрементная |
| Будет ли присутствие пользователей ограниченно в ЖЦ | Да | Да | Нет | Да | Да | Да |
| Будут ли пользователи знакомы с определением системы | Нет | Нет | Да | Да | Нет | Да |
| Будут ли пользователи ознакомлены с проблемами предметной области | Нет | Нет | Да | Нет | Да | Да |
| Будут ли пользователи вовлечены во все фазы ЖЦ | Нет | Нет | Да | Нет | Да | Нет |
| Будет ли заказчик отслеживать ход выполнения проекта | Нет | Нет | Да | Да | Нет | Нет |

Приложение Д – Диаграмма Гантта

