Содержание

[Введение 5](#_Toc512424535)

[1 Формирование требований 6](#_Toc512424536)

[1.1 Описание предметной области 6](#_Toc512424537)

[1.2 Описание требований к информационной системе 6](#_Toc512424538)

[2 Разработка моделей 7](#_Toc512424539)

[2.1 Разработка функциональных моделей 7](#_Toc512424540)

[2.2 Разработка моделей данных 15](#_Toc512424544)

[2.3 Разработка моделей ОО ПС 19](#_Toc512424551)

[Заключение 28](#_Toc512424558)

[ПРИЛОЖЕНИЕ А (обязательное) Листинг программы 29](#_Toc512424559)

[ПРИЛОЖЕНИЕ Б 39](#_Toc512424560)

[ПРИЛОЖЕНИЕ В (справочное) Авторская справка 42](#_Toc512424564)

[ПРИЛОЖЕНИЕ Г (справочное) Перечень обозначений и сокращений 43](#_Toc512424568)

[Приложение Д (справочное) Библиографический список 44](#_Toc512424569)

# Введение

Информационная система – взаимосвязанная совокупность средств, методов и персонала, используемых для хранения, обработки и выдачи информации в интересах достижения поставленной цели.

Более узкое понимание информационной системы ограничивает её состав данными, программами и аппаратным обеспечением. Интеграция этих компонентов позволяет автоматизировать процессы управления информацией и целенаправленной деятельности конечных пользователей, направленной на получение, модификацию и хранение информации.

Цель автоматизации информационных процессов - повышение производительности и эффективности труда работников, улучшение качества информационной продукции и услуг, повышение сервиса и оперативности обслуживания пользователей. Автоматизация базируется на использование средств вычислительной техники (СВТ) и необходимого ПО. Знание методов моделирования, принципов построения моделей и выбора средств их реализаций на сегодняшний день являются необходимыми для IT-специалистов, руководителей, инженеров, технологов и других.

Для успешной реализации проекта объект проектирования должен быть, прежде всего, адекватно описан, должны быть построены полные и непротиворечивые функциональные и информационные модели системы.

Проектирование информационных систем всегда начинается с определения цели проекта. Основная задача любого успешного проекта заключается в том, чтобы на момент запуска системы и в течение всего времени ее эксплуатации можно было обеспечить:

* требуемую функциональность системы и степень адаптации к изменяющимся условиям ее функционирования;
* требуемое время реакции системы на запрос;
* готовность и доступность системы для обработки запросов;
* простоту эксплуатации и поддержки системы;
* необходимую безопасность.

Проектирование информационных систем охватывает три основные области:

* проектирование объектов данных;
* проектирование программ, экранных форм, отчетов;
* учет конкретной среды и технологии.

Таким образом, цель настоящей курсовой работы состоит в начальном проектировании информационной системы, предназначенной для дальнейшей разработки программного обеспечения подсистемы автоматизации и учета результатов прохождения медкомиссии.

Для достижения поставленной цели необходимо сформировать структуру, необходимую для понимания взаимосвязей и подчиненности объектов и модулей в соответствии с предметной областью, и разработать функциональные модели и модели данных.

1. Формирование требований

## Описание предметной области

Медкомиссия в военкомате – процесс освидетельствования молодых людей для определения категории их пригодности к службе в Вооруженных силах. Медицинский осмотр юношей призывной категории осуществляется врачебным составом военного комиссариата на медкомиссии. **Медосмотр в военкомате является обязательной процедурой для каждого юноши.**

## Описание требований к информационной системе

Система учета результатов должна представлять собой набор правил и условий, которые прямо влияют на результат прохождения медкомиссии. Благодаря предварительному результату, который формирует программа, приемная комиссия тратит меньше времени на анализ данных, полученных в ходе осмотра пациента врачами медкомиссии. Это увеличивает продуктивность работы медкомиссии в военкомате.

Подсистема учета результатов напрямую связана с БД, куда заносится информация о каждом призывнике: предварительная категория, анализы.

Реализация подсистемы автоматизации и учета результатов прохождения медкомиссии подразумевает под собой создание приложения. Клиентская часть при этом обеспечивает надежный механизм обмена данных с сервером БД и должна поддерживать:

* сохранение данных о призывнике;
* поиск и выбор призывника;
* сохранение категории призывника;
* сохранение основания выбранной категории;
* изменение категории призывника;
* расчет предварительной категории;
* добавления комментария врача;
* вывод данных о призывнике;
* сохранение результата теста.

1. Разработка моделей

## Разработка функциональных моделей

Создание современных информационных систем представляет собой сложнейшую задачу, решение которой требует применения специальных методик и инструментов.

Перед разработкой системы заказчик и разработчик должны ясно представлять, какие функциональные возможности будут заложены в систему, и как будет организовано функциональное взаимодействие внутри системы. При разработке функциональной модели может возникнуть множество проблем. Построение функциональной модели должно решить большую часть этих проблем.

Для построения функциональных модели DFD используется пакет Microsoft Visio 2010.

### Модель IDEF0

IDEF0-методология – методология функционального моделирования, согласно которой система представляется как совокупность взаимодействующих процессов. Такая чисто функциональная ориентация является принципиальной – функции системы анализируются независимо от объектов, которыми они оперируют. Это позволяет более четко смоделировать логику и взаимодействие процессов организации. Поэтому исследование или разработка любой сложной системы начинается с функционального анализа и моделирования как системы в целом, так и всех ее подсистем.

Методология IDEF0 предназначена для функционального моделирования, то есть моделирования выполнения функций объекта, путем создания описательной графической модели, показывающей что, как и кем делается в рамках функционирования любого предприятия. Разработанные IDEF0 модели предназначены для документирования процессов производства, отображения как информация и ресурсы используются на каждом этапе.

IDEF0 используется для создания функциональной модели, отображающей структуру и функции системы, а также потоки информации и материальных объектов, связующие эти функции.

Компоненты синтаксиса IDEF0 – блоки, стрелки, диаграммы и правила.

Блоки представляют функции, определяемые как деятельность, процесс, операция, действие или преобразование. Стрелки представляют данные или материальные объекты, связанные с функциями. Правила определяют, как следует применять компоненты; диаграммы обеспечивают формат графического и словесного описания моделей.

Блок описывает функцию. Внутри каждого блока помещается его имя и номер. Имя должно быть активным глаголов или глагольным оборотом, описывающим функцию.

Стрелки не представляют поток или последовательность событий, как в традиционных блок-схемах потоков или процессов. Они лишь показывают, какие данные или материальные объекты должны поступить на вход функции для того, чтобы эта функция могла выполняться.

Каждая сторона функционального блока имеет стандартное значение с точки зрения связи блок/стрелки. В свою очередь, сторона блока, к которой присоединена стрелка, однозначно определяет ее роль.

Стрелки, входящие в левую сторону блока – входы. Входы преобразуются или расходуются функцией, чтобы создать то, что появится на ее выходе.

Стрелки, входящие в блок сверху – управления. Управления определяют условия, необходимые функции, чтобы произвести правильный выход.

Стрелки, придающие блок справа – выходы, т.е. данные или материальные объекты, произведенные функцией.

Стрелки, подключенные к нижней стороне блока, представляют механизмы. Стрелки, направленные вверх, идентифицируют средства, поддерживающие выполнение функции.

Каждая модель должна иметь контекстную диаграмму верхнего уровня, на которой объект моделирования представлен единственным блоком со стрелками.

Единственная функция, представленная на контекстной диаграмме верхнего уровня, может быть разложена на основные подфункции посредством создания дочерней диаграммы. В свою очередь, каждая из этих подфункций может быть разложена на составные части посредством создания дочерней диаграммы следующего, более низкого уровня, на которой некоторые или все функции также могут быть разложены на составные части. Каждая дочерняя диаграмма содержит дочерние блоки и стрелки, обеспечивающие дополнительную детализацию родительского блока.

На заглавном блоке диаграммы определена единственная функция «Учет результатов прохождения медкомиссии» (Рисунок 2.1).

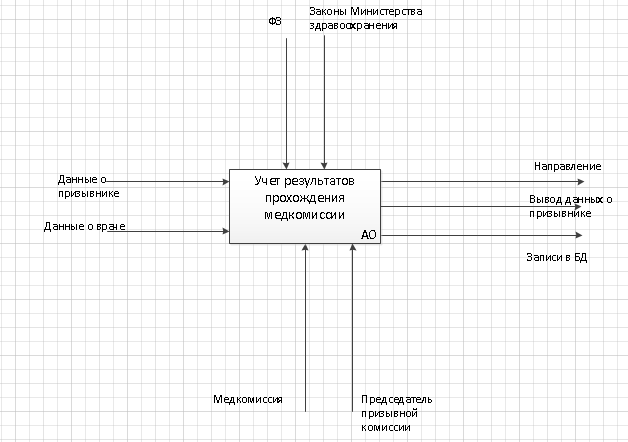
**

Рисунок 2.1 – Блок «Учет результатов прохождения медкомиссии» на контекстной диаграмме IDEF0

Входная информация – «Данные о призывнике», «Данные о враче». Выходная информация – «Направление», «Вывод данных о призывнике» и «Записи в БД». В качестве управления участвует «Федеральный закон» и «Закон Министерства здравоохранения». В качестве механизма системы выступает «Медкомиссия» и «Председатель призывной комиссии».

Основную функцию можно декомпозировать (Рисунок 2.2) на четыре блока, которые представляют основные блоки системы:

* «Авторизация врача»;
* «Прохождение тестов»;
* «Прохождение кабинетов врачей»;
* «Прогнозирование категории».

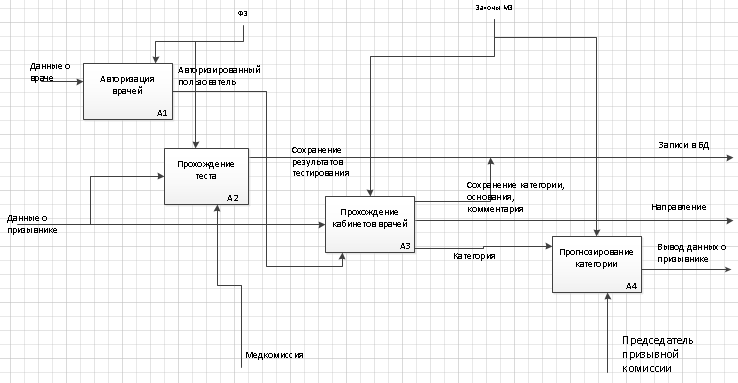
**

Рисунок 2.2 – IDEF0 декомпозиция блока «Учет результатов прохождения медкомиссии»

После декомпозиции контекстной диаграммы переходим к декомпозиции диаграммы следующего уровня, декомпозируя блок «Прохождение кабинетов врачей» (Рисунок 2.3). Данный блок можно разделить на четыре блока:

* «Занесение категории»;
* «Занесение диагноза»;
* «Занесение комментария»;
* «Сохранение в БД».

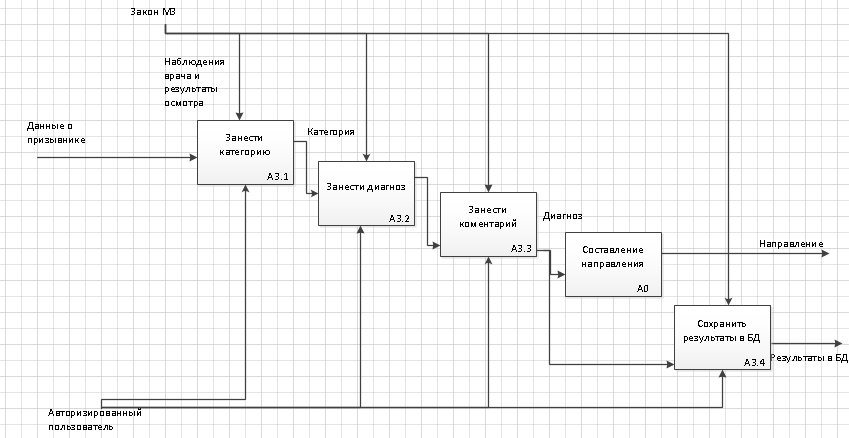
**

Рисунок 2.3 – IDEF0 декомпозиция блока «Прохождение кабинетов врачей»

### Модель IDEF3

IDEF3 является стандартом документирования технологических процессов, происходящих на предприятии, и предоставляет инструментарий для наглядного исследования и моделирования их сценариев.

Сценарием называют описание последовательности изменений свойств объекта в рамках рассматриваемого процесса.

Средства документирования и моделирования IDEF3 позволяют выполнять следующие задачи:

* документировать имеющиеся данные о технологии процесса;
* определять и анализировать точки влияния потоков сопутствующего документооборота на сценарий технологических процессов;
* определять ситуации, в которых требуется принятие решения, влияющего на жизненный цикл процесса;
* содействовать принятию оптимальных решений при реорганизации технологических процессов;
* разрабатывать имитационные модели технологических процессов.

IDEF3-моделирование органично дополняет традиционное моделирование с использованием стандарта методологии IDEF0.

Как и в модели IDEF0, на первом уровне указывается контекстная диаграмма с входными и выходными потоками. Затем контекстный блок декомпозируется, аналогично IDEF0, но с добавлением логики – соединения, служащие для объединения и разделения потоков.

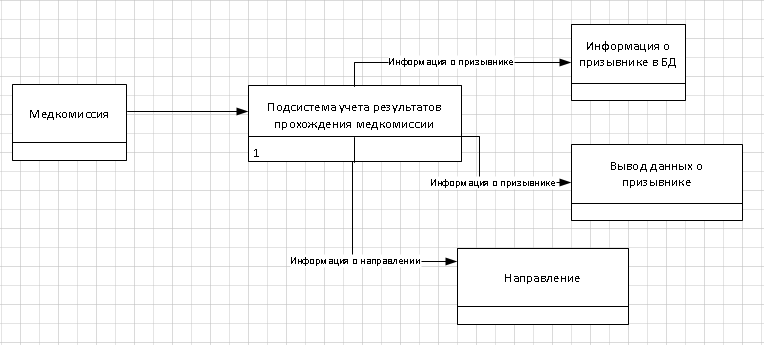
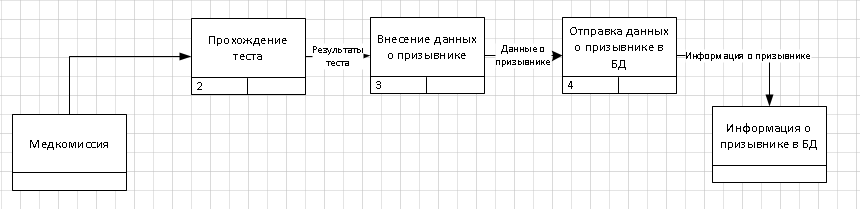
**Используя модель IDEF3, можно отследить возможные варианты развития событий бизнес-процесса, так как связи в модели IDEF3 отражают временную зависимость. Представим контекстную диаграмму «Подсистема учета результатов прохождения медкомиссии» (Рисунок 2.7).

Рисунок 2.4 – Блок «Подсистема учета результатов прохождения медкомиссии» на контекстной диаграмме IDEF3

Входные данные «Медкомиссия». Выходные данные «Информация о призывнике в БД», «Направление» и «Вывод данных о призывнике».

**Далее проведем декомпозицию контекстной диаграммы (Рисунок 2.5).

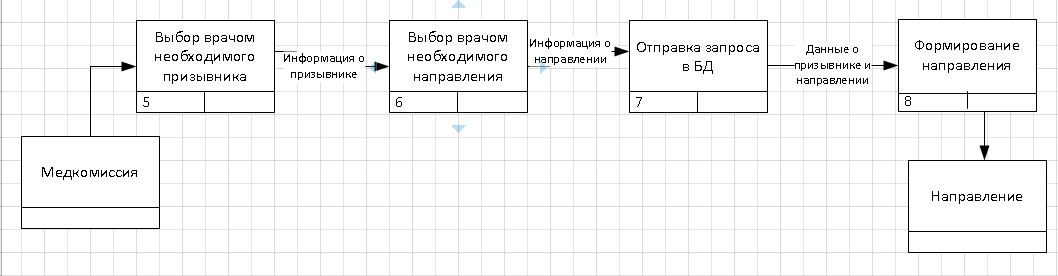
Рисунок 2.5 – Блок «Внесение данных о призывнике»

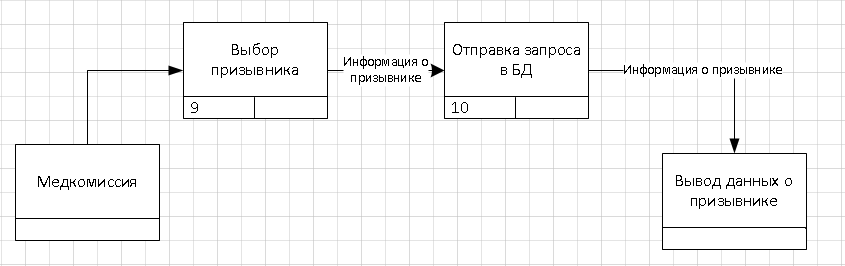
Рисунок 2.6 – Блок «Направление»

Рисунок 2.7 – Блок «Вывод данных о призывнике»

### Модель DFD

При построении функциональной модели системы альтернативной методологии IDEF0 является DFD – методология графического структурного анализа, описывающая внешние по отношению к системе источники и адресаты данных, логические функции, потоки данных и хранилища данных, к которым осуществляется доступ.

Нотация диаграмм потоков данных позволяет отображать на диаграмме как шаги бизнес-процесса, так и поток документов и управления.

Диаграммы потоков данных:

* являются основным средством моделирования функциональных требований к проектируемой системе;
* создаются для моделирования существенного процесса движения информации;
* используются для описания документооборота.

Диаграммы DFD могут дополнить то, что уже отражено в модели IDEF0, поскольку они описывают потоки данных, позволяя проследить, каким образом происходит обмен информацией как внутри системы между бизнес-функциями, так и системы в целом с внешней информационной средой.

Главная цель декомпозиции DFD-функций – продемонстрировать, как каждый процесс преобразует свои входные данные в выходные, а также выявить отношения между этими процессами.

Основным элементом диаграммы является хранилище данных – графическое представление потоков данных импортируемых/экспортируемых из соответствующих баз данных.

Центральной и единственной функцией проектируемой системы является «Подсистема учета результатов прохождения медкомиссии». На ее вход подаются ID от члена медкомиссии, возвращая при этом Предварительный отчет. Так же подается информация о призывнике с сервера БД.

Таким образом получим контекстную диаграмму (Рисунок 2.8).

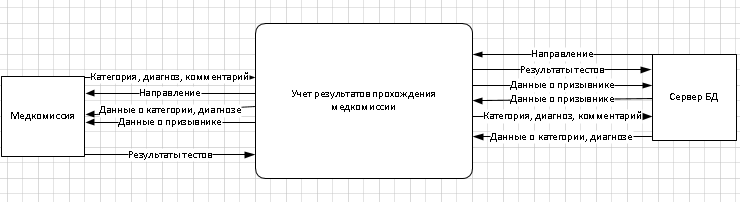
**

Рисунок 2.8 – Блок «Подсистема учета результатов прохождения медкомиссии» на контекстной диаграмме DFD

Далее проведем декомпозицию контекстной диаграммы (Рисунок 2.9).

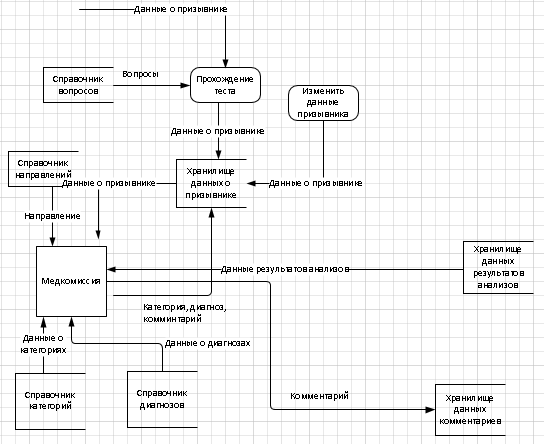
**

Рисунок 2.9 – DFD декомпозиция блока «Подсистема учета результатов прохождения медкомиссии»

На данной декомпозиции продемонстрировано хранилище данных – это «Хранилище данных о призывнике», «Хранилище данных результатов анализов», «Справочник диагнозов», «Справочник категорий», «Хранилище данных комментариев», «Справочник направлений».

## Разработка моделей данных

### Определение списка сущностей и ассоциаций

Для проектируемой информационной системы необходимо выделить следующий набор сущностей:

* член медкомиссии – врач, работающий с программой;
* призывник – человек, подлежащий прохождению военной службы.

Так же выделим ассоциации:

* справочник категорий – хранит список названий категорий;
* справочник диагнозов – хранит список названий диагнозов;
* справочник направлений – хранит список направлений;
* справочник вопросов теста – хранит список вопросов теста;
* справочник врачей – хранит список названий врачей их пароли и логины.

### Определение идентификаторов

Для того, чтобы однозначно определить экземпляры сущностей будут введены идентификаторы. Данные идентификаторы являются атрибутами целого типа, значения которых уникальны в рамках сущности, а также являются константами. Таким образом, ключами для связи сущностей будут использоваться введенные идентификаторы.

Выберем следующие идентификаторы:

* ID\_priz – номер призывника;
* ID\_doc – номер врача;
* ID\_nap – номер направления;
* ID\_login\_doc – имя пользователя;
* ID\_category – номер категории;
* ID\_diag– номер диагноза;
* ID\_test – номер вопроса;

### Определение атрибутов

Охарактеризуем имеющиеся сущности наборами свойств. А также зафиксируем необходимые наборы свойств, которыми будем характеризовать имеющиеся сущности.

* призывник - ID\_priz;
* ФИО - FIO;
* дата рождения – Date;
* номер паспорта - Num\_pasport;
* номер призывного (военного) билета – Nomer\_priz;
* адрес проживания – Adres;
* прописка – Prop;
* телефон – Phone;
* дата последнего прохождения медкомиссии – date\_viz;
* анализы – analysis;
* врач - ID\_doc;
* ФИО - FIO\_doc;
* специальность – Specialnost;
* телефон – Phone;
* имя пользователя – ID\_login\_doc;
* пароль – Password;
* справочник категорий – ID\_category;
* категория – Category;
* справочник направлений – ID\_rout;
* направление – Rout;
* номер вопроса – ID\_test;
* вопрос – Question;
* справочник диагнозов - ID\_table\_diag;
* название диагноза - Name\_diag.

### Описание ограничений целостности

Целостность данных – механизм поддержания соответствия базы данных предметной области. В реляционной модели данных определены два базовых требования обеспечения целостности:

* целостность ссылок;
* целостность сущностей.

Объект реального мира представляется в реляционной базе данных как кортеж некоторого отношения. Требование целостности по ссылкам состоит в том, что для каждого значения внешнего ключа, появляющегося в дочернем отношении в родительском отношении должен найтись кортеж с таким же значением первичного ключа. Требование целостности сущностей заключается в том, что каждый кортеж любого отношения должен отличаться от любого другого кортежа этого отношения.

Существуют следующие типы ограничений:

* NOT NULL:
* UNIQUE;
* CHECK;
* PRIMARY KEY;
* FOREIGN KEY.

В разрабатываемой базе данных необходимы следующие ограничения:

* Первичные ключи «ID\_priz», «ID\_doc», «ID\_login\_doc», «ID\_table\_category», «ID\_test», «ID\_nap», и «ID\_table\_diag» должны быть уникальными;

Ограничение NOT NULL, для всех элементов базы.

### Логическая модель данных

Логическая модель данных описывает понятия предметной области, их взаимосвязь, а также ограничения на данные, налагаемые предметной областью.

Основными компонентами логической модели являются сущности, атрибуты и связи. Каждый атрибут является характеристикой сущности, описанием ее свойств. Связи показывают, как сущности соотносятся друг с другом логически.

Для построения логической модели использовалось программное обеспечение Microsoft Visio 2016 (Приложение Б, рисунок 1).

### Физическая модель данных

Для построения физической модели данных использовался Microsoft Visio 2016. Данная программа имеет два уровня представлена модели – логический и физический. На логическом уровне данные не связаны с конкретной системой управления данными. Физический уровень данных – по существу отображение системного каталога, который зависит от конкретной реализации системы управления базой данных. На физическом уровне объекты базы данных могут называться так, как того требуют ограничения. На логическом уровне можно этим объектам дать синонимы – имена более понятные неспециалистам, в том числе на кириллице и с использованием специальных символов.

Физическая модель приведена на рисунке приложение Б, рисунок 2.

Для данной модели была выбрана СУБД Microsoft SQL.

## Разработка моделей ОО ПС

Для проектирования программных систем была использована Microsoft Visio 2016.

В распоряжении проектировщика Microsoft Visio 2016 предоставляет следующие типы диаграмм:

* use case (диаграммы прецедентов);
* activity (диаграммы активности);
* sequence (диаграммы последовательностей);
* class (диаграммы классов);

### Разработка диаграммы слоев

Схема слоев упорядочивает физические артефакты системы в логические, абстрактные группы, называемые слоями. Слои описывают основные компоненты системы или задачи, выполняемые этими артефактами. Каждый слой может также содержать вложенные слои, описывающие более подробные задачи.

Между слоями можно установить зависимости. Эти зависимости, представленные в виде стрелок, показывают, какие слои могут использовать функции и процессы, представленные другими слоями. Упорядочивая системы в слои, описывающие различные функции, схемы слоев помогают изучать, повторно использовать и обслуживать код.

С помощью схемы слоев можно также выполнять следующие задачи:

* представлять существующую или предполагаемую логическую архитектуру системы;
* выявлять конфликты между существующим кодом и предполагаемой архитектурой;
* визуализировать влияние изменений на предполагаемую архитектуру при обновлении или развитии системы;
* дополнительно контролировать предполагаемую архитектуру в процессе разработки и обслуживания кода за счет добавления проверки операций возврата и построения.

Элементы схемы слоев:

* уровень – совокупность артефактов в системе;
* артефакт – пространства имен, классы, методы и так далее;
* зависимость – указывает, какие слои могут использовать функции других слоев и наоборот, если зависимость двусторонняя;
* комментарий – добавляет примечания к схеме или элементам.

Для разработки приложения был выбран язык программирования (ЯП): C#.

C# - высокоуровневый, объектно-ориентированный язык программирования. Имеет C-подобный синтаксис, который наиболее близок к C++ и Java.

Схема слоев представлена на рисунке 2.10

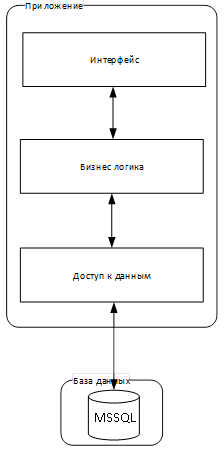


Рисунок 2.10 – Диаграмма слоев

### Разработка диаграммы Use Case

Диаграммы Use Case предназначены для:

* определения поведения системы с точки зрения пользователя.
* первичного моделирования динамики системы.
* выяснения требований к разрабатываемой системе.
* фиксации этих требований в унифицированной форме.

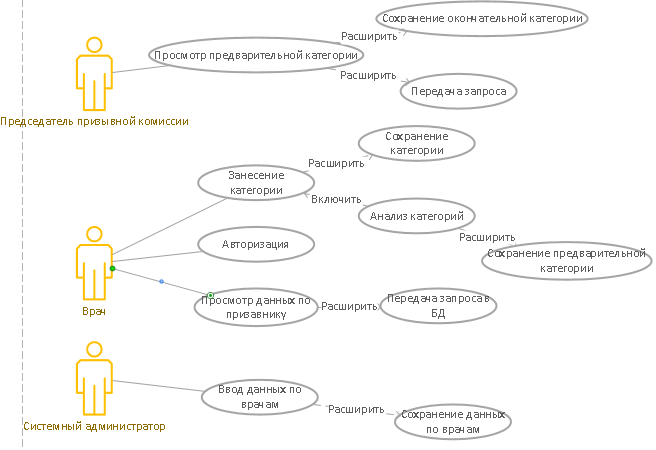
Диаграммы Use Case включает в себя:

* элементы Use Case. Это описание последовательности действий, которые выполняются системой и производят для отдельного актера видимый результат;
* актеры. Это роль объекта вне системы, который прямо взаимодействует с ее частью;
* пользователь. Это физический объект, использующий систему. Он может играть несколько ролей и может моделироваться несколькими актерами;
* отношения: зависимости, обобщения, ассоциации;
* примечания;
* ограничения.

Для данной информационной системы «Системы учета результатов» предполагается наличие следующих актеров:

* врач
* системный администратор
* председатель призывной комиссии
* призывник

Сформируем диаграмму Use Case с применением актеров, элементов и отношений (Рисунок 2.11).

Рисунок 2.11 – Диаграмма Use Case

### Разработка диаграммы активности (Active)

Диаграммы активности предназначены для отображения взаимосвязанных последовательностей действий, производимых объектом в течение его существования.

Этот тип диаграмм позволяет показать не только последовательность процессов, но и ветвление и даже синхронизацию процессов, а также позволяет проектировать алгоритмы поведения объектов любой сложности.

Для каждого из компонентов проектируемой системы характерна общая диаграмма активности (Приложение Г рисунок 3).

### Разработка диаграммы последовательности (Sequenсe)

Взаимодействие объектов в системе происходит посредством приема и передачи сообщений объектами-клиентами и обработки этих сообщений объектами-серверами. При этом в разных ситуациях одни и те же объекты могут выступать и в качестве клиентов, и в качестве серверов.

Данный тип диаграмм позволяет отразить последовательность передачи сообщений между объектами, он не акцентирует внимание на конкретном взаимодействии, главный акцент уделяется последовательности приема/передачи сообщений.

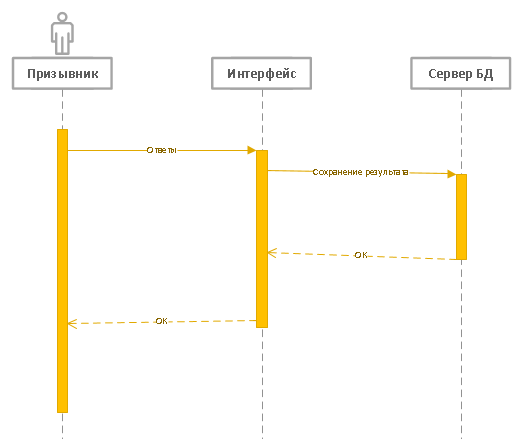
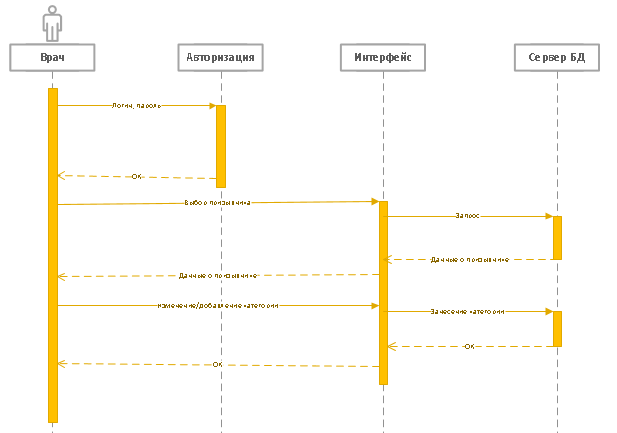
Для системы обмена данными с сервером разработана диаграмма последовательностей – «Прохождение теста» (Рисунок 2.13).

Рисунок 2.13 – Sequence диаграмма сценария «Прохождение теста»

Также разработана диаграмма «Занесение категории» (Рисунок 2.14).

Рисунок 2.14 – Sequence диаграмма сценария «Занесение категории»

Также разработана диаграмма «Просмотр предварительной категории» (Рисунок 2.15).

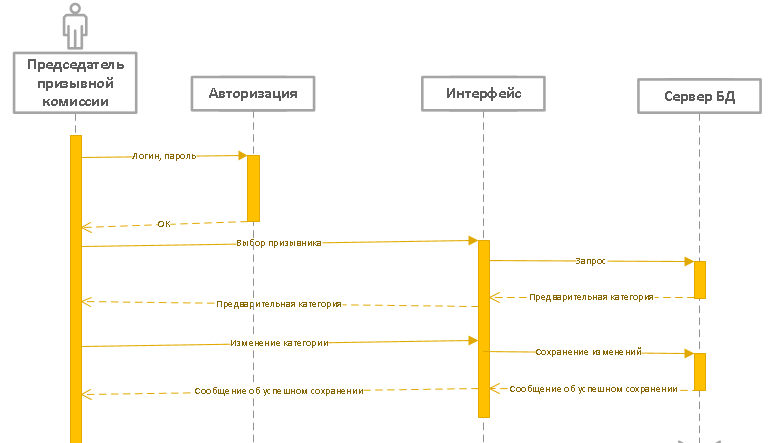


Рисунок 2.15 – Sequence диаграмма сценария «Просмотр предварительной категории»

### Разработка диаграммы классов (Class)

Диаграммы классов позволяют создавать логическое представление системы, на основе которого создается исходный код, описанных классов.

Значки диаграммы позволяют отображать сложную иерархию систем, взаимосвязи классов и интерфейсов.

Диаграмма классов системы (Рисунок 2.16) включает в себя следующие классы:

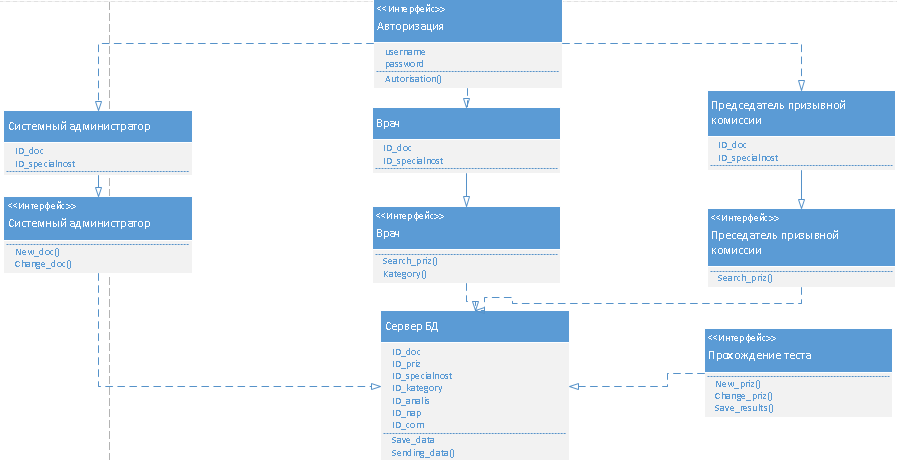
* класс «Системный администратор»;
* класс «Врач»;
* класс «Сервер БД»;
* класс «Председатель призывной комиссии»;
* класс «Интерфейс Председатель призывной комиссии»;
* класс «Интерфейс врач»;
* класс «Интерфейс системный администратор»;
* класс «Интерфейс прохождение теста»
* класс «Система авторизации».

Рисунок 2.16 – Class диаграмма

### Разработка диаграммы компонентов

Диаграмма компонентов предназначена для отображения физической структуры системы. Это делается для создания единого представления о том, какие программы и объекты будут сгруппированы в рамках тех или иных модулей. Эту группировку выполняют из соображений связности. Ее результаты используют, в частности, при назначении заданий проектировщикам.

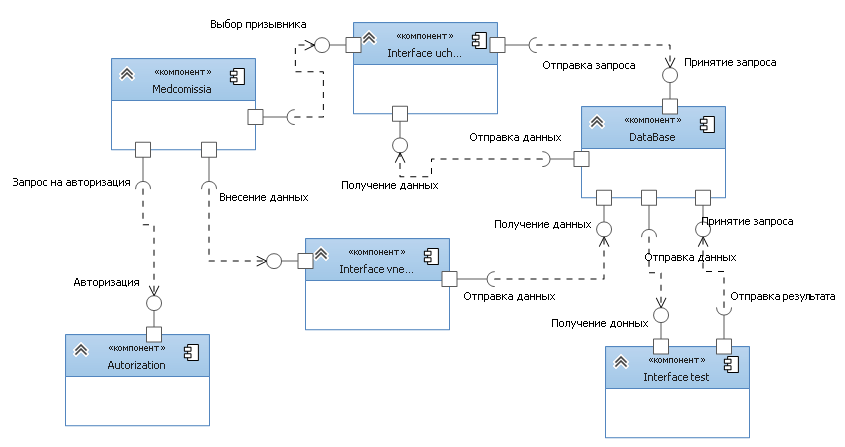
Диаграмма компонентов для данной системы (Рисунок 2.17):

Рисунок 2.17 – Диаграмма компонентов

# Заключение

В рамках выполнения курсовой работы были разработаны:

* требования к предметной области;
* функциональные модели IDEF0, IDEF3, DFD с использованием Microsoft Visio 2016;
* логическая и физическая модели данных с использованием Microsoft Visio 2016;
* моделирование UML диаграмм с помощью Microsoft Visio 2016.

В результате проделанной работы была спроектирована информационная система, предназначенная для дальнейшей разработки программного обеспечения подсистемы учета результатов прохождения медкомиссии.

При разработке были приобретены навыки использования пакетов Microsoft Visio, Visual Studio, а также закреплены знания по дисциплине «Методы и средства проектирования информационных систем и технологий».

Таким образом, можно сделать вывод, что поставленные задачи были выполнены.

# ПРИЛОЖЕНИЕ А (обязательное) Листинг программы

Скрипт создания БД:

GO

/\*\*\*\*\*\* Object: Table [dbo].[Prizivnik] Script Date: 16.04.2018 22:36:11 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Prizivnik](

[ID\_prizivnik] [int] NOT NULL,

[FIO] [nchar](70) NOT NULL,

[Date] [date] NOT NULL,

[Nomer\_priz] [int] NOT NULL,

[Adress] [nchar](150) NOT NULL,

[Prop] [nchar](150) NOT NULL,

[Date\_viz] [date] NULL,

[Phone] [int] NOT NULL,

[ID\_category] [int] NOT NULL,

CONSTRAINT [PK\_Prizivnik] PRIMARY KEY CLUSTERED

(

[ID\_prizivnik] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Rout] Script Date: 16.04.2018 22:36:11 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Rout](

[ID\_rout] [int] NOT NULL,

[Name] [nchar](50) NOT NULL,

[File\_name] [nchar](25) NOT NULL,

CONSTRAINT [PK\_Rout] PRIMARY KEY CLUSTERED

(

[ID\_rout] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Specialnost] Script Date: 16.04.2018 22:36:11 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Specialnost](

[ID\_specialnost] [int] NOT NULL,

[specialnost] [nchar](10) NULL,

CONSTRAINT [PK\_Specialnost] PRIMARY KEY CLUSTERED

(

[ID\_specialnost] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Test] Script Date: 16.04.2018 22:36:11 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Test](

[ID\_test] [int] NOT NULL,

[Question] [nchar](300) NOT NULL,

CONSTRAINT [PK\_Test] PRIMARY KEY CLUSTERED

(

[ID\_test] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[Analys\_priz] WITH CHECK ADD CONSTRAINT [FK\_Analys\_priz\_Analys] FOREIGN KEY([ID\_analys])

REFERENCES [dbo].[Analys] ([ID\_analys])

GO

ALTER TABLE [dbo].[Analys\_priz] CHECK CONSTRAINT [FK\_Analys\_priz\_Analys]

GO

ALTER TABLE [dbo].[Analys\_priz] WITH CHECK ADD CONSTRAINT [FK\_Analys\_priz\_Prizivnik] FOREIGN KEY([ID\_priz])

REFERENCES [dbo].[Prizivnik] ([ID\_prizivnik])

GO

ALTER TABLE [dbo].[Analys\_priz] CHECK CONSTRAINT [FK\_Analys\_priz\_Prizivnik]

GO

ALTER TABLE [dbo].[Category\_priz] WITH CHECK ADD CONSTRAINT [FK\_Category\_priz\_Category] FOREIGN KEY([ID\_category])

REFERENCES [dbo].[Category] ([ID\_category])

GO

ALTER TABLE [dbo].[Category\_priz] CHECK CONSTRAINT [FK\_Category\_priz\_Category]

GO

ALTER TABLE [dbo].[Category\_priz] WITH CHECK ADD CONSTRAINT [FK\_Category\_priz\_Doc] FOREIGN KEY([ID\_doc])

REFERENCES [dbo].[Doc] ([ID\_doc])

GO

ALTER TABLE [dbo].[Category\_priz] CHECK CONSTRAINT [FK\_Category\_priz\_Doc]

GO

ALTER TABLE [dbo].[Category\_priz] WITH CHECK ADD CONSTRAINT [FK\_Category\_priz\_Prizivnik] FOREIGN KEY([ID\_priz])

REFERENCES [dbo].[Prizivnik] ([ID\_prizivnik])

GO

ALTER TABLE [dbo].[Category\_priz] CHECK CONSTRAINT [FK\_Category\_priz\_Prizivnik]

GO

ALTER TABLE [dbo].[Commentarii] WITH CHECK ADD CONSTRAINT [FK\_Commentarii\_Doc] FOREIGN KEY([ID\_doc])

REFERENCES [dbo].[Doc] ([ID\_doc])

GO

ALTER TABLE [dbo].[Commentarii] CHECK CONSTRAINT [FK\_Commentarii\_Doc]

GO

ALTER TABLE [dbo].[Commentarii] WITH CHECK ADD CONSTRAINT [FK\_Commentarii\_Prizivnik] FOREIGN KEY([ID\_priz])

REFERENCES [dbo].[Prizivnik] ([ID\_prizivnik])

GO

ALTER TABLE [dbo].[Commentarii] CHECK CONSTRAINT [FK\_Commentarii\_Prizivnik]

GO

ALTER TABLE [dbo].[Diag\_priz] WITH CHECK ADD CONSTRAINT [FK\_Diag\_priz\_Diagnoz] FOREIGN KEY([ID\_diagnoz])

REFERENCES [dbo].[Diagnoz] ([ID\_diag])

GO

ALTER TABLE [dbo].[Diag\_priz] CHECK CONSTRAINT [FK\_Diag\_priz\_Diagnoz]

GO

ALTER TABLE [dbo].[Diag\_priz] WITH CHECK ADD CONSTRAINT [FK\_Diag\_priz\_Doc] FOREIGN KEY([ID\_doc])

REFERENCES [dbo].[Doc] ([ID\_doc])

GO

ALTER TABLE [dbo].[Diag\_priz] CHECK CONSTRAINT [FK\_Diag\_priz\_Doc]

GO

ALTER TABLE [dbo].[Diag\_priz] WITH CHECK ADD CONSTRAINT [FK\_Diag\_priz\_Prizivnik] FOREIGN KEY([ID\_priz])

REFERENCES [dbo].[Prizivnik] ([ID\_prizivnik])

GO

ALTER TABLE [dbo].[Diag\_priz] CHECK CONSTRAINT [FK\_Diag\_priz\_Prizivnik]

GO

ALTER TABLE [dbo].[Doc] WITH CHECK ADD CONSTRAINT [FK\_Doc\_Specialnost] FOREIGN KEY([ID\_specialnost])

REFERENCES [dbo].[Specialnost] ([ID\_specialnost])

GO

ALTER TABLE [dbo].[Doc] CHECK CONSTRAINT [FK\_Doc\_Specialnost]

GO

ALTER TABLE [dbo].[Otvet] WITH CHECK ADD CONSTRAINT [FK\_Otvet\_Prizivnik] FOREIGN KEY([ID\_priz])

REFERENCES [dbo].[Prizivnik] ([ID\_prizivnik])

GO

ALTER TABLE [dbo].[Otvet] CHECK CONSTRAINT [FK\_Otvet\_Prizivnik]

GO

ALTER TABLE [dbo].[Otvet] WITH CHECK ADD CONSTRAINT [FK\_Otvet\_Test] FOREIGN KEY([ID\_test])

REFERENCES [dbo].[Test] ([ID\_test])

GO

ALTER TABLE [dbo].[Otvet] CHECK CONSTRAINT [FK\_Otvet\_Test]

GO

ALTER TABLE [dbo].[Prizivnik] WITH CHECK ADD CONSTRAINT [FK\_Prizivnik\_Category] FOREIGN KEY([ID\_category])

REFERENCES [dbo].[Category] ([ID\_category])

GO

ALTER TABLE [dbo].[Prizivnik] CHECK CONSTRAINT [FK\_Prizivnik\_Category]

Листинг программы:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Data.SqlClient;

using System.Windows.Forms;

using Word = Microsoft.Office.Interop.Word;

namespace WindowsFormsApp3

{

public partial class Form1 : Form

{

public static string k;

public static int p;

public Form1()

{

InitializeComponent();

open();

}

private void dataGridView1\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

k = dataGridView1.CurrentCell.Value.ToString();

p = dataGridView1.CurrentRow.Index;

}

public void open()

{

try

{

this.prizivnikTableAdapter.FillBy(this.vKRDataSet.Prizivnik);

dataGridView1.DataSource = prizivnikTableAdapter.GetData();

}

catch (System.Exception ex)

{

System.Windows.Forms.MessageBox.Show(ex.Message);

}

}

private void button1\_Click(object sender, EventArgs e)

{

Word.Application appWord = new Word.Application();

switch (comboBox1.SelectedIndex)

{

case 0:

{

string fio, adr, prop, data, dataviz, nomer;

DataRow [] cat;

int categoryid;

DataTable tablecat = new VKRDataSet.CategoryDataTable();

fio = dataGridView1.Rows[p].Cells[1].Value.ToString();

adr = dataGridView1.Rows[p].Cells[2].Value.ToString();

prop = dataGridView1.Rows[p].Cells[3].Value.ToString();

data = dataGridView1.Rows[p].Cells[4].Value.ToString();

dataviz = dataGridView1.Rows[p].Cells[6].Value.ToString();

nomer = dataGridView1.Rows[p].Cells[5].Value.ToString();

try

{

this.categoryTableAdapter1.Category(this.vKRDataSet.Category);

tablecat = categoryTableAdapter1.GetData();

}

catch (System.Exception ex)

{

System.Windows.Forms.MessageBox.Show(ex.Message);

}

categoryid = System.Convert.ToInt32(dataGridView1.Rows[p].Cells[8].Value.ToString());

cat = tablecat.Select("ID\_category="+categoryid);

appWord.Visible = true;

var doc = appWord.Documents.Add();

doc.Application.Run("InformationPriz", fio, adr, prop, data, dataviz, nomer, cat[0][1].ToString());

break;

}

case 1:

{

DataTable tablecat = new VKRDataSet.CategoryDataTable();

DataTable tableprizcat = new VKRDataSet.Category\_prizDataTable();

string fio;

string[] category = new string[7];

int prizid;

int g;

DataRow[] cat;

DataRow[] catpriz;

try

{

this.categoryTableAdapter1.Category(this.vKRDataSet.Category);

tablecat = categoryTableAdapter1.GetData();

}

catch (System.Exception ex)

{

System.Windows.Forms.MessageBox.Show(ex.Message);

}

prizid = System.Convert.ToInt32(dataGridView1.Rows[p].Cells[0].Value.ToString());

fio = dataGridView1.Rows[p].Cells[1].Value.ToString();

try

{

this.category\_prizTableAdapter1.catpriz(this.vKRDataSet.Category\_priz);

tableprizcat = category\_prizTableAdapter1.GetData();

}

catch (System.Exception ex)

{

System.Windows.Forms.MessageBox.Show(ex.Message);

}

catpriz = tableprizcat.Select("ID\_priz=" + prizid);

for (int i = 0; i < catpriz.GetLength(0); i++)

{

g = System.Convert.ToInt32(catpriz[i][1]);

cat = tablecat.Select("ID\_category=" + g);

category[i] = cat[0][1].ToString();

}

appWord.Visible = true;

var doc = appWord.Documents.Add();

doc.Application.Run("InformationCat", fio, category[0],category[1], category[2], category[3], category[4], category[5], category[6]);

break;

}

default:

{

break;

}

}

# ПРИЛОЖЕНИЕ Б

## Логическая модель

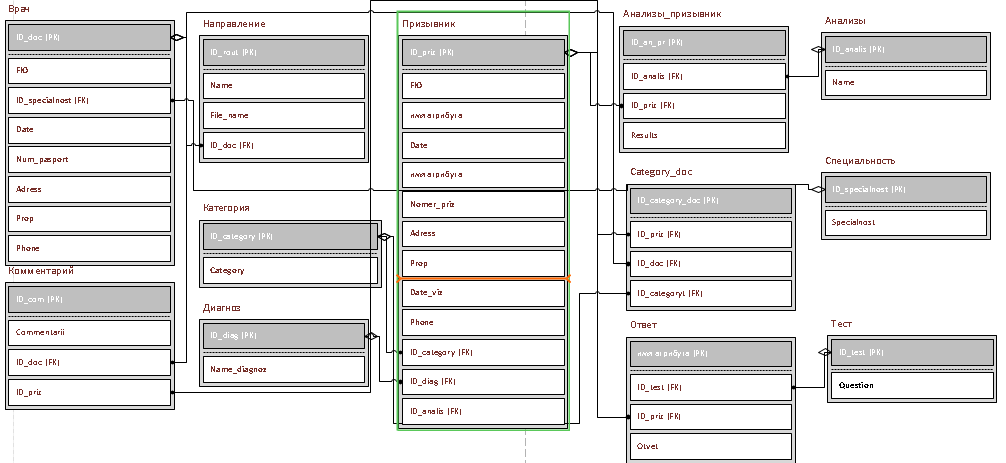


Рисунок 1 – Логическая модель

## Физическая модель

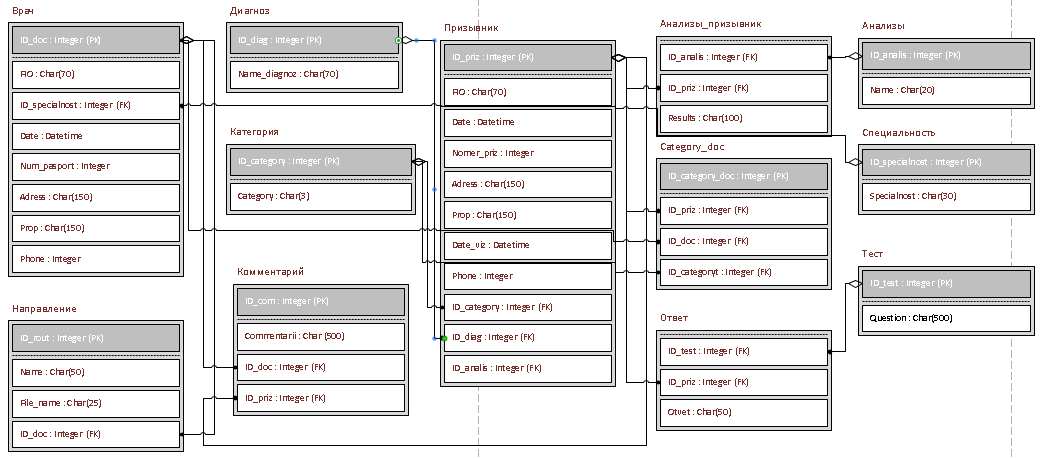


Рисунок 2 – Физическая модель

## Диаграмма активности

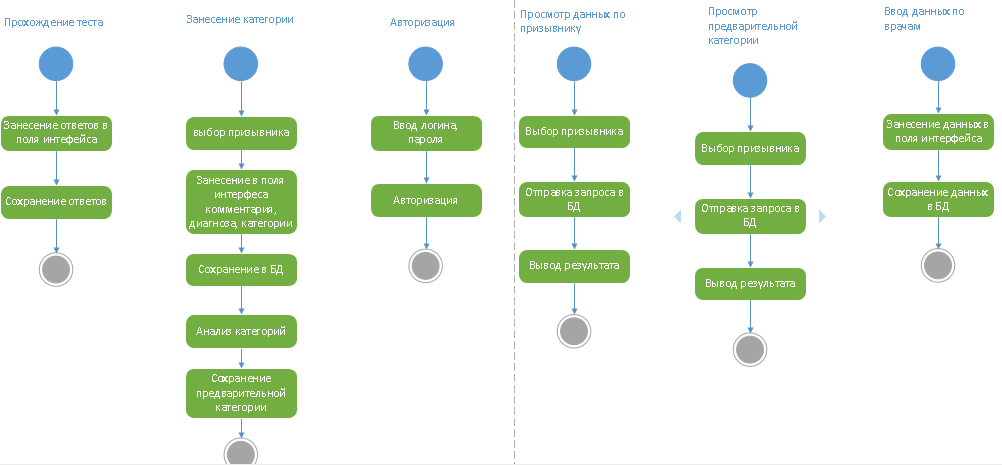


Рисунок 3 – Диаграмма активности

# ПРИЛОЖЕНИЕ В

**Авторская справка**

|  |  |  |
| --- | --- | --- |
| Я, | Пермяков Илья Александрович | |
| автор выпускной квалификационной работы | | Разработка информационной подсисте- |
| мы для прохождения медкомиссии в военкомате | | |

сообщаю, что мне известно о персональной ответственности за разглашение сведений, подлежащих защите законами РФ о защите объектов интеллектуальной собственности.

Одновременно сообщаю, что:

1. При подготовке к защите выпускной квалификационной работы не использованы источники (документы, отчеты, диссертации, литература и т.п.), имеющие гриф секретности или "Для служебного пользования" ВятГУ или другой организации.

2. Данная работа не связана с незавершенными исследованиями или уже с завершенными, но еще официально не разрешенными к опубликованию ВятГУ или другими организациями.

3. Данная работа не содержит коммерческую информацию, способную нанести ущерб интеллектуальной собственности ВятГУ или другой организации.

4. Данная работа не является результатом НИР или ОКР, выполняемой по договору с организацией.

5. В предлагаемом к опубликованию тексте нет данных по незащищенным объектам интеллектуальной собственности других авторов.

6. Не согласен на использование результатов своей работы безвозмездно в ВятГУ для учебного процесса, а также на размещение своей работы в электронной информационно-образовательной среде ВятГУ.

7. Использование моей выпускной квалификационной работы в научных исследованиях оформляется в соответствии с законодательством РФ о защите интеллектуальной собственности.

## Автор \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_ И.А. Пермяков \_\_\_

/личная подпись/ /И. О. Фамилия/

"\_\_\_\_"\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_\_ г.

Сведения по авторской справке подтверждаю:

## И. о. зав кафедрой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_ Ю. В. Ланских\_\_\_\_\_

/личная подпись/ /И. О. Фамилия/

## "\_\_\_\_"\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_\_ г.

# ПРИЛОЖЕНИЕ Г

**Перечень обозначений и сокращений**

ОО ПС – объектно-ориентированное проектирование системы.

СУБД – система управления базами данных.

# Приложение Д (справочное) Библиографический список

1. Корпоративный менеджмент. Основы IDEF3: [Электронынй ресурс]. М. URL: <http://www.cfin.ru/vernikov/idef/idef3.shtml>.
2. Разработка функциональной модели: [Электронный ресурс]. М. URL: <http://dit.isuct.ru/IVT/BOOKS/CASE/case10/IDEF1X/DOC24.HTM>.
3. Сеть разработчиков Microsoft UML-диаграмм: [Электронный ресурс]. М. URL: <https://msdn.microsoft.com/en-us/library/dd409437.aspx>.