

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«Рязанский государственный радиотехнический университет имени В.Ф.
Уткина»**

(ФБГОУ ВО «Рязанский государственный радиотехнический университет им.
В.Ф. Уткина», ФБГОУ ВО «РГРТУ», РГРТУ)

Факультет вычислительной техники

Кафедра электронных вычислительных машин

К защите

Руководитель проекта

дата, подпись

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К КУРСОВОМУ ПРОЕКТУ**

по дисциплине

«Базы данных и клиент-серверные приложения»

по теме

«Разработка информационной системы учета призывников с клиент-
серверной архитектурой»

Выполнил:

студент группы 846

Шмаков А.Ю.

дата сдачи на проверку,
подпись

Руководитель проекта:

Ассистент кафедры ЭВМ

Хизриева Н.И.

оценка

дата защиты, подпись

Рязань 2021

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	3
1 Концептуальное описание предметной области	7
2 Характеритики ИС	9
3 Анализ существующих решений для предметной области.....	11
4 Разработка общей структуры ИС	12
5 Разработка серверной части информационной системы	14
5.1 Инфологическое проектирование БД.....	14
5.2 Даталогическое проектирование БД	24
5.3 Программирование объектов БД	37
6 Разработка клиентского приложения.....	42
6.1 Выбор программных компонентов клиентской части	42
6.2 Разработка интерфейса пользователя.....	42
6.2.1 Разработка форм	42
6.3 Разработка сценария инсталляции клиентской программы	48
7. Разработка программной документации	53
8. Тестирование ИС	54
Заключение.....	55
СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ	56
ПРИЛОЖЕНИЕ А: сценарий создания объектов БД.	57
ПРИЛОЖЕНИЕ Б: сценарий заполнения таблиц БД.	67
ПРИЛОЖЕНИЕ В: исходный текст клиентской программы.	70
ПРИЛОЖЕНИЕ Г: сценарий инсталляции программы.....	83

ВВЕДЕНИЕ

В самом общем смысле база данных - это набор записей и файлов, организованных специальным образом. В компьютере, например, можно хранить фамилии и адреса друзей или клиентов. Один из типов баз данных - это документы, набранные с помощью текстовых редакторов и сгруппированные по темам. Другой тип - файлы электронных таблиц, объединяемые в группы по характеру их использования.

Первые модели данных

С ростом популярности СУБД в 70-80-х годах появилось множество различных моделей данных. У каждой из них имелись свои достоинства и недостатки, которые сыграли ключевую роль в развитии реляционной модели данных, появившейся во многом благодаря стремлению упростить и упорядочить первые модели данных.

Системы управления файлами

До появления СУБД все данные, которые содержались в компьютерной системе постоянно, хранились в виде отдельных файлов. Система управления файлами, которая обычно является частью операционной системы компьютера, следила за именами файлов и местами их расположения. В системах управления файлами модели данных, как правило, не использовались; эти системы ничего не знали о внутреннем содержимом файлов. Для такой системы файл, содержащий документ текстового процессора, ничем не отличается от файла, содержащего данные о начисленной зарплате.

Стремительный рост популярности SQL является одной из самых важных тенденций в современной компьютерной промышленности. За несколько последних лет SQL стал единственным языком баз данных. На сегодняшний день SQL поддерживают свыше ста СУБД, работающих как на персональных компьютерах, так и на больших ЭВМ. Был принят, а затем дополнен официальный международный стандарт на SQL. Язык SQL является

важным звеном в архитектуре систем управления базами данных, выпускаемых всеми ведущими поставщиками программных продуктов, и служит стратегическим направлением разработок компании Microsoft в области баз данных.

Зародившись в результате выполнения второстепенного исследовательского проекта компании IBM, SQL сегодня широко известен и в качестве мощного рыночного фактора.

Язык SQL

SQL является инструментом, предназначенным для обработки и чтения данных, содержащихся в компьютерной базе данных. SQL - это сокращенное название структурированного языка запросов (Structured Query Language). Как следует из названия, SQL является языком программирования, который применяется для организации взаимодействия пользователя с базой данных. На самом деле SQL работает только с базами данных одного определенного типа, называемых реляционными. На рис. 2.1 изображена схема работы SQL. Согласно этой схеме, в вычислительной системе имеется база данных, в которой хранится важная информация.

Если вычислительная система относится к сфере бизнеса, то в базе данных может храниться информация о материальных ценностях, выпускаемой продукции, объемах продаж и зарплате. В базе данных на персональном компьютере может храниться информация о выписанных чеках, телефонах и адресах или информация, извлеченная из более крупной вычислительной системы. Компьютерная программа, которая управляет базой данных, называется системой управления базой данных, или СУБД.

Если пользователю необходимо прочитать данные из базы данных, он запрашивает их у СУБД с помощью SQL. СУБД обрабатывает запрос, находит требуемые данные и посылает их пользователю. Процесс запрашивания данных и получения результата называется запросом к базе данных: отсюда и название — структурированный язык запросов. Однако это название не совсем соответствует действительности. Во-первых, сегодня SQL представляет собой

нечто гораздо большее, чем простой инструмент создания запросов, хотя именно для этого он и был первоначально предназначен. Несмотря на то, что чтение данных по-прежнему остается одной из наиболее важных функций SQL, сейчас этот язык используется для реализации всех функциональных возможностей, которые СУБД предоставляет пользователю, а именно:

- ✚ Организация данных. SQL дает пользователю возможность изменять структуру представления данных, а также устанавливать отношения между элементами базы данных.
- ✚ Чтение данных. SQL дает пользователю или приложению возможность читать из базы данных, содержащиеся в ней данные и пользоваться ими.
- ✚ Обработка данных. SQL дает пользователю или приложению возможность изменять базу данных, т.е. добавлять в нее новые данные, а также удалять или обновлять уже имеющиеся в ней данные.
- ✚ Управление доступом. С помощью SQL можно ограничить возможности пользователя по чтению и изменению данных и защитить их от несанкционированного доступа.
- ✚ Совместное использование данных. SQL координирует совместное использование данных пользователями, работающими параллельно, чтобы они не мешали друг другу.
- ✚ Целостность данных. SQL позволяет обеспечить целостность базы данных, защищая ее от разрушения из-за несогласованных изменений или отказа системы.

Таким образом, SQL является достаточно мощным языком для взаимодействия с СУБД. Во-вторых, SQL — это не полноценный компьютерный язык типа COBOL, FORTRAN или C. В SQL нет оператора IF для проверки условий, нет оператора GOTO для организации переходов и нет операторов DO или FOR для создания циклов. SQL является подязыком баз данных, в который входит около тридцати операторов, предназначенных для управления базами данных. Операторы SQL встраиваются в базовый язык, например COBOL, FORTRAN или C, и дают возможность получать доступ к

базам данных. Кроме того, из такого языка, как С, операторы SQL можно посылать СУБД в явном виде, используя интерфейс вызовов функций. Наконец, SQL — это слабо структурированный язык, особенно по сравнению с такими сильно структурированными языками, как С или Pascal. Операторы SQL напоминают английские предложения и содержат "слова-пустышки", не влияющие на смысл оператора, но облегчающие его чтение. В SQL почти нет нелогичностей, к тому же имеется ряд специальных правил, предотвращающих создание операторов SQL, которые выглядят как абсолютно правильные, но не имеют смысла.

Несмотря на не совсем точное название, SQL на сегодняшний день является единственным стандартным языком для работы с реляционными базами данных. SQL — это достаточно мощный и в то же время относительно легкий для изучения язык.

1 КОНЦЕПТУЛЬНОЕ ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

Организация воинского учета в органах государственной власти, органах исполнительной власти субъектов Российской Федерации, органах местного самоуправления поселений, органах местного самоуправления муниципальных округов и органах местного самоуправления городских округов (далее - органы местного самоуправления) и организациях входит в содержание мобилизационной подготовки и мобилизации.

2. Основной целью воинского учета является обеспечение полного и качественного укомплектования призывными людскими ресурсами Вооруженных Сил Российской Федерации, других войск, воинских формирований и органов в мирное время, а также обеспечение в периоды мобилизации, военного положения и в военное время:

а) потребностей Вооруженных Сил Российской Федерации, других войск, воинских формирований, органов и специальных формирований в мобилизационных людских ресурсах путем заблаговременной приписки (предназначения) граждан, пребывающих в запасе, в их состав;

б) потребностей органов государственной власти, органов местного самоуправления и организаций в трудовых ресурсах путем закрепления (бронирования) за ними необходимого количества руководителей и специалистов из числа граждан, пребывающих в запасе, работающих в этих органах и организациях.

3. Основными задачами воинского учета являются:

а) обеспечение исполнения гражданами воинской обязанности, установленной законодательством Российской Федерации;

б) документальное оформление сведений воинского учета о гражданах, состоящих на воинском учете;

в) анализ количественного состава и качественного состояния призывных и мобилизационных людских ресурсов для их эффективного использования в интересах обеспечения обороны страны и безопасности государства;

г) проведение плановой работы по подготовке необходимого количества военно-обученных граждан, пребывающих в запасе, для обеспечения мероприятий по переводу Вооруженных Сил Российской Федерации, других войск, воинских формирований и органов с мирного на военное время, в период мобилизации и поддержание их укомплектованности на требуемом уровне в военное время.

4. Основным требованием, предъявляемым к системе воинского учета, является постоянное обеспечение полноты и достоверности данных, определяющих количественный состав и качественное состояние призывных и мобилизационных людских ресурсов.

5. Функционирование системы воинского учета обеспечивается Министерством обороны Российской Федерации, Министерством внутренних дел Российской Федерации, Службой внешней разведки Российской Федерации, Федеральной службой безопасности Российской Федерации, органами исполнительной власти субъектов Российской Федерации, органами местного самоуправления и организациями.

6. Должностные лица органов государственной власти, органов исполнительной власти субъектов Российской Федерации, органов местного самоуправления и организаций обеспечивают исполнение гражданами обязанностей в области воинского учета в соответствии с законодательством Российской Федерации.

За состояние воинского учета отвечают военные комиссары.

2 ХАРАКТЕРИТИКИ ИС

Предметная область: Система воинского учета призывников

Пусть в базе данных требуется хранить следующую информацию:

- 1) ФИО призывника, его год рождения, адрес и телефон;
- 2) Название, адрес и телефон военкомата к которому приписан призывник;
- 3) Группу профессионального отбора призывника;
- 4) Отсрочку призывника и дату ее выдачи;
- 5) Специальную подготовку призывника;
- 6) Результаты медосмотра и дату его проведения;
- 7) ФИО, номер паспорта, должность и стаж сотрудников военкомата;
- 8) Состав медицинской комиссии;
- 9) Наименование, адрес и телефон учреждений, которые проводят специальную подготовку призывников.

Бизнес правила применяемые в информационной системе учета призывников:

- В отношении Военкомат атрибуты Наименование и Телефон, должны быть уникальными и атрибутам Адрес и Наименование не может быть присвоено значение NULL;
- В отношении Специальное учреждение атрибуты Наименование и Телефон, должны быть уникальными и атрибуту Наименование не может быть присвоено значение NULL;
- В отношении Профессиональный отбор атрибут описание должен быть уникальным;
- В отношении Отсрочка атрибуту срок не может быть присвоено значение NULL;
- В отношении Сотрудник атрибут номер паспорта должен быть уникален и атрибуту военкомат не может быть присвоено значение NULL;

- В отношении Призывник атрибутам ФИО, военкомат, группа профотбора не может быть присвоено значения NULL;
- В отношении Специальная подготовка атрибут код учреждения не может иметь значение NULL;
- В отношении Состав Медкомиссии атрибуту медкомитет не может быть присвоено значение NULL;
- В отношении Отсрочка призывника атрибуту отсрочка не может быть присвоено значение NULL;
- В отношении Направление на специальную подготовку атрибуту Наименование специальной подготовки не может быть присвоено значение NULL;
- В отношении Заключение врачебной комиссии атрибуту Медкомитет и Степень годности не может быть присвоено значение NULL.

3 АНАЛИЗ СУЩЕСТВУЮЩИХ РЕШЕНИЙ ДЛЯ ПРЕДМЕТНОЙ ОБЛАСТИ

В настоящее время в большинстве военных учреждений на бумажных носителях храниться достаточно много информации, порой данной информации намного больше чем цифровой. Данный факт затрудняет обеспечение надежности, сохранности и удобства работы с данными.

В курсовом проекте разрабатывается система учета призывников, которая должна обеспечивать качественный контроль граждан состоящих на воинском учете.

4 РАЗРАБОТКА ОБЩЕЙ СТРУКТУРЫ ИС

В настоящее время существует несколько вариантов представления взаимодействия сервера и персонального компьютера.

Файл – серверная архитектура.

База данных расположена на мощном выделенном компьютере (сервере), а персональные компьютеры подключены к нему по локальной сети. На этих компьютерах установлены клиентские программы, обращающиеся к базе данных по сети. Преимущество такой архитектуры заключается в возможности одновременной работы нескольких пользователей с одной базой данных.

Недостаток такого подхода - большие объемы информации, передаваемой по сети. Вся обработка выполняется на клиентских местах, где фактически формируется копия базы данных. Это приводит к ограничению максимально возможного числа пользователей и большим задержкам при работе с базой. Эти задержки вызываются тем, что на уровне конкретной таблицы одновременный доступ невозможный. Пока программа на одном из клиентских мест не закончит работу с таблицей (например, не выполнит модификацию записей), другие программы не могут обращаться к этой таблице. Это называется блокировкой на уровне таблицы и исключает возникновение путаницы в ее содержимом.

Клиент – серверная архитектура.

В такой архитектуре на сервере не только хранится БД, но и работает программа СУБД, обрабатывающая запросы пользователей и возвращающая им наборы записей. При этом программы пользователей уже не работают, например, с БД как набором физических фалов, а обращаются к СУБД, которая выполняет операции. Нагрузка с клиентских мест при этом снимается, так как большая часть работы происходит на сервере. СУБД автоматически следит за целостностью и сохранностью БД, а также контролирует доступ к информации с помощью службы паролей. Клиент –

серверные СУБД допускают блоки на уровне записи и даже отдельного поля.

Это означает, что с таблицей может работать любое число пользователей, но доступ к функции изменения конкретной записи или одного из ее полей обеспечен только одному из них.

Основной недостаток этой архитектуры не очень высокая надежность. Если сервер выходит из строя, вся работа останавливается.

Распределенная архитектура.

В сети работает несколько серверов, и таблицы баз данных распределены между ними для достижения повышенной эффективности. На каждом сервере функционирует своя копия СУБД. Кроме того, в подобной архитектуре обычно используются специальные программы, так называемые серверы приложений. Они позволяют оптимизировать обработку запросов большого числа пользователей и равномерно распределить нагрузку между компьютерами в сети.

Недостаток распределенной архитектуры заключается в довольно сложном и дорогостоящем процессе ее создания и сопровождения (администрирования), а также, а высоких требованиях к серверным компьютерам.

Интернет – архитектура.

Доступ к базе данных и СУБД (распространенных на одном компьютере или в сети) осуществляется из браузера по стандартному протоколу. Это предъявляет минимальные требования к клиентскому оборудованию. Такие программы называют «тонкими клиентами», потому что они способны работать даже на ПК с процессором 80386. Благодаря стандартизации всех протоколов и внедрять. Например, можно не организовывать локальную сеть, а обращаться к серверу через Интернет в локальной сети (в таком случае говорят о технологиях интернет). В этом случае не требуется разрабатывать специальные клиентские программы или придумывать собственные спецификации обмена данными между сервером и клиентскими местами. Достаточно использовать готовые браузеры и программные решения.

5 РАЗРАБОТКА СЕРВЕРНОЙ ЧАСТИ ИНФОРМАЦИОННОЙ СИСТЕМЫ

5.1 Инфологическое проектирование БД

Инфологическое проектирование основано на ER-методе.

Суть ER-метода: построение ER-диаграммы, отображающей в графической форме основные объекты предметной области, связи между ними и определение характеристик этих связей.

Затем, по определенным правилам выполняется переход от ER-диаграммы к таблицам БД. Результатом является схема БД.

В рамках ER-метода существует два типа диаграмм:

1. ER-диаграммы для экземпляров сущностей и связей.
2. ER-диаграммы для классов сущностей и связей.

Связь между сущностями имеет две характеристики:

Степень или кратность связи:

- 1:1 – один к одному
- 1:N – один ко многим
- N:M – многие ко многим

Класс принадлежности сущности к связи:

Обязательный – каждый экземпляр одной сущности связан с конкретным экземпляром другой сущности, в противном случае класс сущности является необязательным.

Правила формирования отношений по ER-диаграммам.

Формирование отношений для бинарных связей 1:1.

Правило 1:

Если степень бинарной связи 1:1 и класс принадлежности обеих сущностей к связи обязательный, то формируют одно отношение. Ключом может быть ключ любой сущности.

Правило 2:

Если связь 1:1 и класс принадлежности одной сущности обязательный, а другой необязательный, то формируется два отношения, по одному для каждой сущности, где ключом является ключ соответствующей сущности. Кроме этого ключ сущности с необязательным классом принадлежности добавляется в качестве атрибута в отношение для сущности с обязательным классом принадлежности.

Правило 3:

Если степень связи 1:1 и класс принадлежности обеих сущностей к связи необязательный, то формируют три отношения: два для сущностей и одно для связи. В первых двух отношениях ключом является ключ соответствующей сущности. Отношение для связи должно содержать ключи обеих сущностей. В этом отношении ключом может быть ключ любой сущности.

Формирование отношений для бинарных связей 1:N

В такой ситуации используются два правила. Каждое из них определяется классом принадлежности N-связной сущности. Класс принадлежности односвязной сущности на результат не влияет.

Правило 4:

Если степень бинарной связи 1:N и класс принадлежности N-связной сущности обязательный, то формируется два отношения, по одному для каждой сущности. Ключ отношения - это ключ соответствующей сущности. Кроме этого в отношение для N-связной сущности добавляется в качестве атрибута ключ односвязной сущности.

Правило 5:

Если степень бинарной связи 1:N, а класс принадлежности N-связной сущности необязательный, то формируют три отношения: по одному для каждой сущности и одно для связи. Ключом в первых двух отношениях является ключ соответствующей сущности. Отношение для связи должно содержать ключи обеих сущностей. Ключ этого отношения –это ключ N-связной сущности.

Формирование отношений для бинарных связей N:M

Используется одно правило. Класс принадлежности сущности к связи на результат не влияет.

Правило 6:

Если степень бинарной связи N:M, то формируется три отношения: два отношения для сущностей, где ключом является ключ соответствующей сущности и одно отношение для связи, где первичный ключ состоит из ключей обеих сущностей.

Формирование отношений для тернарных связей.

Правило 7:

При наличии тернарной связи необходимо использовать 4 отношения: три для каждой сущности, где ключом является ключ соответствующей сущности и одно для связи, где ключ содержит ключи всех сущностей. При наличии n -ой связи строится $n+1$ отношение.

Основные понятия ER-метода:

Сущность – это объект предметной области.

Экземпляр сущности – конкретный представитель объекта.

Связь – соединение между двумя и более сущностями.

Экземпляр связи – конкретная связь между экземплярами объектов.

Ключ сущности – множество атрибутов для однозначной идентификации сущности.

Проектирование базы данных с помощью ER-метода включает следующие этапы:

Выявление сущностей и связей.

В предметной области можно выделить следующие сущности:

- Военкомат (Код);
- Сотрудники (ФИО);
- Призывник (Личное Дело);
- Отсрочка (Код);
- Профессиональный отбор (Группа);
- Специальная подготовка (Наименование);
- Специальное учреждение (Код);
- Медкомитет (Код);

**В предметной области можно выделить следующие связи
между сущностями:**

- В Военкомате работают Сотрудники;
- К Военкомату приписан Призывник;
- Призывник проходит Специальную подготовку;
- Призывник имеет Отсрочку;
- Призывник проходит Профессиональный отбор;
- Сотрудник состоит в Медкомитете;
- Специальное учреждение проводит Специальную подготовку;
- Медкомитет выставляет медицинское заключение Призывнику

Построение ER-диаграмм

1) Связь в Военкомате работают Сотрудники

Для степени связи:

- ✓ В Военкомате может работать несколько Сотрудников
- ✓ Сотрудник может работать только в одном Военкомате

Для класса принадлежности сущности к связи:

- ✓ В Военкомате обязательно работает Сотрудник
- ✓ Сотрудник обязательно работает в Военкомате



2) Связь к Военкомату приписан Призывник

Для степени связи:

- ✓ К Военкомату может быть приписано несколько Призывников
- ✓ Призывник может быть приписан только к одному Военкомату

Для класса принадлежности сущности к связи:

- ✓ К Военкомату необязательно приписан Призывник
- ✓ Призывник обязательно приписан к Военкомату



3) Связь Призывник проходит Специальную подготовку

Для степени связи:

- ✓ Специальную подготовку могут проходить несколько Призывников
- ✓ Призывник может проходить только одну Специальную подготовку

Для класса принадлежности сущности к связи:

- ✓ Специальная подготовка необязательно проходит Призывником
- ✓ Призывник необязательно проходит Специальную подготовку



4) Связь Призывник имеет Отсрочку

Для степени связи:

- ✓ Призывник может иметь только одну Отсрочку
- ✓ Отсрочка может быть приписана к нескольким Призывникам

Для класса принадлежности сущности к связи:

- ✓ Призывник обязательно имеет Отсрочку
- ✓ Отсрочка обязательно имеется у Призывника



5) Связь Призывник проходит Профессиональный отбор

Для степени связи:

- ✓ Призывник может иметь только один Профотбор
- ✓ Профотбор может быть приписан к нескольким Призывникам

Для класса принадлежности сущности к связи:

- ✓ Призывник обязательно имеет Профотбор
- ✓ Профотбор обязательно приписан к Призывнику



6) Связь Сотрудник состоит в Медкомитете

Для степени связи:

- ✓ Медкомитет может иметь несколько Сотрудников
- ✓ Сотрудник может состоять только в одном Медкомитете

Для класса принадлежности сущности к связи:

- ✓ Медкомитет необязательно имеет Сотрудника
- ✓ Сотрудник необязательно состоит в Медкомитете



7) Связь Специальное учреждение проводит Специальную подготовку

Для степени связи:

- ✓ Специальное учреждение может проводить несколько Специальных подготовок
- ✓ Специальную подготовку может проводить только одно Специальное учреждение

Для класса принадлежности сущности к связи:

- ✓ Специальное учреждение необязательно проводит Специальную подготовку
- ✓ Специальную подготовку обязательно проводит Специальное учреждение



8) Медкомитет выставляет заключение Призывнику

Для степени связи:

- ✓ Много Медкомитетов выставляют много медицинских заключений Призывникам

Для класса принадлежности сущности к связи:

- ✓ Медкомитет обязательно выставляет медицинское заключение Призывнику
- ✓ Медицинское заключение Призывника обязательно выставляется Медкомитетом



Общая ER-диаграмма представлена на рисунке 5.1.

Общая ER-диаграмма

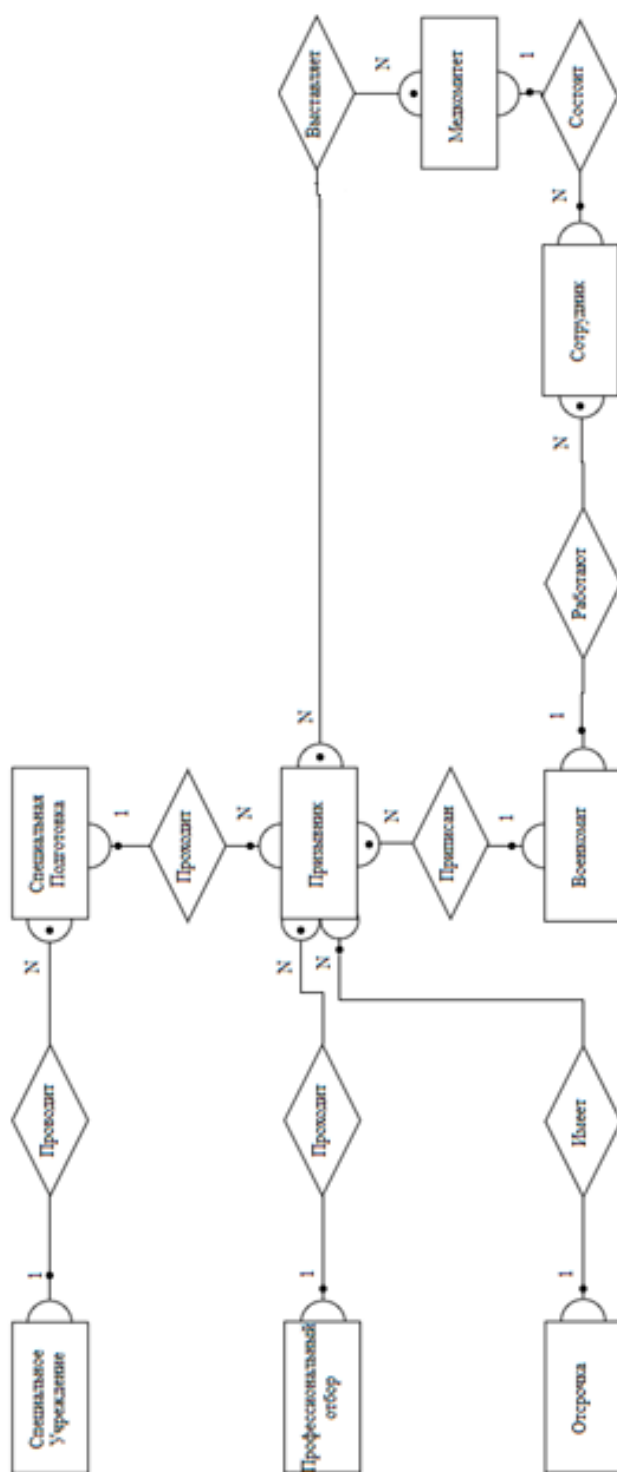


Рисунок 5.1 – Общая ER-диаграмма

5.2 Даталогическое проектирование БД

Формирование предварительных отношений по ER-диаграмме

В Военкомате работают Сотрудники

По правилу 4 имеем следующие отношения:

- Военкомат (Код);
- Сотрудник (ФИО, Военкомат).

К Военкомату приписан Призывник

По правилу 4 имеем следующие отношения:

- Военкомат (Код);
- Призывник (Личное дело, Военкомат).

Призывник проходит Специальную подготовку

По правилу 5 имеем следующие отношения:

- Призывник (Личное дело, Военкомат);
- Специальная подготовка (Наименование);
- Направление на специальную подготовку (Призывник, Наименование специальной подготовки).

Призывник имеет Отсрочку

По правилу 5 имеем следующие отношения:

- Призывник (Личное дело, Военкомат);
- Отсрочка (Код);
- Отсрочка призывника (Призывник, Код отсрочки).

Призывник проходит Профессиональный отбор

По правилу 4 имеем следующие отношения:

- Призывник (Личное дело, Военкомат, Группа профессионального отбора);
- Профессиональный отбор (Группа).

Сотрудник состоит в Медкомитете

По правилу 5 имеем следующие отношения:

- Сотрудник (ФИО, Военкомат);
- Медкомитет (Код);
- Состав Медкомиссии (Сотрудник, Медкомитет).

Специальное учреждение проводит Специальную подготовку

По правилу 4 имеем следующие отношения:

- Специальное учреждение (Код);
- Специальная подготовка (Наименование, Код учреждения).

Медкомитет выставляет медицинское заключение Призывнику

По правилу 6 имеем следующие отношения:

- Медкомитет (Код);
- Призывник (Личное дело, Военкомат, Группа профессионального отбора);
- Заключение врачебной комиссии (Медкомитет, Призывник);

Подготовка списка атрибутов. Распределение их по отношениям

- 1) Военкомат (Код, Наименование, Адрес, Телефон);
- 2) Сотрудник (ФИО, Военкомат, Должность, Стаж, Номер Паспорта);
- 3) Призывник (Личное дело, ФИО, Военкомат, Группа профессионального отбора, Телефон, Адрес, Год Рождения);
- 4) Состав Медкомиссии (Сотрудник, Медкомитет);
- 5) Медкомитет (Код, Дата основания, Описание);
- 6) Заключение врачебной комиссии (Призывник, Дата Проведения, Медкомитет, Степень годности);
- 7) Отсрочка (Код, Срок, Описание);
- 8) Отсрочка призывника (Призывник, Отсрочка, Дата Выдачи);
- 9) Специальное учреждение (Код, Наименование, Адрес, Телефон);

- 10) Специальная подготовка (Наименование, Код Учреждения);
- 11) Направление на специальную подготовку (Призывник,
Наименование специальной подготовки);
- 12) Профессиональный отбор (Группа, Описание).

Проверка отношений на БКНФ

Существует два метода проектирования базы данных: это ER-метод и метод нормальных форм. В настоящее время используются оба метода. ER-метод непосредственно для проектирования базы данных, а метод нормальных форм для проверки правильности результата проектирования.

Метод нормальных форм предполагает, что вся информация первоначально хранится в одном отношении. Процесс проектирования заключается в переводе отношения из первой нормальной формы в более высокие нормальные формы по определенным правилам. Каждой нормальной форме соответствует определенный набор ограничений. Выделяют следующие нормальные формы:

- 1НФ (первая нормальная форма)
- 2НФ (вторая нормальная форма)
- 3НФ (третья нормальная форма)
- БКНФ (нормальная форма Бойса-Кодда)
- 4НФ (четвертая нормальная форма)
- 5НФ (пятая нормальная форма или проекционно-соединительная нормальная форма)
- ДКНФ (доменно-ключевая нормальная форма)

Каждая нормальная форма сохраняет свойства предыдущих нормальных форм. Переход к более высокой нормальной форме выполняется путем декомпозиции отношения на два или более отношений, которые удовлетворяют требованиям этой нормальной формы.

1НФ - отношение находится в 1НФ, если на пересечении каждой строки и столбца находится ровно одно значение.

2НФ – отношение находится в 2НФ, если оно находится в 1НФ и каждый не ключевой атрибут функционально полно зависит от любого потенциального ключа (в частности первичного)

3НФ – отношение находится в 3НФ, если оно находится в 2НФ и в нем нет транзитивных зависимостей не ключевых атрибутов от любого потенциального ключа (в частности первичного)

БКНФ – отношение находится в БКНФ, если оно находится в 3НФ и детерминанты всех функциональных зависимостей являются потенциальными ключами.

4НФ – отношение находится в 4НФ, если оно находится в БКНФ и не содержит нетривиальных многозначных зависимостей.

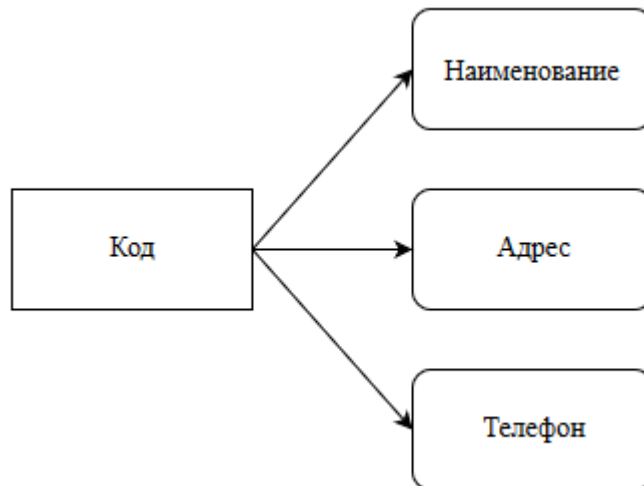
5НФ – отношение находится в 5НФ, если оно находится в 4НФ и любая многозначная зависимость соединения в ней является тривиальной. Пятая нормальная форма в большей степени является теоретическим исследованием и практически не применяется при реальном проектировании баз данных.

ДКНФ – отношение находится в ДКНФ, если не имеет аномалий модификации. Другими словами, что бы ни менялось — ничего не потеряется, если соблюдены все ограничения относительно ключей и доменов. Формулировка слишком общая, но суть ее заключается в том, что если выполнять некоторые правила, то при любых действиях с таблицей ее целостность не пострадает и вся необходимая информация сохранится.

Более высокая нормальная форма не всегда предпочтительней в процессе проектирования базы данных, так как чем выше нормальная форма, тем больше таблиц в базе данных и тем больше потребуется выполнить операций соединения для получения окончательного результата. Следовательно, тем медленней информационная система будет реагировать на запросы пользователя. В данной курсовой работе база данных находится в нормальной форме Бойса-Кодда.

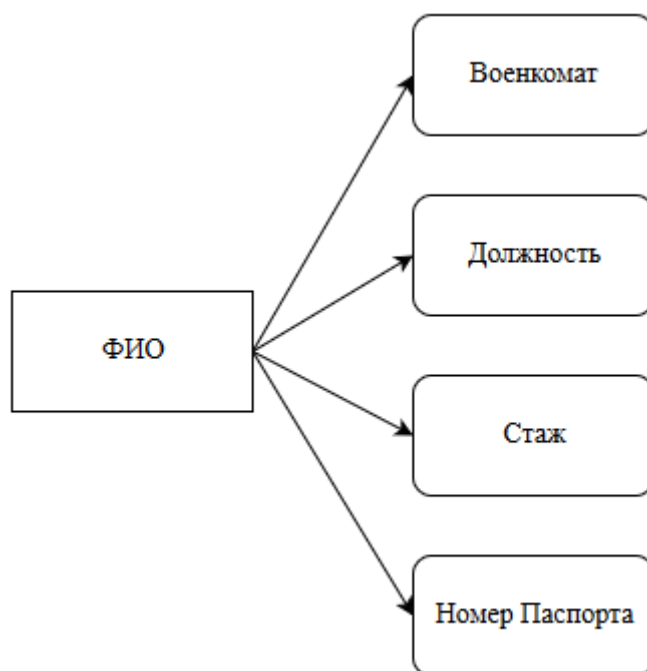
Военкомат (Код, Наименование, Адрес, Телефон):

- Отношение находится в 1НФ, так как на пересечении каждой строки и столбца находится одно значение.
- Отношение находится в 2НФ, так как первичный ключ является простым;
- Отношение находится в 3НФ, так как отсутствует транзитивная зависимость;
- Отношение находится в БКНФ, так как первичный ключ является детерминантом всех функциональных зависимостей.



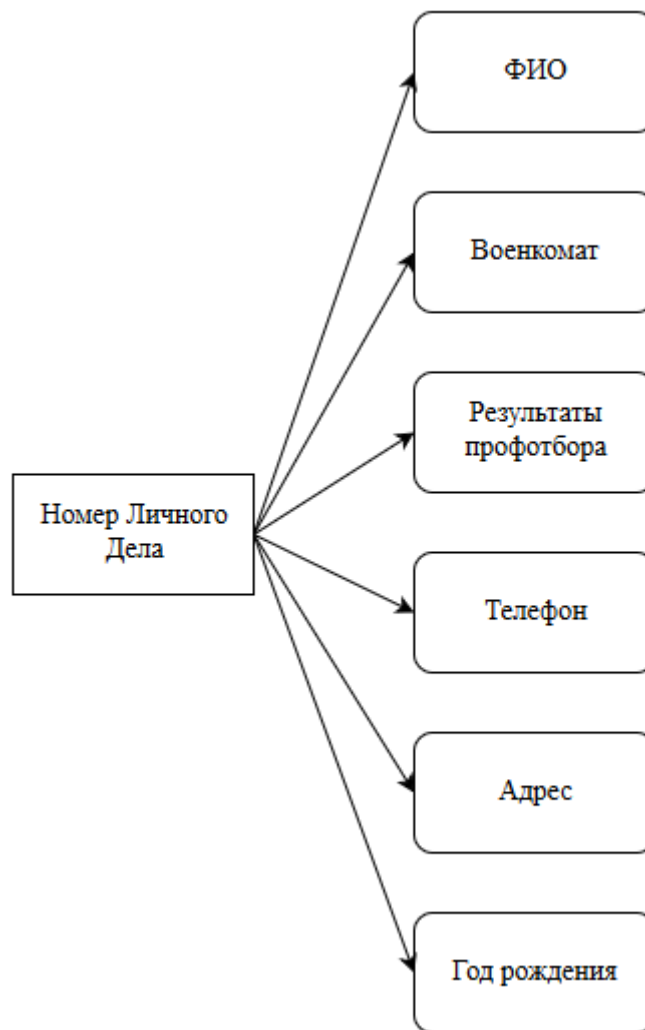
Сотрудник (ФИО, Военкомат, Должность, Стаж, Номер Паспорта):

- Отношение находится в 1НФ, так как на пересечении каждой строки и столбца находится одно значение.
- Отношение находится в 2НФ, так как первичный ключ является простым;
- Отношение находится в 3НФ, так как отсутствует транзитивная зависимость;
- Отношение находится в БКНФ, так как первичный ключ является детерминантом всех функциональных зависимостей.



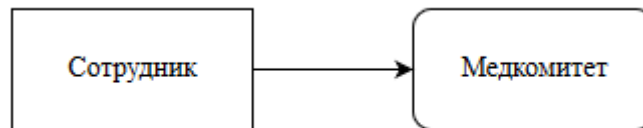
Призывник (Личное Дело, ФИО, Военкомат, Группа профессионального отбора, Телефон, Адрес, Год Рождения):

- Отношение находится в 1НФ, так как на пересечении каждой строки и столбца находится одно значение.
- Отношение находится в 2НФ, так как первичный ключ является простым;
- Отношение находится в 3НФ, так как отсутствует транзитивная зависимость;
- Отношение находится в БКНФ, так как первичный ключ является детерминантом всех функциональных зависимостей.



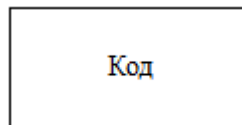
Состав Медкомиссии (Сотрудник, Медкомитет):

- Отношение находится в 1НФ, так как на пересечении каждой строки и столбца находится одно значение.
- Отношение находится в 2НФ, так как первичный ключ является простым;
- Отношение находится в 3НФ, так как отсутствует транзитивная зависимость;
- Отношение находится в БКНФ, так как первичный ключ является детерминантом всех функциональных зависимостей.



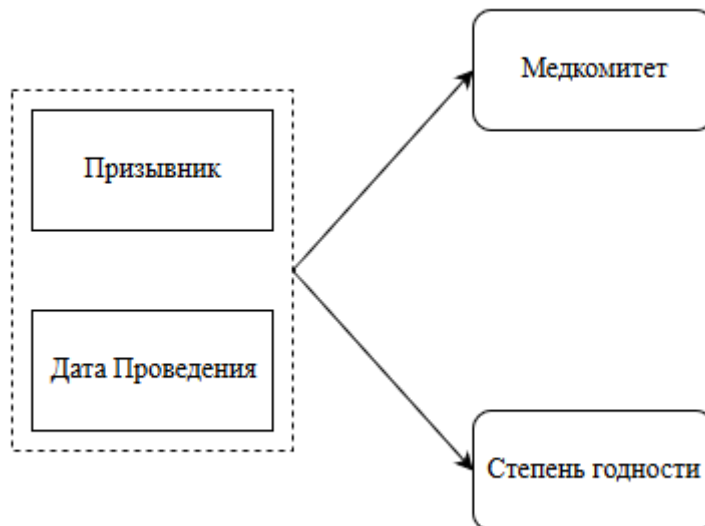
Медкомитет (Код):

- Отношение находится в 1НФ, так как на пересечении каждой строки и столбца находится одно значение.
- Отношение находится в 2НФ, так как первичный ключ является простым;
- Отношение находится в 3НФ, так как отсутствует транзитивная зависимость;
- Отношение находится в БКНФ, так как первичный ключ является детерминантом всех функциональных зависимостей.



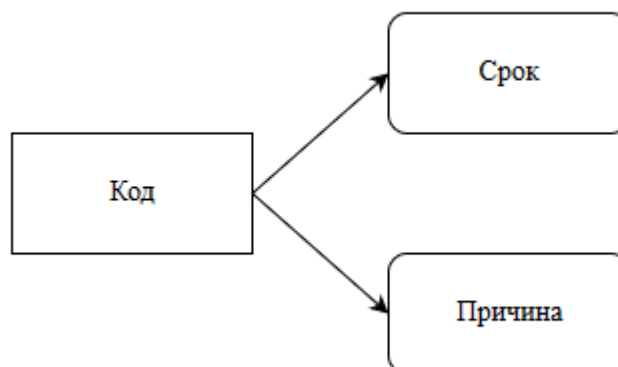
**Заключение Врачебной комиссии (Призывник, Дата Проведения,
Медкомитет, Степень годности):**

- Отношение находится в 1НФ, так как на пересечении каждой строки и столбца находится одно значение.
- Отношение находится в 2НФ, так как первичный ключ является простым;
- Отношение находится в 3НФ, так как отсутствует транзитивная зависимость;
- Отношение находится в БКНФ, так как первичный ключ является детерминантом всех функциональных зависимостей.



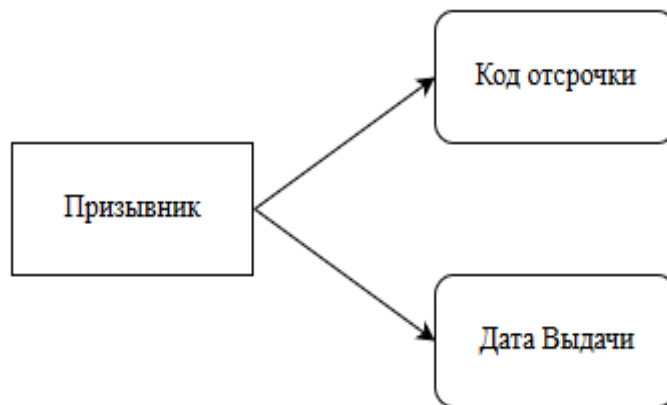
Отсрочка (Код, Срок, Описание):

- Отношение находится в 1НФ, так как на пересечении каждой строки и столбца находится одно значение.
- Отношение находится в 2НФ, так как не ключевые атрибуты функционально полно зависят от первичного ключа.
- Отношение находится в 3НФ, так как отсутствует транзитивная зависимость;
- Отношение находится в БКНФ, так как первичный ключ является детерминантом всех функциональных зависимостей.



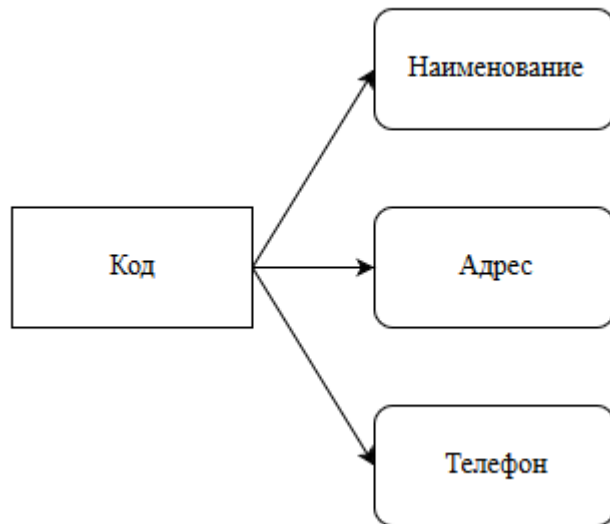
Отсрочка призывника (Призывник, Код отсрочки, Дата Выдачи):

- Отношение находится в 1НФ, так как на пересечении каждой строки и столбца находится одно значение.
- Отношение находится в 2НФ, так как первичный ключ является простым;
- Отношение находится в 3НФ, так как отсутствует транзитивная зависимость;
- Отношение находится в БКНФ, так как первичный ключ является детерминантом всех функциональных зависимостей.



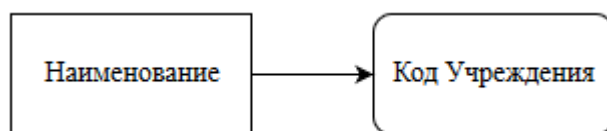
Специальное учреждение (Код, Наименование, Адрес, Телефон):

- Отношение находится в 1НФ, так как на пересечении каждой строки и столбца находится одно значение.
- Отношение находится в 2НФ, так как первичный ключ является простым;
- Отношение находится в 3НФ, так как отсутствует транзитивная зависимость;
- Отношение находится в БКНФ, так как первичный ключ является детерминантом всех функциональных зависимостей.



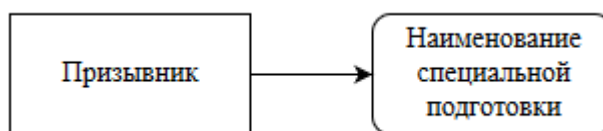
Специальная подготовка (Наименование, Код Учреждения):

- Отношение находится в 1НФ, так как на пересечении каждой строки и столбца находится одно значение.
- Отношение находится в 2НФ, так как первичный ключ является простым;
- Отношение находится в 3НФ, так как отсутствует транзитивная зависимость;
- Отношение находится в БКНФ, так как первичный ключ является детерминантом всех функциональных зависимостей.



**Направление на специальную подготовку (Призывник,
Наименование специальной подготовки):**

- Отношение находится в 1НФ, так как на пересечении каждой строки и столбца находится одно значение.
- Отношение находится в 2НФ, так как первичный ключ является простым;
- Отношение находится в 3НФ, так как отсутствует транзитивная зависимость;
- Отношение находится в БКНФ, так как первичный ключ является детерминантом всех функциональных зависимостей.



Профессиональный отбор (Группа, Описание):

- Отношение находится в 1НФ, так как на пересечении каждой строки и столбца находится одно значение.
- Отношение находится в 2НФ, так как первичный ключ является простым;
- Отношение находится в 3НФ, так как отсутствует транзитивная зависимость;
- Отношение находится в БКНФ, так как первичный ключ является детерминантом всех функциональных зависимостей.

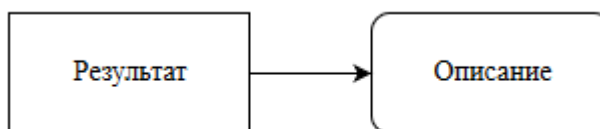


Схема базы данных «Система учета призывников» представлена на рисунке 5.2.

Схема базы данных «Система Учета Призывников»

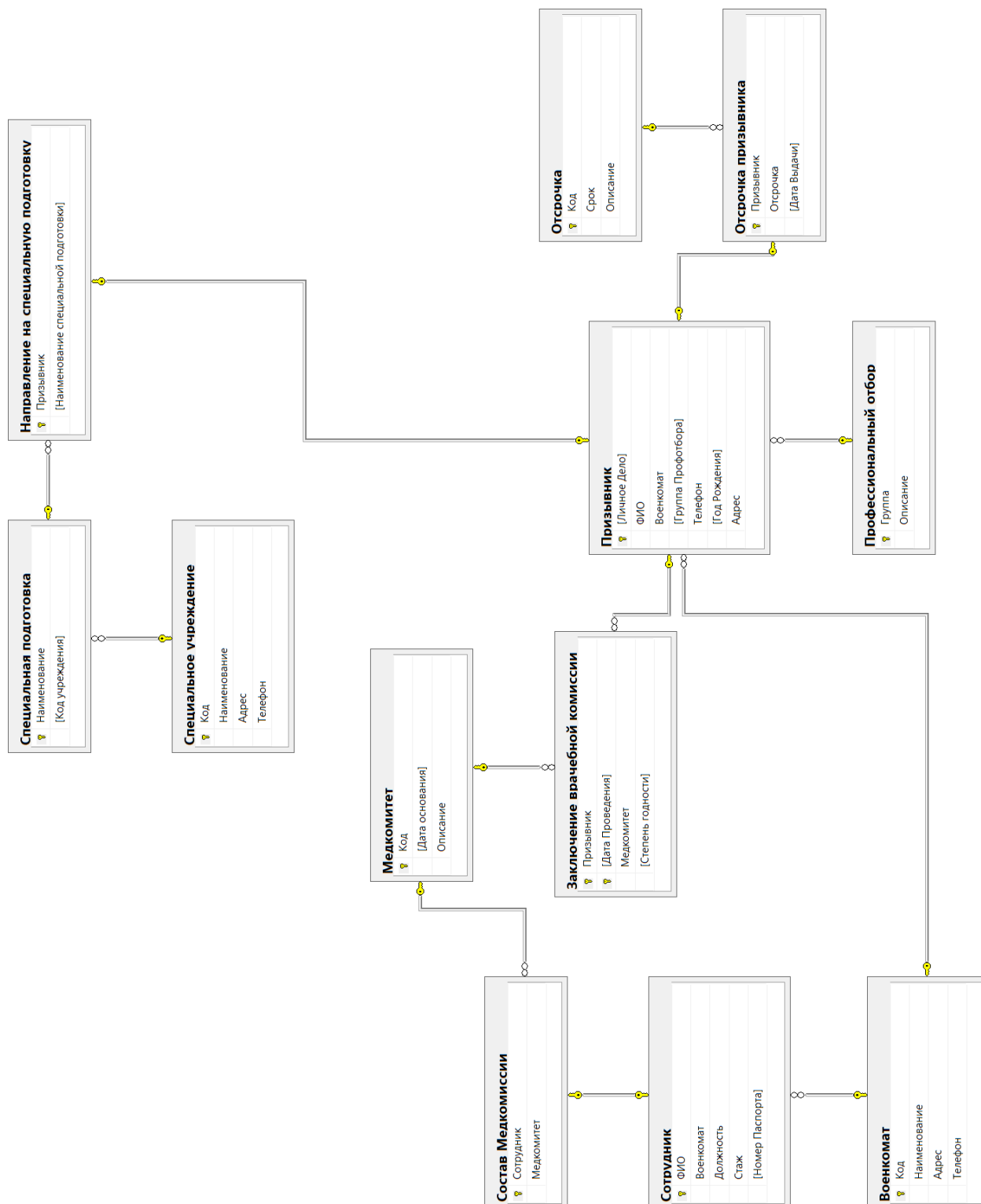


Рисунок 5.2 - схема базы данных «Система Учета Призывников»

5.3 Программирование объектов БД

Представления или просмотры – это виртуальные таблицы, информация в которых не хранится постоянно как в базовых таблицах, а формируется динамически при обращении к представлению. Использование представлений позволяет обеспечить каждому пользователю наиболее подходящий способ работы с данными, что решает проблему простоты их использования и безопасности.

Представление – хранимый запрос, который выполняется всякий раз при участии представления в какой-либо команде.

Представления:

Представление «Результаты Последнего медосмотра», которое хранит информацию о последнем медосмотре представлено на рисунке 5.3.

	Призывник	ФИО	Степень годности	Дата Проведения	Медкомитет
1	1	Иваненко Андрей Викторович	B3	2018-05-04 00:00:00.00000000	4
2	2	Карасев Никита Андреевич	B2	2018-05-07 00:00:00.00000000	4
3	3	Пименов Антон Владимирович	A1	2018-03-07 00:00:00.00000000	4
4	4	Тарасенко Максим Петрович	A2	2019-11-09 00:00:00.00000000	5
5	5	Феофанов Феофан Зингирович	A3	2019-09-09 00:00:00.00000000	3
6	6	Арутюнян Артур Игоревич	A2	2019-03-10 00:00:00.00000000	1
7	7	Григоренко Родион Викторович	B3	2014-01-10 00:00:00.00000000	2
8	8	Кариенко Иван Антонович	G2	2012-11-10 00:00:00.00000000	3
9	9	Шамилов Игнат Николаевич	A1	2013-12-10 00:00:00.00000000	4
10	10	Сидоров Николай Николаевич	A1	2019-12-10 00:00:00.00000000	5

Рисунок 5.3 - представление «Результаты Последнего медосмотра»,

Представление «Все О Призывнике», которое хранит всю информацию о Призывнике представлено на рисунке 5.3.1.

Личное Де...	ФИО	Группа Профоб...	Год Рожден...	Степень годно...	Дата выдачи отсрочки	Срок Отсро...	Наименование специальной подготовки	Военный комиссариат
1	Иваненко Андрей Викторович	1	2001	B3	2018-04-12 00:00:00.00000000	1	Курсы по вождению гусеничной техники (гусеничные тя...	Военный комиссариат Октябрьского и Советского райо...
2	Карасев Никита Андреевич	1	1999	B2	2018-08-12 00:00:00.00000000	1	Перехватная подготовка	Военный комиссариат Октябрьского и Советского райо...
3	Пименов Антон Владимирович	1	1993	A1	2015-09-07 00:00:00.00000000	1	NULL	Военный комиссариат Октябрьского и Советского райо...
4	Тарасенко Максим Петрович	2	2002	A2	NULL	NULL	Получения водительского удостоверения категории С+E	Военный комиссариат Московского и Железнодорожно...
5	Феофанов Феофан Зингирович	2	1992	A3	NULL	NULL	Получения водительского удостоверения категории В+E	Военный комиссариат Московского и Железнодорожно...
6	Арутюнян Артур Игоревич	1	1998	A2	NULL	NULL	Обучение управлению спортивными самолётами и сан...	Военный комиссариат Рязанской области
7	Григоренко Родион Викторович	3	1996	B3	NULL	NULL	Получения водительского удостоверения категории С+E	Военный комиссариат по Рязанскому и Спасскому рай...
8	Кариенко Иван Антонович	4	1997	G2	2012-12-06 00:00:00.00000000	1	Получения водительского удостоверения категории С	Военный комиссариат Рязанской области
9	Шамилов Игнат Николаевич	4	2001	A1	NULL	NULL	NULL	Военный комиссариат Рыбновского района Рязанской ...
10	Сидоров Николай Николаевич	4	1999	A1	NULL	NULL	NULL	Сборный пункт военного комиссариата Рязанской обл...

Рисунок 5.3.1 - представление «Все О Призывнике»

Представление «Годные к призыву», которое хранит всю информацию о Призывнике годного к призыву представлено на рисунке 5.3.2.

	Личное Де...	ФИО	Год Рожден...	Степень годно...	Группа Профотб...	Наименование специальной подготовки	Военкомат
1	4	Тарасенко Максим Петрович	2002	A2	2	Получения водительского удостоверения категории С+Е	Военный комиссариат Московского и Железнодорожно...
2	5	Феофанов Феофан Зингирович	1992	A3	2	Получения водительского удостоверения категории В+Е	Военный комиссариат Московского и Железнодорожно...
3	6	Арутюнян Артур Игоревич	1998	A2	1	Обучение управлению спортивными самолетами и сам...	Военный комиссариат Рязанской области

Рисунок 5.3.2 - представление «Годные к призыву»,

Хранимая процедура представляет собой последовательность операторов языка SQL, которые хранятся в базе данных.

Хранимые процедуры:

Процедура «Получить Призывника», которая по заданному номеру личного дела призывника выводит информацию о призывнике;

Процедура «Получить Призывников Военкомата», которая по заданному коду военкомата выводит информацию о призывниках приписанных к этому военкомату;

Процедура «Добавить Призывника», которая позволяет добавить информацию о призывнике, в качестве входных параметров процедура принимает:

- Номер личного дела призывника;
- ФИО призывника;
- Военкомат, к которому приписан призывник;
- Группу профотбора призывника;
- Телефон призывника;
- Год рождения призывника;
- Адрес призывника;
- Медкомитет, который осмотрел призывника;
- Степень годности призывника;
- Дата Медицинского осмотра;

Процедура «Добавить Сотрудника», которая позволяет добавить информацию о Сотруднике, в качестве входных параметров процедура принимает:

- ФИО;
- Военкомат;
- Должность;
- Стаж;
- Номер паспорта;

Процедура «Добавить Сотрудника в состав медкомиссии», которая позволяет добавить сотрудника в состав медкомиссии, в качестве входных параметров процедура принимает:

- Сотрудник;
- Медкомитет;

Процедура «Добавить Призывнику Отсрочку», которая позволяет добавить призывнику информацию об отсрочке, в качестве входных параметров процедура принимает:

- Номер личного дела призывника;
- Код отсрочки;
- Дата получения отсрочки;

Процедура «Обновление медкомитета сотруднику, состоящему в медкомиссии», которая позволяет обновить медкомитет сотруднику, который состоит в медкомиссии, в качестве входных параметров процедура принимает:

- Сотрудник;
- Медкомитет;

Процедура «Обновить должность сотруднику», которая позволяет обновить должность сотруднику, в качестве входных параметров процедура принимает:

- Сотрудник;
- Должность;

Процедура «Обновить Призывнику Отсрочку», которая позволяет обновить призывнику информацию об отсрочке, в качестве входных параметров процедура принимает:

- Номер личного дела призывника;
- Код отсрочки;
- Дата получения отсрочки.

Процедура «Удалить Призывнику Отсрочку», которая позволяет удалить призывнику отсрочку, в качестве входных параметров процедура принимает:

- Номер личного дела призывника;

Процедура «Удалить Сотрудника», которая позволяет удалить сотрудника, в качестве входных параметров процедура принимает:

- ФИО Сотрудника;

Процедура «Удаление сотрудника из состава медкомиссии», которая позволяет удалить сотрудника из медкомиссии, в качестве входных параметров процедура принимает:

- ФИО Сотрудника;

Триггер – это специальный тип хранимой процедуры, который активизируется при выполнении определенных действий над БД. Таким действием может быть вставка, удаление и обновление данных. С помощью триггеров обеспечивается целостность данных в базе. Хотя триггер и является разновидностью хранимой процедуры, вызвать его непосредственно нельзя. Он реагирует только на то событие, для которого он определен. Триггер выполняется как единое целое, т.е транзакция, которая либо фиксируется, либо откатывается.

Триггеры:

Триггер «ТриггерСтаж», который запрещает уменьшение стажа сотрудника;

6 РАЗРАБОТКА КЛИЕНТСКОГО ПРИЛОЖЕНИЯ

6.1 Выбор программных компонентов клиентской части

Разработка клиентской части информационной системы производилась в среде Microsoft Visual Studio 2019. В качестве языка программирования был выбран язык высокого уровня C#.

6.2 Разработка интерфейса пользователя

Интерфейс пользователя представляет собой набор компонентов, предоставляемых средой разработки Microsoft Visual Studio 2019. При проектировании интерфейса учитывались такие параметры как: удобство расположение главных элементов на форме, понятность функционирования элементов, а также визуальная составляющая.

6.2.1 Разработка форм

После запуска приложения, пользователю представлена главная форма программы, которая представлена на рисунке 6.2.1

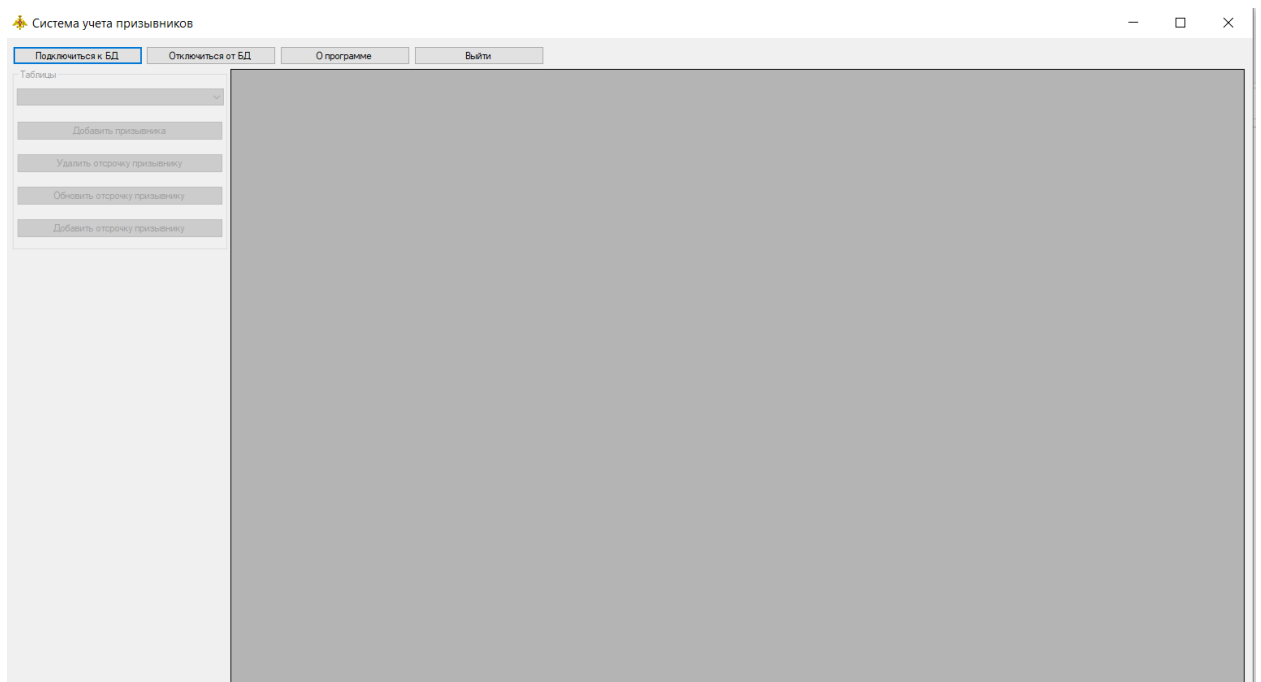


Рисунок 6.2.1 – главная форма программы

Основные элементы главной формы:

- Кнопка соединения с базой данных
- Кнопка отсоединения от базы данных
- Кнопка вызова информации о программе
- Кнопка выход
- Компонент groupBox, который не активен до момента соединения с базой данных, внутри данного компонента расположены кнопки вызова форм, для работы с данными, а также компонент comboBox, для выбора отображения информации из необходимой таблицы.
- Компонент dataGridView, в котором отображаются выбранные таблицы.

Работа с приложением начинается с нажатия кнопки «Подключиться к БД», данная кнопка вызывает диалоговое окно для подключения к базе данных. Данный вариант создания подключения является наиболее простым и в то же время гибким. Диалоговое окно представлено на рисунке 6.2.1.1

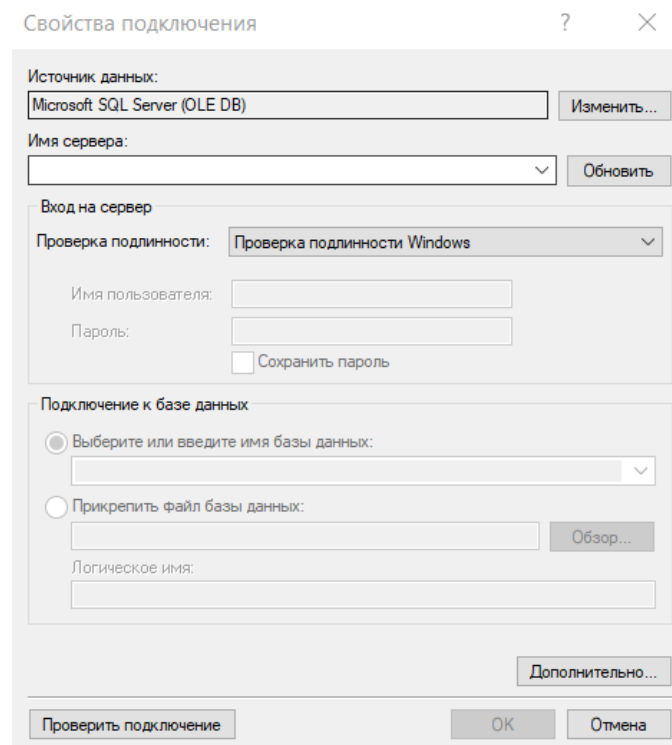


Рисунок 6.2.1.1 – диалоговое окно подключения к базе данных

При успешном подключении станет активным компонент groupBox и отобразится информация из главного представления «Все О Призывнике», а также всплывет окно, в котором программа уведомит пользователя об успешном подключении. Успешное подключение представлено на рисунке 6.2.1.2

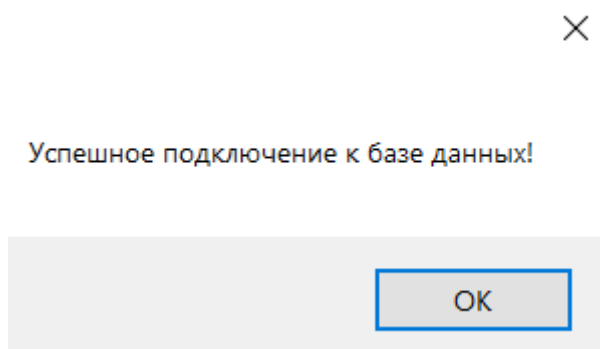


Рисунок 6.2.1.2 – Успешное подключение к базе данных

В случае неуспешного подключения к базе данных программа так же уведомит об этом пользователя при помощи всплывающего окна, которое представлено на рисунке 6.2.1.3

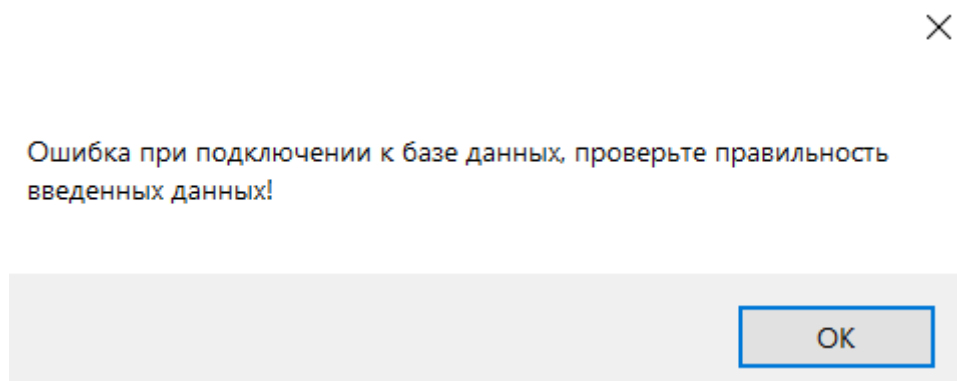


Рисунок 6.2.1.3 – неудачное подключение к базе данных

При нажатии кнопки закрыть соединение, будет показано всплывающее окно, которое представлено на рисунке 6.2.1.4.

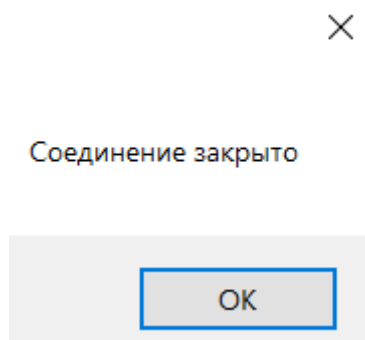


Рисунок 6.2.1.4 – всплывающее окно при закрытии соединения с базой данных

Кнопка «Добавить призывника» открывает форму, для заполнения необходимых данных о призывнике, окно формы представлено на рисунке 6.2.1.5.

A window titled 'Добавление призывника' (Add conscript) with a standard Windows title bar. The form is divided into two columns. The left column contains: 'Личное дело:' (text input), 'ФИО:' (text input), 'Военкомат:' (dropdown menu), 'Группа профотбора:' (dropdown menu), and 'Степень годности:' (dropdown menu). The right column contains: 'Телефон:' (text input), 'Год Рождения:' (text input), 'Адрес:' (text input), 'Медкомитет:' (dropdown menu), and 'Дата медосмотра:' (text input). At the bottom center is a button labeled 'Добавить призывника'.

Рисунок 6.2.1.5 – форма добавление призывника

Кнопка «Удалить отсрочку призывнику» открывает форму, для заполнения личного дела призывника, на рисунке 6.2.1.6 представлена данная форма.

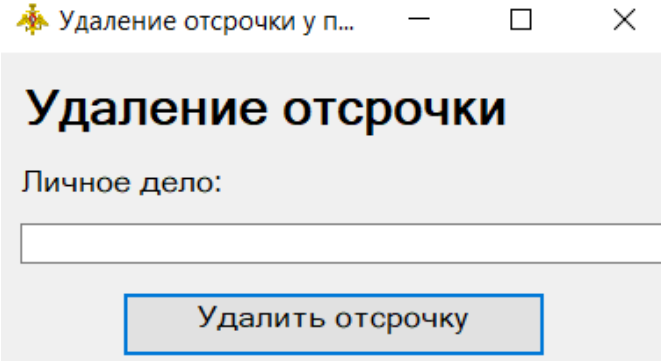
The image shows a software window titled 'Удаление отсрочки у п...' (Deletion of exemption for a conscript...). The window has a standard Windows title bar with a minimize button, a maximize button, and a close button. The main content area has a title 'Удаление отсрочки' (Deletion of exemption) in bold black text. Below the title is a label 'Личное дело:' (Personal case:). Underneath the label is a single-line text input field. At the bottom of the form is a button with the text 'Удалить отсрочку' (Delete exemption).

Рисунок 6.2.1.6 – форма удаления отсрочки у призывника

Кнопка «Обновить отсрочку призывнику» открывает форму, для заполнения необходимых данных о призывнике, для обновления его отсрочки, на рисунке 6.2.1.7 представлена данная форма.

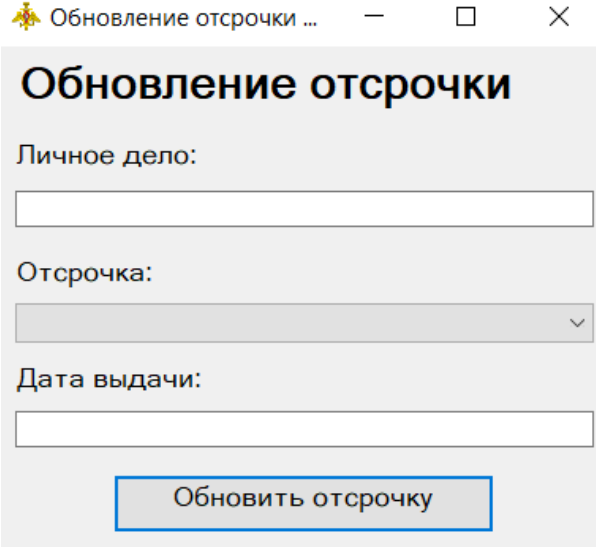
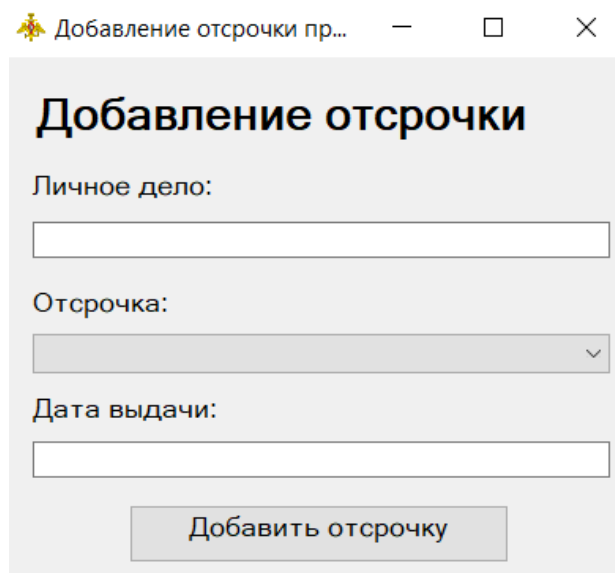
The image shows a software window titled 'Обновление отсрочки ...' (Update exemption ...). The window has a standard Windows title bar with a minimize button, a maximize button, and a close button. The main content area has a title 'Обновление отсрочки' (Update exemption) in bold black text. Below the title is a label 'Личное дело:' (Personal case:). Underneath the label is a single-line text input field. Below this is another label 'Отсрочка:' (Exemption:). Underneath this label is a dropdown menu with a downward arrow icon. Below the dropdown is a label 'Дата выдачи:' (Date of issue:). Underneath this label is a single-line text input field. At the bottom of the form is a button with the text 'Обновить отсрочку' (Update exemption).

Рисунок 6.2.1.7 – форма обновления отсрочки у призывника

Кнопка «Добавить отсрочку призывнику» открывает форму, для заполнения необходимых данных о призывнике, для добавления его отсрочки, на рисунке 6.2.1.8 представлена данная форма.



Добавление отсрочки призывнику

Личное дело:

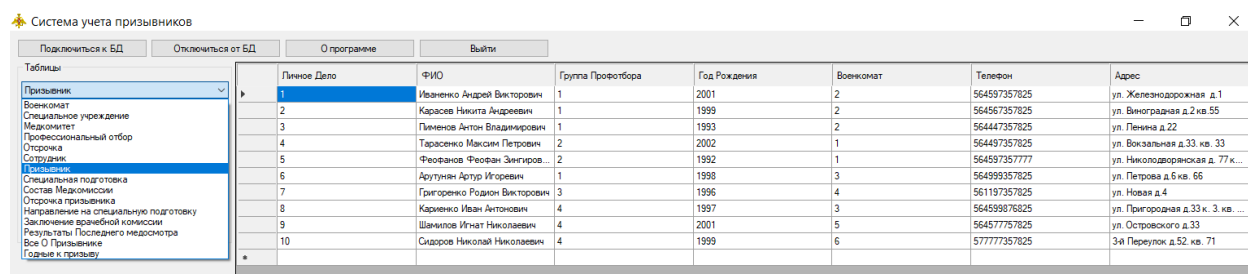
Отсрочка:

Дата выдачи:

Добавить отсрочку

Рисунок 6.2.1.8 – форма добавления отсрочки у призывника

Выбор необходимой таблицы для вывода из нее информации представлен на рисунке 6.2.1.9.



Личное Дело	ФИО	Группа Проведения	Год Рождения	Военкомат	Телефон	Адрес
1	Иваненко Андрей Викторович	1	2001	2	564597357825	ул. Железнодорожная д.1
2	Карасев Никита Андреевич	1	1999	2	564597357825	ул. Виноградная д.2 кв.55
3	Павленко Антон Владимирович	1	1993	2	564447357825	ул. Пенная д.22
4	Тарасенко Максим Петрович	2	2002	1	564497357825	ул. Вокзальная д.33 кв.33
5	Феосанов Феофан Зингирович	2	1992	1	564597357777	ул. Николодворская д.77 кв...
6	Артунян Артур Игоревич	1	1998	3	564599357825	ул. Петрова д.6 кв.66
7	Григоренко Родион Викторович	3	1996	4	561197357825	ул. Новая д.4
8	Карпенко Иван Антонович	4	1997	3	564599876825	ул. Пригородная д.33 кв.3 кв...
9	Шампилов Игнат Николаевич	4	2001	5	564577757825	ул. Островского д.33
10	Сидоров Николай Николаевич	4	1999	6	577777357825	Зна Переулок д.52 кв.71

Рисунок 6.2.1.9 – выбор таблицы для вывода из нее информации

6.3 Разработка сценария инсталляции клиентской программы

Для создания установочного файла использовалась бесплатно распространяемая программа «Smart Install Maker 5.04». Рассмотрим основные этапы создания установщика.

Задание основных характеристик инсталлятора представлено на рисунке 6.3.

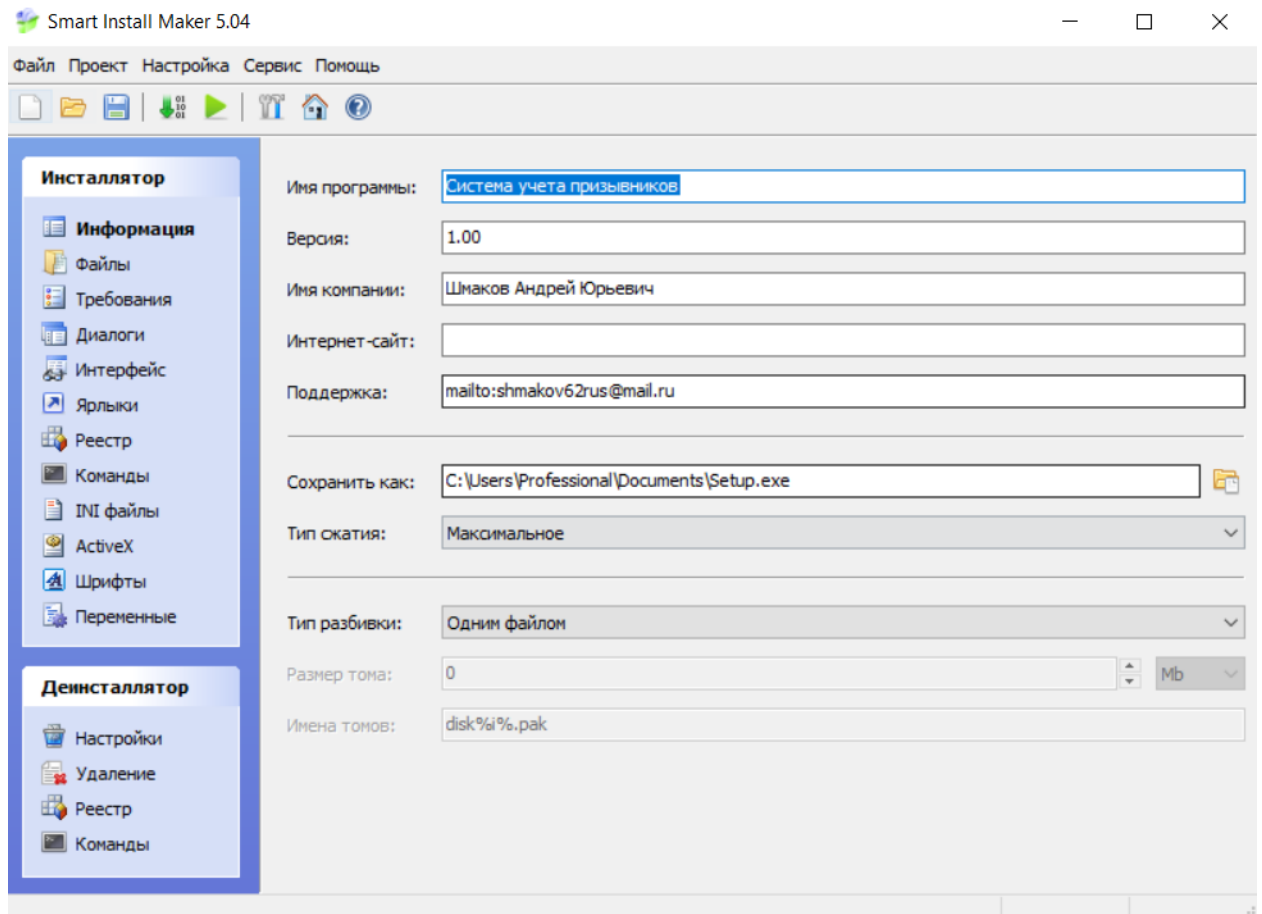


Рисунок 6.3 – основные характеристики инсталлятора

Добавление всех необходимых файлов представлено на рисунке 6.3.1.

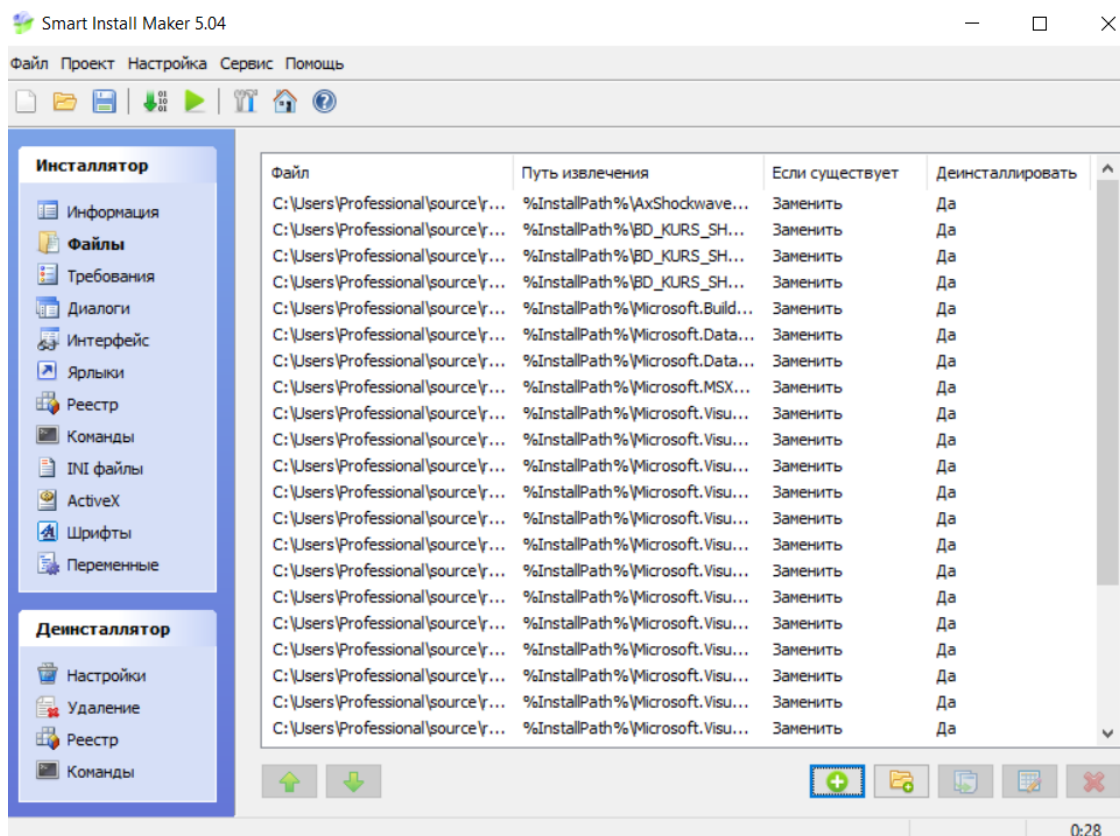


Рисунок 6.3.1 – добавление необходимых файлов в инсталлятор

После успешной компиляции запускается инсталлятор программы, который представлен на рисунке 6.3.2.

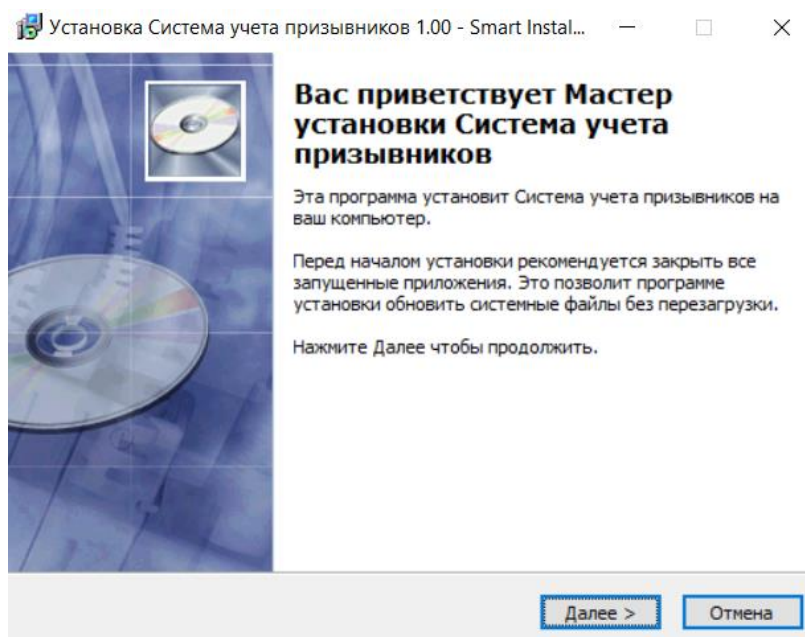


Рисунок 6.3.2 – инсталлятор приложения

Выбор пути для установки приложения представлен на рисунке 6.3.3.

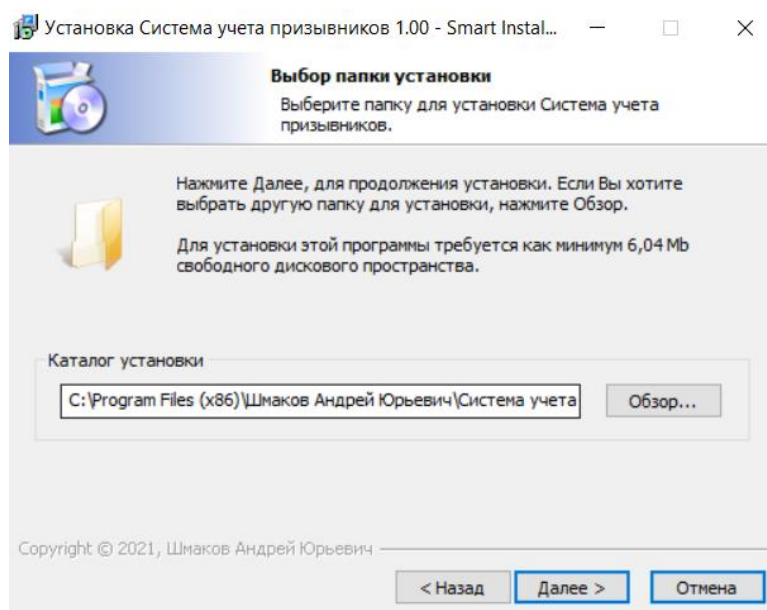


Рисунок 6.3.3 – выбор пути установки приложения

Сведения об устанавливаемом приложении представлены на рисунке 6.3.4.

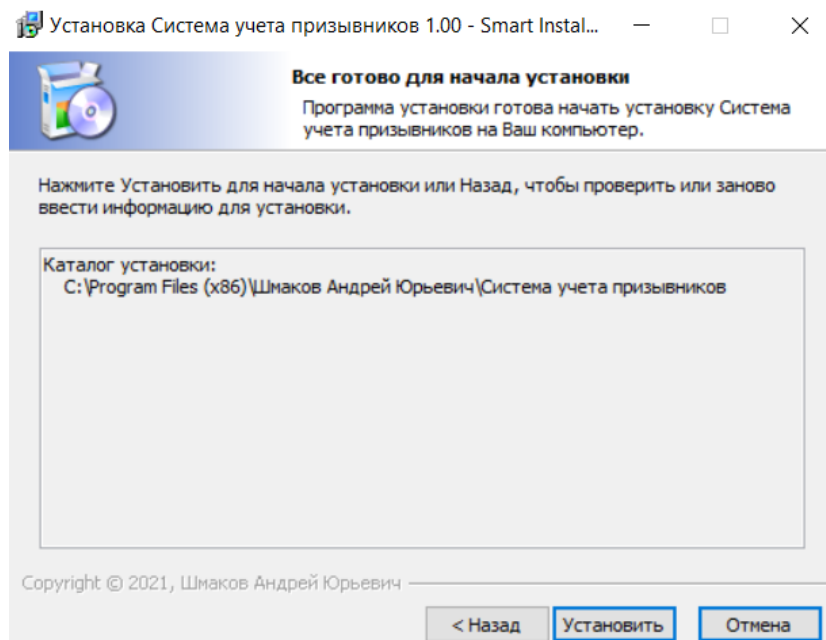


Рисунок 6.3.4 – сведения об устанавливаемом приложении

После успешной установки приложения инсталлятор уведомит об этом пользователя, успешная установка представлена на рисунке 6.3.5

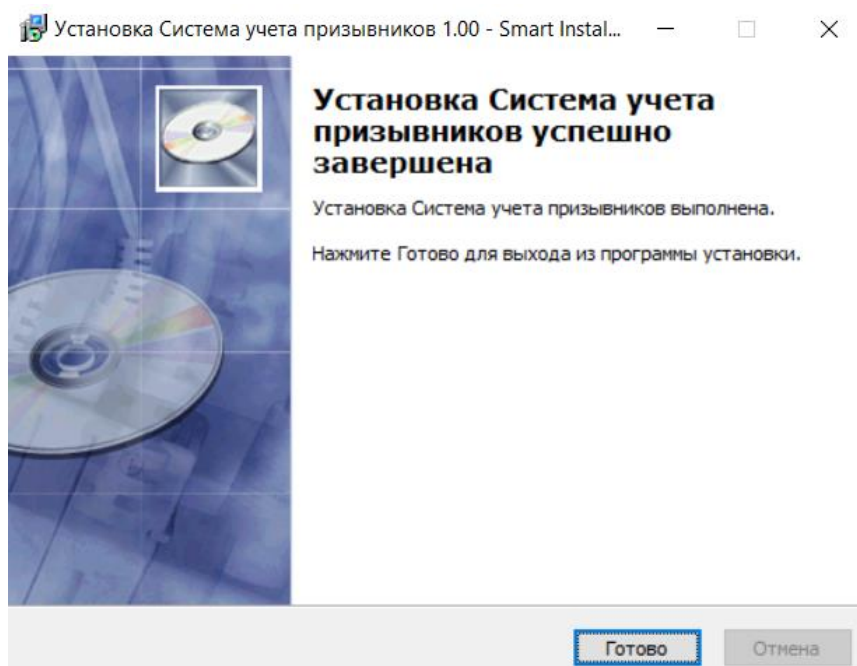


Рисунок 6.3.5 – успешная установка приложения

Пакет установщика приложения кроме необходимых файлов создает еще деинсталлятор, который представлен на рисунке 6.3.6

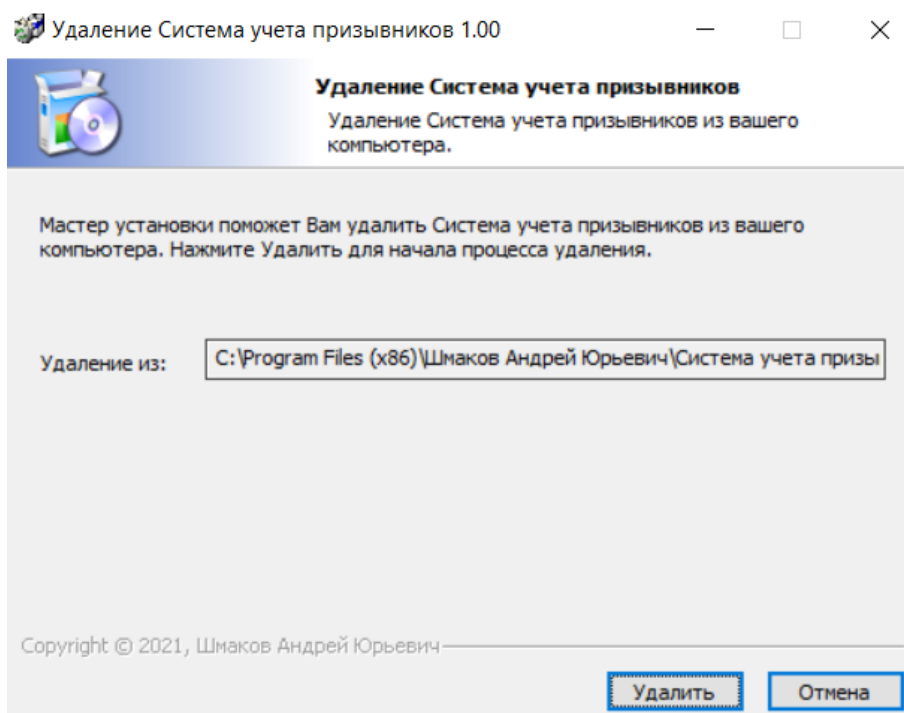


Рисунок 6.3.6 – деинсталлятор приложения

Успешное удаление приложения при помощи деинсталлятора представлено на рисунке 6.3.7

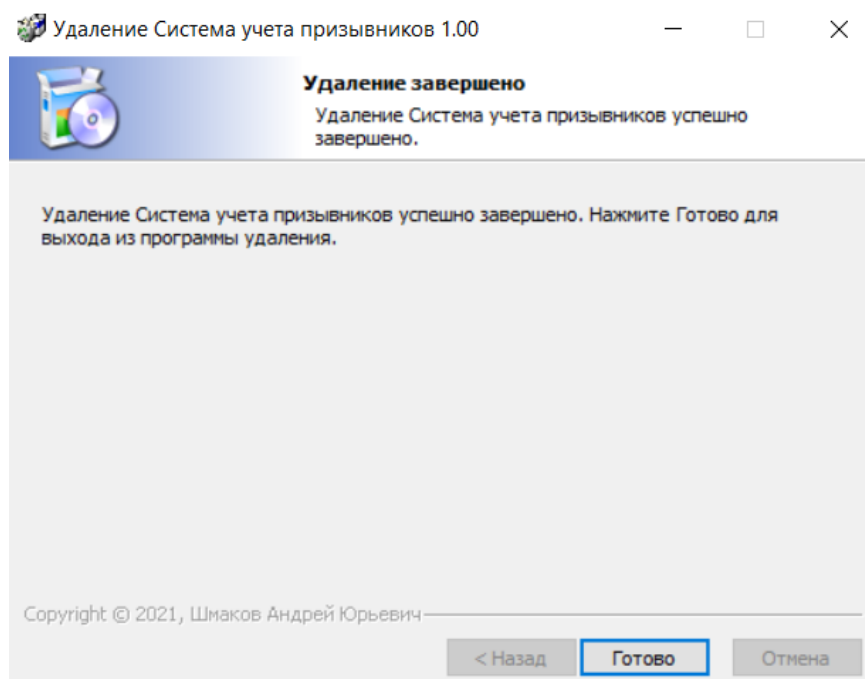


Рисунок 6.3.6 – успешное удаление приложения

Таким образом был создан полностью функционально рабочий пакет установщик для разработанного приложения.

7. РАЗРАБОТКА ПРОГРАММНОЙ ДОКУМЕНТАЦИИ

8. ТЕСТИРОВАНИЕ ИС

ЗАКЛЮЧЕНИЕ

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Дейт К. Дж.: Введение в системы баз данных /Пер. с англ. [Текст]
/ Дейт К. Дж.; -М.: Издательский дом «Вильямс», 2001. – 1072с.:
Библиогр.: с.4
2. Райордан Р. :Основы реляционных баз данных/Пер. с англ. [Текст]
/ Райордан Р. ; — М.: Издательско-торговый дом «Русская
Редакция», 2001. — 384 с. ; Библиогр.: с.6
3. Хомоненко А.Д., Циганков В.М., Мальцев М.Г.:Базы
данных:Учебник для высших учебных заведений. [Текст] /Под
ред. проф. Хомоненко А.Д.; - СПб:КОРОНА принт, 2004. – 736 с.;
Библиогр.: с. 2

ПРИЛОЖЕНИЕ А: СЦЕНАРИЙ СОЗДАНИЯ ОБЪЕКТОВ БД.

```
USE master

CREATE DATABASE СистемаУчетаПризывников

GO

USE СистемаУчетаПризывников

GO

-- Пользовательские типы:
CREATE TYPE PHONE FROM NVARCHAR(12);
GO
CREATE TYPE PASSPORT FROM NVARCHAR(10);
GO
CREATE TYPE DEGOFsuitab FROM NVARCHAR(2) NOT NULL;
GO

-- Таблица [Военкомат]:
CREATE TABLE [Военкомат]
(
    Код INT PRIMARY KEY,
    Наименование NVARCHAR(100) NOT NULL,
    Адрес NVARCHAR(200) NOT NULL,
    Телефон PHONE,
)
GO

-- Таблица [Специальное учреждение]:
CREATE TABLE [Специальное учреждение]
(
    Код INT PRIMARY KEY,
    Наименование NVARCHAR(100) NOT NULL,
    Адрес NVARCHAR(200),
    Телефон PHONE,
)
GO

-- Таблица [Медкомитет]:
CREATE TABLE [Медкомитет]
(
    Код INT PRIMARY KEY,
    [Дата основания] DATETIME2,
    Описание NVARCHAR(50)
)
GO

-- Таблица [Профессиональный отбор]:
CREATE TABLE [Профессиональный отбор]
(
    Группа INT PRIMARY KEY,
    Описание NVARCHAR(100),
)
GO

-- Таблица [Отсрочка]:
CREATE TABLE [Отсрочка]
(
    Код INT PRIMARY KEY,
    Срок INT NOT NULL,
```

```

        Описание NVARCHAR(50)
    )
GO

-- Таблица [Сотрудник]:
CREATE TABLE [Сотрудник]
(
    ФИО NVARCHAR(100) PRIMARY KEY,
    Военкомат INT NOT NULL,
    Должность NVARCHAR(30),
    Стаж INT,
    [Номер Паспорта] PASSPORT
)
GO

-- Таблица [Призывник]:
CREATE TABLE [Призывник]
(
    [Личное Дело] INT PRIMARY KEY,
    ФИО NVARCHAR(100) NOT NULL,
    Военкомат INT NOT NULL,
    [Группа Профотбора] INT NOT NULL,
    Телефон PHONE,
    [Год Рождения] INT,
    Адрес NVARCHAR(200),
)
GO

-- Таблица [Специальная подготовка]:
CREATE TABLE [Специальная подготовка]
(
    Наименование NVARCHAR(100) PRIMARY KEY,
    [Код учреждения] INT NOT NULL,
)
GO

-- Таблица [Состав Медкомиссии]:
CREATE TABLE [Состав Медкомиссии]
(
    Сотрудник NVARCHAR(100) PRIMARY KEY,
    Медкомитет INT NOT NULL,
)
GO

-- Таблица [Отсрочка призывника]:
CREATE TABLE [Отсрочка призывника]
(
    Призывник INT PRIMARY KEY,
    Отсрочка INT NOT NULL,
    [Дата Выдачи] DATETIME2
)
GO

-- Таблица [Направление на специальную подготовку]:
CREATE TABLE [Направление на специальную подготовку]
(
    Призывник INT PRIMARY KEY,

```

```

        [Наименование специальной подготовки] NVARCHAR(100) NOT NULL,
    )

GO

-- Таблица [Заключение врачебной комиссии]:
CREATE TABLE [Заключение врачебной комиссии]
(
    Призывник INT NOT NULL,
    [Дата Проведения] DATETIME2 NOT NULL,
    Медкомитет INT NOT NULL,
    [Степень годности] DEGOFSUITAB,
)
GO

-- Ограничения на уникальность:
ALTER TABLE Военкомат ADD CONSTRAINT Военкомат_unique UNIQUE (Телефон, Наименование)
ALTER TABLE [Специальное учреждение] ADD CONSTRAINT СпециальноеУчреждение_unique UNIQUE
(Телефон, Наименование)
ALTER TABLE [Профессиональный отбор] ADD CONSTRAINT [ПрофессиональныйОтбор_unique] UNIQUE
(Описание)
ALTER TABLE [Сотрудник] ADD CONSTRAINT [Сотрудник_unique] UNIQUE ([Номер Паспорта])
-- Внешние ключи таблиц:
ALTER TABLE [Сотрудник] ADD CONSTRAINT FK_Сотрудник FOREIGN KEY (Военкомат) REFERENCES
Военкомат(Код)
ALTER TABLE [Призывник] ADD CONSTRAINT FK_Призывник_Военкомат FOREIGN KEY (Военкомат)
REFERENCES Военкомат(Код)
ALTER TABLE [Призывник] ADD CONSTRAINT FK_Призывник_Профотбор FOREIGN KEY ([Группа
Профотбора]) REFERENCES [Профессиональный отбор](Группа)
ALTER TABLE [Специальная подготовка] ADD CONSTRAINT FK_СпециальнаяПодготовка FOREIGN KEY
([Код учреждения]) REFERENCES [Специальное учреждение](Код)
ALTER TABLE [Состав Медкомиссии] ADD CONSTRAINT FK_СоставМедкомиссии_Сотрудник FOREIGN
KEY (Сотрудник) REFERENCES Сотрудник(ФИО)
ALTER TABLE [Состав Медкомиссии] ADD CONSTRAINT FK_СоставМедкомиссии_Медкомитет FOREIGN
KEY (Медкомитет) REFERENCES Медкомитет(Код)
ALTER TABLE [Отсрочка призывника] ADD CONSTRAINT FK_ОтсрочкаПризывника_Призывник FOREIGN
KEY (Призывник) REFERENCES Призывник([Личное Дело])
ALTER TABLE [Отсрочка призывника] ADD CONSTRAINT FK_ОтсрочкаПризывника_Отсрочка FOREIGN
KEY (Отсрочка) REFERENCES Отсрочка(Код)
ALTER TABLE [Направление на специальную подготовку] ADD CONSTRAINT
FK_НаправлениеНаСпециальнуюПодготовку_Призывник FOREIGN KEY (Призывник) REFERENCES
Призывник([Личное Дело])
ALTER TABLE [Направление на специальную подготовку] ADD CONSTRAINT
FK_НаправлениеНаСпециальнуюПодготовку_СпециальнаяПодготовка FOREIGN KEY ([Наименование
специальной подготовки]
REFERENCES [Специальная подготовка](Наименование)
ALTER TABLE [Заключение врачебной комиссии] ADD CONSTRAINT
FK_РезультатВрачебнойКомиссии_Медкомитет FOREIGN KEY (Медкомитет) REFERENCES
Медкомитет(Код)
ALTER TABLE [Заключение врачебной комиссии] ADD CONSTRAINT
FK_РезультатВрачебнойКомиссии_Призывник FOREIGN KEY (Призывник) REFERENCES
[Призывник]([Личное Дело])
-- Составной первичный ключ:
ALTER TABLE [Заключение врачебной комиссии] ADD CONSTRAINT PK_РезультатВрачебнойКомиссии
PRIMARY KEY (Призывник, [Дата Проведения])
-- Значения таблиц по умолчанию:
ALTER TABLE [Сотрудник] ADD CONSTRAINT Сотрудник_Должность_def DEFAULT 'Засекречено' FOR
[Должность]
ALTER TABLE [Призывник] ADD CONSTRAINT Призывник_Адрес_def DEFAULT 'Прописка отсутствует'
FOR [Адрес]
ALTER TABLE [Призывник] ADD CONSTRAINT Призывник_Телефон_def DEFAULT 'Неизвестно' FOR
[Телефон]

```

```

-- Правила для таблиц:
ALTER TABLE [Заключение врачебной комиссии] ADD CONSTRAINT СтепеньГодности_ch CHECK
([Степень годности] LIKE '[АБВГ][1-4]')
ALTER TABLE [Сотрудник] ADD CONSTRAINT НомерПаспорта_ch CHECK (len([Номер Паспорта]) =
10)
ALTER TABLE [Отсрочка] ADD CONSTRAINT Срок_ch CHECK ([Срок] >=1)

GO

-- Представления

-- Результаты Последнего медосмотра
CREATE VIEW [Актуальный медосмотр] AS
    SELECT [Заклучение врачебной комиссии].Призывник, MAX([Заклучение врачебной
комиссии].[Дата Проведения]) AS [Дата Медосмотра]
    FROM [Заклучение врачебной комиссии]
    GROUP BY [Заклучение врачебной комиссии].Призывник

GO

CREATE VIEW [Результаты Последнего медосмотра] AS
    SELECT [Личное Дело] AS Призывник, ФИО, [Заклучение врачебной комиссии].[Степень
годности],
        [Заклучение врачебной комиссии].[Дата Проведения], [Заклучение врачебной
комиссии].Медкомитет
    FROM ((dbo.Призывник LEFT JOIN dbo.[Актуальный медосмотр] ON
dbo.Призывник.[Личное Дело] = dbo.[Актуальный медосмотр].Призывник)
    LEFT JOIN [Заклучение врачебной комиссии] ON (dbo.[Актуальный
медосмотр].Призывник = [Заклучение врачебной комиссии].Призывник
    AND dbo.[Актуальный медосмотр].[Дата Медосмотра] =
dbo.[Заклучение врачебной комиссии].[Дата Проведения]))

GO

-- Все о призывнике
CREATE VIEW [Все О Призывнике] AS
    SELECT [Личное Дело], Призывник.ФИО, [Группа Профотбора], [Год Рождения],
[Результаты Последнего медосмотра].[Степень годности],
    [Отсрочка призывника].[Дата Выдачи] AS [Дата выдачи отсрочки],
Отсрочка.Срок AS [Срок Отсрочки],
    [Направление на специальную подготовку].[Наименование специальной
подготовки], Военкомат.Наименование AS [Военкомат],
    Призывник.Адрес, Призывник.Телефон
    FROM (((Призывник LEFT JOIN [Результаты Последнего
медосмотра] ON Призывник.[Личное Дело] = [Результаты Последнего медосмотра].Призывник)
    LEFT JOIN [Направление на специальную подготовку] ON
Призывник.[Личное Дело] = [Направление на специальную подготовку].Призывник)
    JOIN Военкомат ON Призывник.Военкомат =
Военкомат.Код) LEFT JOIN [Отсрочка призывника] ON Призывник.[Личное Дело] = [Отсрочка
призывника].Призывник
    LEFT JOIN Отсрочка ON [Отсрочка
призывника].Отсрочка = Отсрочка.Код

GO

-- Годные к призыву
CREATE VIEW [Годные к призыву] AS
    SELECT [Личное Дело], ФИО, [Год Рождения], [Степень годности], [Группа
Профотбора], [Наименование специальной подготовки], Военкомат
    FROM [Все О Призывнике]
    WHERE ([Степень годности] LIKE '[АБ]_' AND [Группа Профотбора] <= 3
AND [Дата выдачи отсрочки] IS NULL)

GO

-- Хранимые процедуры

-- Добавление призывника

```

```

/*
0 - Призывник успешно добавлен
1 - Военкомат не найден
2 - Призывник с таким номером личного дела уже существует
3 - Неправильный формат степени годности
4 - Медкомитет не может быть пустым
5 - Дата Медосмотра не может быть пустой
6 - Группа профотбора не может быть пустой
7 - ФИО не может быть пустым
8 - Номер личного дела не может быть пустым
*/

CREATE PROC [Добавить Призывника]
    @ЛичноеДело INT,
    @ФИО NVARCHAR(100),
    @Военкомат INT,
    @Профотбор INT,
    @Телефон NVARCHAR(15),
    @ГодРождения INT,
    @Адрес NVARCHAR(200),
    @Медкомитет INT,
    @СтепеньГодности NVARCHAR(2),
    @ДатаМедосмотра DATETIME2
AS
DECLARE @Код int
SET @Код = 0
    IF (@ЛичноеДело IS NOT NULL) AND (@ЛичноеДело <> '')
        IF (@ФИО IS NOT NULL) AND (@ФИО <> '')
            IF (@Профотбор IS NOT NULL) AND (@Профотбор <> '')
                IF (@ДатаМедосмотра IS NOT NULL) AND (@ДатаМедосмотра <> '')
                    IF (@Медкомитет IS NOT NULL) AND (@Медкомитет <> '')
                        IF @СтепеньГодности LIKE '[АБВГ][1-4]'
                            IF @ЛичноеДело NOT IN (SELECT
Призывник.[Личное Дело] FROM Призывник)
                                IF @Военкомат IN (SELECT
Военкомат.Код FROM Военкомат)
                                    BEGIN
                                        INSERT INTO Призывник
VALUES (@ЛичноеДело, @ФИО, @Военкомат, @Профотбор, @Телефон, @ГодРождения, @Адрес)
                                        INSERT INTO
[Заключение врачебной комиссии] VALUES (@ЛичноеДело, @ДатаМедосмотра, @Медкомитет,
@СтепеньГодности)
                                    END
                                ELSE SET @Код = 1
                            ELSE SET @Код = 2
                        ELSE SET @Код = 3
                    ELSE SET @Код = 4
                ELSE SET @Код = 5
            ELSE SET @Код = 6
        ELSE SET @Код = 7
    ELSE SET @Код = 8
RETURN @Код
GO

-- Добавить сотрудника
/*
0 - Сотрудник успешно добавлен
1 - Военкомат не найден
2 - Стаж не может быть пустым
3 - Должность не может быть пустой
4 - ФИО не может быть пустым

```

```

*/

CREATE PROC [Добавить сотрудника]
    @ФИО NVARCHAR(100),
    @Военкомат INT,
    @Должность NVARCHAR(30),
    @Стаж int,
    @НомерПаспорта PASSPORT

AS
DECLARE @Код int
SET @Код = 0
    IF (@ФИО IS NOT NULL) AND (@ФИО <> '')
        IF (@Должность IS NOT NULL) AND (@Должность <> '')
            IF (@Стаж IS NOT NULL) AND (@Стаж <> '')
                IF @Военкомат IN (SELECT Военкомат.Код FROM Военкомат)
                    BEGIN
                        INSERT INTO Сотрудник VALUES (@ФИО,
@Военкомат, @Должность, @Стаж, @НомерПаспорта)
                    END
                ELSE SET @Код = 1
            ELSE SET @Код = 2
        ELSE SET @Код = 3
    ELSE SET @Код = 4
RETURN @Код

GO

-- Добавить сотрудника в состав медкомиссии

/*
0 - Сотрудник успешно добавлен в состав медкомиссии
1 - Сотрудника не существует
2 - Медкомитета не существует
*/

CREATE PROC [Добавить сотрудника в состав медкомиссии]
    @Сотрудник NVARCHAR(100),
    @Медкомитет INT

AS
DECLARE @Код int
SET @Код = 0
    IF @Сотрудник IN (SELECT Сотрудник.ФИО FROM Сотрудник)
        IF @Медкомитет IN (SELECT Медкомитет.Код FROM Медкомитет)
            BEGIN
                INSERT INTO [Состав Медкомиссии] VALUES (@Сотрудник,
@Медкомитет)
            END
        ELSE SET @Код = 1
    ELSE SET @Код = 2
RETURN @Код

GO

-- Добавление отсрочки призывнику

/*
0 - Отсрочка успешно добавлена призывнику
1 - Призывник уже имеет отсрочку
2 - Не найден тип отсрочки
3 - Призывник не найден
*/

```

```

CREATE PROC [Добавить Призывнику Отсрочку]
    @ЛичноеДело INT,
    @Отсрочка INT,
    @ДатаВыдачи DATETIME2
AS
    DECLARE @Код int
    SET @Код = 0

    IF @ЛичноеДело IN (SELECT Призывник.[Личное Дело] FROM Призывник)
        IF @Отсрочка IN (SELECT Отсрочка.Код FROM Отсрочка)
            IF @ЛичноеДело NOT IN (SELECT [Отсрочка призывника].Призывник FROM
[Отсрочка призывника])
                BEGIN
                    INSERT INTO [Отсрочка призывника]
                    VALUES (@ЛичноеДело, @Отсрочка, @ДатаВыдачи)
                END
            ELSE SET @Код = 1
        ELSE SET @Код = 2
    ELSE SET @Код = 3
RETURN @Код

GO

-- Обновление медкомитета сотруднику, состоящему в медкомиссии

/*
0 - Сотруднику успешно обновлен медкомитет
1 - Сотрудника не состоит в медкомитете
*/

CREATE PROC [Обновить медкомитет сотруднику]
    @Сотрудник NVARCHAR(100),
    @Медкомитет INT
AS
    DECLARE @Код int
    SET @Код = 0

    IF @Сотрудник IN (SELECT [Состав Медкомиссии].Сотрудник FROM [Состав
Медкомиссии])
        BEGIN
            UPDATE [Состав Медкомиссии]
            SET Медкомитет = @Медкомитет
            WHERE Сотрудник = @Сотрудник
        END
    ELSE SET @Код = 1
RETURN @Код

GO

-- Обновление должности сотруднику

/*
0 - Сотруднику успешно обновлена должность
1 - Сотрудника не существует
*/

CREATE PROC [Обновить должность сотруднику]
    @Сотрудник NVARCHAR(100),
    @Должность NVARCHAR(30)
AS

```

```

DECLARE @Код int
SET @Код = 0

IF @Сотрудник IN (SELECT Сотрудник.ФИО FROM [Сотрудник])
BEGIN
    UPDATE Сотрудник
    SET Должность = @Должность
    WHERE ФИО = @Сотрудник
END
ELSE SET @Код = 1
RETURN @Код

GO

-- Обновление отсрочки призывнику

/*
0 - Отсрочка успешно обновлена у призывника
1 - Призывник не имеет отсрочки
2 - Не найден тип отсрочки
3 - Призывник не найден
*/

CREATE PROC [Обновить Призывнику Отсрочку]
    @ЛичноеДело INT,
    @Отсрочка INT,
    @ДатаВыдачи DATETIME2
AS
    DECLARE @Код int
    SET @Код = 0

    IF @ЛичноеДело IN (SELECT Призывник.[Личное Дело] FROM Призывник)
        IF @Отсрочка IN (SELECT Отсрочка.Код FROM Отсрочка)
            IF @ЛичноеДело IN (SELECT [Отсрочка призывника].Призывник FROM
[Отсрочка призывника])
                BEGIN
                    UPDATE [Отсрочка призывника]
                    SET Отсрочка = @Отсрочка, [Дата Выдачи] = @ДатаВыдачи
                    WHERE Призывник = @ЛичноеДело
                END
            ELSE SET @Код = 1
        ELSE SET @Код = 2
    ELSE SET @Код = 3
RETURN @Код

GO

-- Удаление отсрочки у призывника

/*
0 - Отсрочка успешно удалена у призывника
1 - Призывник не имеет отсрочки
2 - Призывник не найден
*/

CREATE PROC [Удалить Призывнику Отсрочку]
    @ЛичноеДело INT
AS
    DECLARE @Код int
    SET @Код = 0

    IF @ЛичноеДело IN (SELECT Призывник.[Личное Дело] FROM Призывник)

```



```

        IF @ЛичноеДело IN (SELECT [Отсрочка призывника].Призывник FROM [Отсрочка
призывника])
            BEGIN
                DELETE [Отсрочка призывника] WHERE Призывник = @ЛичноеДело
            END
        ELSE SET @Код = 1
    ELSE SET @Код = 2
RETURN @Код

GO

-- Удаление сотрудника

/*
0 - Сотрудник успешно удален
1 - Сотрудник не найден
*/

CREATE PROC [Удалить Сотрудника]
@Сотрудник NVARCHAR(100)
AS
    DECLARE @Код int
    SET @Код = 0

    IF @Сотрудник IN (SELECT [Сотрудник].ФИО FROM Сотрудник)
        BEGIN
            DELETE [Сотрудник] WHERE ФИО = @Сотрудник
        END
    ELSE SET @Код = 1
RETURN @Код

GO

-- Удаление сотрудника из состава медкомиссии

/*
0 - Сотрудник успешно удален
1 - Сотрудник не состоит в медкомиссии
*/

CREATE PROC [Удалить Сотрудника из Медкомиссии]
@Сотрудник NVARCHAR(100)
AS
    DECLARE @Код int
    SET @Код = 0

    IF @Сотрудник IN (SELECT [Состав Медкомиссии].Сотрудник FROM [Состав
Медкомиссии])
        BEGIN
            DELETE [Состав Медкомиссии] WHERE Сотрудник = @Сотрудник
        END
    ELSE SET @Код = 1
RETURN @Код

GO

-- Триггеры

-- Стаж сотрудника

CREATE TRIGGER [ТриггерСтаж] ON Сотрудник INSTEAD OF UPDATE
AS
    UPDATE Сотрудник
    SET Стаж = inserted.Стаж

```

```
FROM Сотрудник JOIN inserted ON Сотрудник.ФИО = inserted.ФИО  
WHERE inserted.Стаж > Сотрудник.Стаж
```

ПРИЛОЖЕНИЕ Б: СЦЕНАРИЙ ЗАПОЛНЕНИЯ ТАБЛИЦ БД.

USE master

GO

USE СистемаУчетаПризывников

INSERT INTO Военкомат VALUES

(1, 'Военный комиссариат Московского и Железнодорожного районов г. Рязани',
'Московское шоссе, 14, Рязань', 84912351153),
(2, 'Военный комиссариат Октябрьского и Советского районов Рязанской области',
'Фрунзе, 6а, Рязань', 84912253405),
(3, 'Военный комиссариат Рязанской области', 'Фрунзе, 27, Рязань', 84912255191),
(4, 'Военный комиссариат по Рязанскому и Спасскому району Рязанской области',
'Загородная, 22а, Рязань', 84912289050),
(5, 'Военный комиссариат Рыбновского района Рязанской области', 'Советской Армии,
2, Рыбное, Рязанская область', 84913750477),
(6, 'Сборный пункт военного комиссариата Рязанской области', 'Рязань, Островского,
126', 84912986730)

GO

INSERT INTO [Специальное учреждение] VALUES

(1, 'Профессиональное образовательное учреждение Рязанская автомобильная школа
ДОСААФ России', 'г. Рязань, ул.Новая, д.92', 282063),
(2, 'Рязанское гвардейское высшее воздушно-десантное командное училище', 'г.
Рязань, пл. Генерала армии В. Ф. Маргелова, д. 1', 454353),
(3, '137-й гвардейский парашютно-десантный полк', 'г. Рязань, ул. Октябрьский
городок, 1.', 123321)

GO

INSERT INTO Медкомитет VALUES

(1, '07-01-11', 'Врачи высшей категории'),
(2, '07-05-10', 'Врачи высшей категории'),
(3, '04-05-09', 'Врачи высшей категории'),
(4, '07-11-03', 'Врачи высшей категории'),
(5, '07-12-05', 'Врачи высшей категории'),
(6, '07-05-05', 'Врачи высшей категории'),
(7, '07-05-10', 'Врачи средней категории'),
(8, '07-03-12', 'Врачи средней категории'),
(9, '07-01-13', 'Врачи средней категории'),
(10, '09-11-05', 'Врачи средней категории');

GO

INSERT INTO [Профессиональный отбор] VALUES

(1, 'Рекомендован в первую очередь'),
(2, 'Просто рекомендован'),
(3, 'Удовлетворительная оценка с условной рекомендацией'),
(4, 'Не рекомендован');

GO

INSERT INTO [Отсрочка] VALUES

(1, 1, 'По учебе'),
(2, 1, 'По семейным обстоятельствам'),
(3, 1, 'По работе'),
(4, 1, 'По указу президента'),
(5, 1, 'По состоянию здоровья');

GO

INSERT INTO [Специальная подготовка] VALUES

('Получения водительского удостоверения категории С', 1),
('Получения водительского удостоверения категории В', 1),
('Получения водительского удостоверения категории В+Е', 1),
('Получения водительского удостоверения категории С+Е', 1),
('Парашютная подготовка', 2),
('Курсы по вождению гусеничной техники (гусеничные тягачи)', 3),
('Обучение управлению спортивными самолётами и самолётный спорт', 3)

GO

GO

INSERT INTO [Сотрудник] VALUES

('Иванов Петр Иванович', 1, 'Начальник Военкомата', 15, '6485367385'),
('Фетисов Андрей Дмитриевич', 2, 'Начальник Военкомата', 5, '7386848694'),
('Дергачев Иван Михайлович', 3, 'Начальник Военкомата', 25, '5637584748'),
('Климкин Кирил Андреевич', 3, 'Зам.начальника военкомата', 11, '5646486847'),
('Вислоухов Родион Евграфович', 2, 'Зам. начальника военкомата', 7, '5637585838'),
('Меркулов Артем Сергеевич', 1, 'Зам. начальника военкомата', 22, '9475837583'),
('Костин Денис Олегович', 1, 'Член медицинского комитета', 3, '5639395838'),
('Игнатенко Семён Иванович', 1, 'Член медицинского комитета', 19, '4486825963'),
('Теров Роман Романович', 1, 'Член медицинского комитета', 26, '6838593963'),
('Фарисов Егор Егорович', 1, 'Член медицинского комитета', 40, '4729585938'),
('Тариенко Никита Васильевич', 1, 'Член медицинского комитета', 19, '4577777535'),
('Исива Алексей Иванович', 1, 'Член медицинского комитета', 2, '3333675446')

GO

INSERT INTO [Состав Медкомиссии] VALUES

('Игнатенко Семён Иванович', 1),
('Исива Алексей Иванович', 1),
('Костин Денис Олегович', 1),
('Тариенко Никита Васильевич', 2),
('Теров Роман Романович', 2),
('Фарисов Егор Егорович', 3)

GO

INSERT INTO [Призывник] VALUES

(1, 'Иваненко Андрей Викторович', 2, 1, '564597357825', 2001, 'ул. Железнодорожная д.1'),
(2, 'Карасев Никита Андреевич', 2, 1, '564567357825', 1999, 'ул. Виноградная д.2 кв.55'),
(3, 'Пименов Антон Владимирович', 2, 1, '564447357825', 1993, 'ул. Ленина д.22'),
(4, 'Тарасенко Максим Петрович', 1, 2, '564497357825', 2002, 'ул. Вокзальная д.33. кв. 33'),
(5, 'Феофанов Феофан Зингирович', 1, 2, '564597357777', 1992, 'ул. Николодворянская д. 77 кв.777'),
(6, 'Арутюнян Артур Игоревич', 3, 1, '564999357825', 1998, 'ул. Петрова д.6 кв. 66'),
(7, 'Григоренко Родион Викторович', 4, 3, '561197357825', 1996, 'ул. Новая д.4'),
(8, 'Кариенко Иван Антонович', 3, 4, '564599876825', 1997, 'ул. Пригородная д.33 к. 3. кв. 66'),
(9, 'Шамилов Игнат Николаевич', 5, 4, '564577757825', 2001, 'ул. Островского д.33'),
(10, 'Сидоров Николай Николаевич', 6, 4, '577777357825', 1999, '3-й Переулок д.52. кв. 71');

GO

INSERT INTO [Отсрочка призывника] VALUES

```
(1, 1, '12-04-18'),  
(2, 3, '12-08-18'),  
(3, 1, '7-09-15'),  
(8, 3, '6-12-12');
```

GO

INSERT INTO [Направление на специальную подготовку] VALUES

```
(1, 'Курсы по вождению гусеничной техники (гусеничные тягачи)'),  
(2, 'Парашютная подготовка'),  
(5, 'Получения водительского удостоверения категории В+E'),  
(6, 'Обучение управлению спортивными самолётами и самолётный спорт'),  
(7, 'Получения водительского удостоверения категории С+E'),  
(8, 'Получения водительского удостоверения категории С'),  
(4, 'Получения водительского удостоверения категории С+E')
```

INSERT INTO [Заключение врачебной комиссии] VALUES

```
(1, '07-01-16', 4, 'A3'),  
(1, '07-05-17', 4, 'B1'),  
(1, '04-05-18', 4, 'B3'),  
(2, '07-11-16', 4, 'A4'),  
(2, '07-12-17', 4, 'B1'),  
(2, '07-05-18', 4, 'B2'),  
(3, '07-05-16', 4, 'A1'),  
(3, '07-03-18', 4, 'A1'),  
(4, '07-01-16', 5, 'Г4'),  
(4, '09-11-19', 5, 'A2'),  
(5, '09-12-10', 1, 'B3'),  
(5, '09-11-11', 2, 'Г4'),  
(5, '09-09-19', 3, 'A3'),  
(6, '09-03-5', 4, 'B3'),  
(6, '10-04-17', 5, 'B4'),  
(6, '10-03-19', 1, 'A2'),  
(7, '10-01-14', 2, 'B3'),  
(8, '10-11-12', 3, 'Г2'),  
(9, '10-12-13', 4, 'A1'),  
(10, '10-12-19', 5, 'A2');
```

ПРИЛОЖЕНИЕ В: ИСХОДНЫЙ ТЕКСТ КЛИЕНТСКОЙ ПРОГРАММЫ.

Form1

```
using System;
using System.Windows.Forms;

namespace BD_KURS_SHMAKOV_846
{
    /*
        Главная форма приложения
    */
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();

            // Загрузка формы 1
            private void Form1_Load(object sender, EventArgs e)
            {
                groupBoxTable.Enabled = false;

                // Кнопка подключения к БД
                private void ButtonConnection_Click(object sender, EventArgs e)
                {
                    try {
                        SqlCon.ConnectDatabase();

                        foreach (string TableName in SqlCon.TableNames)
                            comboBox1.Items.Add(TableName);

                        // Выбираем по умолчанию показ главного представления все о призывнике
                        comboBox1.SelectedIndex = 14;
                        // Удаление вспомогательного представления
                        comboBox1.Items.RemoveAt(12);
                        // Делаем доступным для пользователя элементы управления
                        groupBoxTable.Enabled = true;

                    } catch {
                        MessageBox.Show("Ошибка при подключении к базе данных, проверьте
                        правильность введенных данных!");
                    }
                }

                // Кнопка отключения от БД
                private void ButtonCloseConnection_Click(object sender, EventArgs e)
                {
                    SqlCon.DisconnectDatabase();
                    groupBoxTable.Enabled = false;
                }

                // Кнопка добавления отсрочки призывнику
                private void ButtonAddStr_Click(object sender, EventArgs e)
                {
                    Form3 Form3 = new Form3();
                    Form3.ShowDialog();
                }
            }
        }
    }
}
```

```

    }

    // Кнопка описания приложения
    private void ButtonAboutApp_Click(object sender, EventArgs e)
    {
        MessageBox.Show("Данное Клиент-Серверное приложение \n было разработано студентом группы 846 \n " +
            "Шмаковым Андреем Юрьевичем \n в рамках курса *Базы данных и \n клиент-серверные приложения*");
    }

    // Кнопка закрытия приложения
    private void ButtonCloseApp_Click(object sender, EventArgs e)
    {
        Close();
    }

    // Кнопка удаления отсрочки призывнику
    private void ButtonDelStr_Click(object sender, EventArgs e)
    {
        Form4 Form4 = new Form4();
        Form4.ShowDialog();
    }

    // Кнопка обновления отсрочки призывнику
    private void ButtonUpStr_Click(object sender, EventArgs e)
    {
        Form2 Form2 = new Form2();
        Form2.ShowDialog();
    }

    // Выбор таблицы и показ ее содержимого
    private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
    {
        dataGridView1.DataSource =
        SqlCon.ShowTable(comboBox1.SelectedItem.ToString()).DefaultView;
    }

    // Кнопка добавления отсрочки призывнику
    private void buttonAddOtsr_Click(object sender, EventArgs e)
    {
        Form5 Form5 = new Form5();
        Form5.ShowDialog();
    }
}
}

```

Form2

```

using System;
using System.Windows.Forms;

namespace BD_KURS_SHMAKOV_846
{
    /*
     * Форма обновления отсрочки у призывника
     */
    public partial class Form2 : Form
    {
        public Form2()
        {
            InitializeComponent();
        }
    }
}

```

```

    }

    // Загрузка формы
    private void Form2_Load(object sender, EventArgs e)
    {
        // Добавление типов отсрочек для выбора
        foreach (string otsr in Otsr.getOtsr())
            comboBox1.Items.Add(otsr);
    }

    // Кнопка обновления отсрочки призывнику
    private void button1_Click(object sender, EventArgs e)
    {
        OtsrPrizyvnik.prizyvnik = textBox1.Text;
        OtsrPrizyvnik.otsr = comboBox1.Text.ToString();
        OtsrPrizyvnik.data = textBox2.Text;

        MessageBox.Show(OtsrPrizyvnik.updateOtsr());
    }
}

```

Form 3

```

using System;
using System.Windows.Forms;

namespace BD_KURS_SHMAKOV_846
{
    /*
     * Форма добавления призывника
     */
    public partial class Form3 : Form
    {
        public static string tab_name;
        public Form3()
        {
            InitializeComponent();

            // Добавление существующих военкоматов для выбора
            foreach (string voenkomatKod in Voenkomat.ShowAllVoenkomatKod())
                comboBox1.Items.Add(voenkomatKod);

            // Добавление существующих проф. отборов для выбора
            foreach (string profOtbor in ProfOtbor.ShowAllProfOtbor())
                comboBox2.Items.Add(profOtbor);

            // Добавление существующих медкомитетов для выбора
            foreach (string medkomitet in Medkomitet.ShowAllMedkomitet())
                comboBox3.Items.Add(medkomitet);
        }

        // Кнопка добавления призывника
        private void button1_Click(object sender, EventArgs e)
        {
            Prizyvnik.numDela = textBox1.Text;
            Prizyvnik.fio = textBox2.Text;
            Prizyvnik.voenkomat = comboBox1.Text.ToString();
            Prizyvnik.profOtbor = comboBox2.Text.ToString();
            Prizyvnik.phone = textBox3.Text;
            Prizyvnik.dataBirth = textBox4.Text;
            Prizyvnik.adress = textBox5.Text;
            Prizyvnik.medkomitet = comboBox3.Text.ToString();
            Prizyvnik.stepenGodnosti = comboBox4.Text.ToString();
        }
    }
}

```



```

        Prizyvnik.dataMedosmotra = textBox6.Text;

        MessageBox.Show(Prizyvnik.AddPrizyvnik());
    }

    private void Form3_Load(object sender, EventArgs e)
    {
    }
}

```

Form4

```

using System;
using System.Windows.Forms;

namespace BD_KURS_SHMAKOV_846
{
    /*
    Форма удаление отсрочки у призывника
    */
    public partial class Form4 : Form
    {
        public Form4()
        {
            InitializeComponent();
        }

        // Кнопка удаления отсрочки у призывника
        private void button1_Click(object sender, EventArgs e)
        {
            OtsrPrizyvnik.prizyvnik = textBox1.Text;
            MessageBox.Show(OtsrPrizyvnik.DeleteOtsr());
        }

        private void Form4_Load(object sender, EventArgs e)
        {
        }
    }
}

```

Form5

```

using System;
using System.Windows.Forms;

namespace BD_KURS_SHMAKOV_846
{
    /*
    Форма добавления отсрочки призывнику
    */
    public partial class Form5 : Form
    {
        public Form5()
        {
            InitializeComponent();
        }

        // Кнопка доавления отсрочки призывнику
        private void button1_Click(object sender, EventArgs e)
        {
            OtsrPrizyvnik.prizyvnik = textBox1.Text;
            OtsrPrizyvnik.otshr = comboBox1.Text.ToString();
        }
    }
}

```

```

        OtsrPrizyvnik.data = textBox2.Text;

        MessageBox.Show(OtsrPrizyvnik.AddOtsr());
    }

    private void Form5_Load(object sender, EventArgs e)
    {
        // Добвление существующих типов отсрочек для выбора
        foreach (string otsr in Otsr.getOtsr())
            comboBox1.Items.Add(otsr);
    }
}

```

Class Medkomitet

```

using System.Collections.Generic;
using System.Data.SqlClient;

namespace BD_KURS_SHMAKOV_846
{
    /*
    Медкомитет
    */
    class Medkomitet
    {
        public static string kod;
        public static string tabName = "Медкомитет";

        // Все существующие медкомитеты
        public static System.Collections.IList ShowAllMedkomitet()
        {
            List<string> medkomitets = new List<string>();
            SqlCommand cmd = SqlCon.con.CreateCommand();
            cmd.CommandText = "SELECT * FROM " + tabName;
            SqlDataReader reader = cmd.ExecuteReader();

            while (reader.Read())
                medkomitets.Add(reader["Код"].ToString());

            reader.Close();

            return medkomitets;
        }
    }
}

```

Class NaprNaSpecPodgonov

```

namespace BD_KURS_SHMAKOV_846
{
    /*
    Направление на специальную подготовку
    */
    class NaprNaSpecPodgonov
    {
        private int prizyvnik;
        private string nameSpecPodgotov;
    }
}

```

Class Otsr

```

using System.Collections.Generic;
using System.Data.SqlClient;

```

```

namespace BD_KURS_SHMAKOV_846
{
    /*
        Отсрочка
    */
    class Otsr
    {
        public static string kod;
        public static string term;
        public static string desc;
        public static string tabName = "Отсрочка";

        // Все существующие отсрочки
        public static System.Collections.IList getOtsr()
        {
            List<string> otrs = new List<string>();
            SqlCommand cmd = SqlCon.con.CreateCommand();
            cmd.CommandText = "SELECT * FROM " + tabName;
            SqlDataReader reader = cmd.ExecuteReader();

            while (reader.Read())
            {
                otrs.Add(reader["Код"].ToString());
            }

            reader.Close();

            return otrs;
        }
    }
}

```

Class OtsrPrizyvnik

```

using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace BD_KURS_SHMAKOV_846
{
    /*
        Отсрочка призывника
    */
    class OtsrPrizyvnik
    {
        public static string prizyvnik;
        public static string otsr;
        public static string data;

        // Обновление отсрочки
        public static string updateOtsr()
        {
            SqlCommand command = new SqlCommand();
            command.Connection = SqlCon.con;
            SqlDataAdapter adapter = new SqlDataAdapter();
            DataSet dataSet = new DataSet();

            command.CommandType = CommandType.StoredProcedure;
            command.CommandText = "[Обновить Призывнику Отсрочку]";

```

```

command.Parameters.AddWithValue("@ЛичноеДело", prizyvnik);
command.Parameters.AddWithValue("@Отсрочка", otsr);
command.Parameters.AddWithValue("@ДатаВыдачи", data);

command.Parameters.Add("@Код", SqlDbType.Int);
command.Parameters["@Код"].Direction = ParameterDirection.ReturnValue;

SqlDataReader rdr = command.ExecuteReader();

string Message;

switch (Convert.ToInt32(command.Parameters["@Код"].Value))
{
    case 0:
        Message = "Отсрочка успешно обновлена у призывника";
        break;
    case 1:
        Message = "Призывник не имеет отсрочки";
        break;
    case 2:
        Message = "Не найден тип отсрочки";
        break;
    case 3:
        Message = "Призывник не найден";
        break;
    default:
        Message = "Неизвестное сообщение";
        break;
}

rdr.Close();
return Message;
}

// Удаление отсрочки
public static string DeleteOtsr()
{
    SqlCommand command = new SqlCommand();
    command.Connection = SqlCon.con;
    SqlDataAdapter adapter = new SqlDataAdapter();
    DataSet dataSet = new DataSet();

    command.CommandType = CommandType.StoredProcedure;
    command.CommandText = "[Удалить Призывнику Отсрочку]";

    command.Parameters.AddWithValue("@ЛичноеДело", prizyvnik);
    command.Parameters.Add("@Код", SqlDbType.Int);
    command.Parameters["@Код"].Direction = ParameterDirection.ReturnValue;

    SqlDataReader rdr = command.ExecuteReader();

    string Message;

    switch (Convert.ToInt32(command.Parameters["@Код"].Value))
    {
        case 0:
            Message = "Отсрочка успешно удалена у призывника";
            break;
        case 1:
            Message = "Призывник не имеет отсрочки";
            break;
        case 2:

```

```

        Message = "Призывник не найден";
        break;
    default:
        Message = "Неизвестное сообщение";
        break;
    }

    rdr.Close();
    return Message;
}

// Добавление отсрочки
public static string AddOtsr()
{
    SqlCommand command = new SqlCommand();
    command.Connection = SqlCon.con;
    SqlDataAdapter adapter = new SqlDataAdapter();
    DataSet dataSet = new DataSet();

    command.CommandType = CommandType.StoredProcedure;
    command.CommandText = "[Добавить Призывнику Отсрочку]";

    command.Parameters.AddWithValue("@ЛичноеДело", prizyvnik);
    command.Parameters.AddWithValue("@Отсрочка", otsr);
    command.Parameters.AddWithValue("@ДатаВыдачи", data);

    command.Parameters.Add("@Код", SqlDbType.Int);
    command.Parameters["@Код"].Direction = ParameterDirection.ReturnValue;

    SqlDataReader rdr = command.ExecuteReader();

    string Message;

    switch (Convert.ToInt32(command.Parameters["@Код"].Value))
    {
        case 0:
            Message = "Отсрочка успешно добавлена призывнику";
            break;
        case 1:
            Message = "Призывник уже имеет отсрочку";
            break;
        case 2:
            Message = "Не найден тип отсрочки";
            break;
        case 3:
            Message = "Призывник не найден";
            break;
        default:
            Message = "Неизвестное сообщение";
            break;
    }

    rdr.Close();
    return Message;
}
}
}

```

Class Prizyvnik

```

using System;
using System.Data;
using System.Data.SqlClient;

```

```

namespace BD_KURS_SHMAKOV_846
{
    /*
        Призывник
    */
    class Prizyvnik
    {
        public static string numDela;
        public static string fio;
        public static string voenkomat;
        public static string profOtbor;
        public static string phone;
        public static string dataBirth;
        public static string adress;
        public static string medkomitet;
        public static string stepenGodnosti;
        public static string dataMedosmotra;

        // Добавлене призывника
        public static string AddPrizyvnik()
        {
            SqlCommand command = new SqlCommand();
            command.Connection = SqlCon.con;
            SqlDataAdapter adapter = new SqlDataAdapter();
            DataSet dataSet = new DataSet();

            command.CommandType = CommandType.StoredProcedure;
            command.CommandText = "[Добавить Призывника]";

            command.Parameters.AddWithValue("@ЛичноеДело", numDela);
            command.Parameters.AddWithValue("@ФИО", fio);
            command.Parameters.AddWithValue("@Военкомат", voenkomat);
            command.Parameters.AddWithValue("@Профотбор", profOtbor);
            command.Parameters.AddWithValue("@Телефон", phone);
            command.Parameters.AddWithValue("@ГодРождения", dataBirth);
            command.Parameters.AddWithValue("@Адрес", adress);
            command.Parameters.AddWithValue("@Медкомитет", medkomitet);
            command.Parameters.AddWithValue("@СтепеньГодности", stepenGodnosti);
            command.Parameters.AddWithValue("@ДатаМедосмотра", dataMedosmotra);
            command.Parameters.Add("@Код", SqlDbType.Int);
            command.Parameters["@Код"].Direction = ParameterDirection.ReturnValue;

            SqlDataReader rdr = command.ExecuteReader();

            string Message;

            switch (Convert.ToInt32(command.Parameters["@Код"].Value))
            {
                case 0:
                    Message = "Призывник успешно добавлен";
                    break;
                case 1:
                    Message = "Военкомат не найден";
                    break;
                case 2:
                    Message = "Призывник с таким номером личного дела уже существует";
                    break;
                case 3:
                    Message = "Неправильный формат степени годности";
                    break;
                case 4:
                    Message = "Медкомитет не может быть пустым";
                    break;
                case 5:

```

```

        Message = "Дата Медосмотра не может быть пустой";
        break;
    case 6:
        Message = "Группа профотбора не может быть пустой";
        break;
    case 7:
        Message = "ФИО не может быть пустым";
        break;
    case 8:
        Message = "Номер личного дела не может быть пустым";
        break;
    default:
        Message = "Неизвестное сообщение";
        break;
    }

    rdr.Close();
    return Message;
}
}
}

```

Class ProfOtbor

```

using System.Collections.Generic;
using System.Data.SqlClient;

namespace BD_KURS_SHMAKOV_846
{
    /*
     * Профессиональный отбор
     */
    class ProfOtbor
    {
        public static string group;
        public static string description;
        public static string TabName = "[Профессиональный отбор]";

        // Существующие проф. отборы
        public static System.Collections.IList ShowAllProfOtbor()
        {
            List<string> ProfOtbors = new List<string>();
            SqlCommand cmd = SqlCon.con.CreateCommand();
            cmd.CommandText = "SELECT * FROM " + TabName;
            SqlDataReader reader = cmd.ExecuteReader();

            while (reader.Read())
                ProfOtbors.Add(reader["Группа"].ToString());

            reader.Close();

            return ProfOtbors;
        }
    }
}

```

Class SostavMedCom

```

namespace BD_KURS_SHMAKOV_846
{
    /*
     * Состав медкомиссии
     */
    class SostavMedCom
    {

```

```

        private string sotrudnik;
        private int medcom;
    }
}

```

Class Sotrudnik

```

namespace BD_KURS_SHMAKOV_846
{
    /*
     * Сотрудник
     */
    class Sotrudnik
    {
        private string fio;
        private int voenkomat;
        private string dol;
        private int sta;
        private int numPass;
    }
}

```

Class SpecPodgotovka

```

namespace BD_KURS_SHMAKOV_846
{
    /*
     * Специальная подготовка
     */
    class SpecPodgotovka
    {
        private string name;
        private int kodY;
    }
}

```

Class SpecY

```

namespace BD_KURS_SHMAKOV_846
{
    /*
     * Спец. учреждение
     */
    class SpecY
    {
        private int kod;
        private string name;
        private string adress;
        private string phone;
    }
}

```

Class SqlCon

```

using System.Collections.Generic;
using System.Data.SqlClient;
using Microsoft.Data.ConnectionUI;
using System.Data;
using System.Windows.Forms;

namespace BD_KURS_SHMAKOV_846
{
    /*
     * Работа с базой данных
     */
}

```



```

class SqlCon
{
    // Строка подключения
    public static string strcon;
    // Объявление класса подключения к базе
    public static SqlConnection con = new SqlConnection();
    // Список таблиц
    public static List<string> TableNames = new List<string>();

    // Подключение к базе
    public static DialogResult ConnectDatabase() {

        DataConnectionDialog dlg = new DataConnectionDialog();
        DataSource.AddStandardDataSources(dlg);
        DataConnectionDialog.Show(dlg);

        strcon = dlg.ConnectionString;
        string[] parts = strcon.Split(';');
        strcon = parts[1] + ";" + parts[2] + ";" + parts[3] + ";" + parts[4] + ";";

        con.ConnectionString = strcon;
        con.ConnectionString = strcon;
        con.Open();

        DataTable schema = con.GetSchema("Tables");

        // Таблицы в базе
        foreach (DataRow row in schema.Rows)
            TableNames.Add(row[2].ToString());

        return MessageBox.Show("Успешное подключение к базе данных!");
    }

    // Отключение от базы
    public static void DisconnectDatabase() {

        if (con.State == ConnectionState.Open)
        {
            con.Close();
            MessageBox.Show("Соединение закрыто");
        }
        else
        {
            MessageBox.Show("Соединение не было открыто");
        }
    }

    // Показать таблицу
    public static DataTable ShowTable(string TabName) {

        SqlDataAdapter da = new SqlDataAdapter();
        SqlCommand com = con.CreateCommand();
        DataSet ds = new DataSet();

        com.CommandText = "select * from [" + TabName + "]";
        com.ExecuteNonQuery();
        da.SelectCommand = com;
        da.Fill(ds, TabName);

        return ds.Tables[0];
    }
}

```

```

    }
}

```

Class Voenkomat

```

using System.Collections.Generic;
using System.Data.SqlClient;

namespace BD_KURS_SHMAKOV_846
{
    /*
    Военкомат
    */
    class Voenkomat
    {
        public int kod;
        public string name;
        public string address;
        public string phone;
        public static string TabName = "Военкомат";

        // Существующие военкоматы
        public static System.Collections.IList ShowAllVoenkomatKod()
        {
            List<string> VoenkomatKods = new List<string>();
            SqlCommand cmd = SqlCon.con.CreateCommand();
            cmd.CommandText = "SELECT * FROM " + TabName;
            SqlDataReader reader = cmd.ExecuteReader();

            while (reader.Read())
                VoenkomatKods.Add(reader["Код"].ToString());

            reader.Close();

            return VoenkomatKods;
        }
    }
}

```

Class ZaclVrKomis

```

namespace BD_KURS_SHMAKOV_846
{
    /*
    Заключение врачебной комиссии
    */
    class ZaclVrKomis
    {
        private int prizyvnik;
        private string data;
        private string medkom;
        private string stepenGodnosti;
    }
}

```

ПРИЛОЖЕНИЕ Г: СЦЕНАРИЙ ИНСТАЛЛЯЦИИ ПРОГРАММЫ.