# Министерство образования Республики Беларусь

УПРАВЛЕНИЕ ОБРАЗОВАНИЯ МОГИЛЕВСКОГО ОБЛАСТНОГО ИСПОЛНИТЕЛЬНОГО КОМИТЕТА

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

«Могилевский государственный политехнический колледж»

# Домашняя контрольная работа №1

По дисциплине: «**Технология разработки программного обеспечения**»

Группа ПО-455

## Выполнил К.В.Инкин

Шифр 19

# 202116 Что такое ассоциация? Что такое наследование? Что такое полиморфизм. Что такое агрегация? Что такое зависимость?

В объектно-ориентированных языках программирования существует три способа организации взаимодействия между классами. Наследование — это когда класс-наследник имеет все поля и методы родительского класса, и, как правило, добавляет какой-то новый функционал или/и поля. Наследование описывается словом «является». Легковой автомобиль является автомобилем. Вполне естественно, если он будет его наследником.

```class Vehicle

{

bool hasWheels;

}

class Car : Vehicle

{

string model = "Porshe";

int numberOfWheels = 4

}```

Ассоциация – это когда один класс включает в себя другой класс в качестве одного из полей. Ассоциация описывается словом «имеет». Автомобиль имеет двигатель. Вполне естественно, что он не будет являться наследником двигателя (хотя такая архитектура тоже возможна в некоторых ситуациях).   
  
Выделяют два частных случая ассоциации: композицию и агрегацию.   
  
Композиция – это когда двигатель не существует отдельно от автомобиля. Он создается при создании автомобиля и полностью управляется автомобилем. В типичном примере, экземпляр двигателя будет создаваться в конструкторе автомобиля.

```

class Engine

{

int power;

public Engine(int p)

{

power = p;

}

}

class Car

{

string model = "Porshe";

Engine engine;

public Car()

{

this.engine = new Engine(360);

}

}

```

Агрегация – это когда экземпляр двигателя создается где-то в другом месте кода, и передается в конструктор автомобиля в качестве параметра.

```

class Engine

{

int power;

public Engine(int p)

{

power = p;

}

}

class Car

{

string model = "Porshe";

Engine engine;

public Car(Engine someEngine)

{

this.engine = someEngine;

}

}

Engine goodEngine = new Engine(360);

Car porshe = new Car(goodEngine);

```

Полиморфизм также один из основных принципов ООП. Под полиморфизмом (греч. Poly - много, morfos - форма) понимается свойство объектов принимать различные внешние формы в зависимости от обстоятельств. Применительно к ООП полиморфизм означает, что действия, выполняемые одноименными методами, могут различаться в зависимости от того, к какому из классов относится тот или иной метод.

К примеру, три объекта соответствующих классов: двигатель автомобиля, электрический свет в комнате и персональный компьютер. Для каждого из них можно определить операцию выключить(). Однако результат выполнения этой операции будет отличаться для каждого из рассмотренных объектов. Так для двигателя автомобиля выполнение операции выключить() означает прекращение подачи топлива и его остановку. Выполнение операции выключить() для электрического света в комнате означает простой щелчок выключателя, после чего комната погружается в темноту. В последнем случае для персонального компьютера выполнение операции выключить() может быть причиной потери данных, если производится нерегламентированным образом.

Полиморфизм объектно-ориентированных языков связан с перегрузкой функций, но не тождествен ей. Важно иметь в виду, что имена методов и свойств тесно связаны с классами, в которых они описаны. Это обстоятельство обеспечивает определенную надежность работы программы, поскольку исключает случайное применение метода для решения несвойственной ему задачи.

**39 Компьютерная фирма**

**Создайте функциональную модель деятельности компьютерной фирмы, учитывая, что фирма торгует компьютерами в собранном виде и комплектующими. Фирма работает как с производителями компьютерной техники, так и с клиентами. Фирма оказывает ряд дополнительных услуг: установка программного обеспечения, подключает к интернету клиентов, гарантийное обслуживание и т.д.**

Создание контекстной диаграммы.

Контекстная диаграмма отражает отношение системы с внешней средой. Она содержит только один процесс, обозначающий в целом деятельность всей моделируемой системы. В данном случае процесс будет называться «Работа фирмы».

Процесс (работа) - функция системы, набор действий, или элементарное действие. На IDEF0 диаграммах изображается пря­моугольником. Детализируется при помощи диаграмм нижних уровней. Диаграмма главного процесса представлена на рисунке 1.

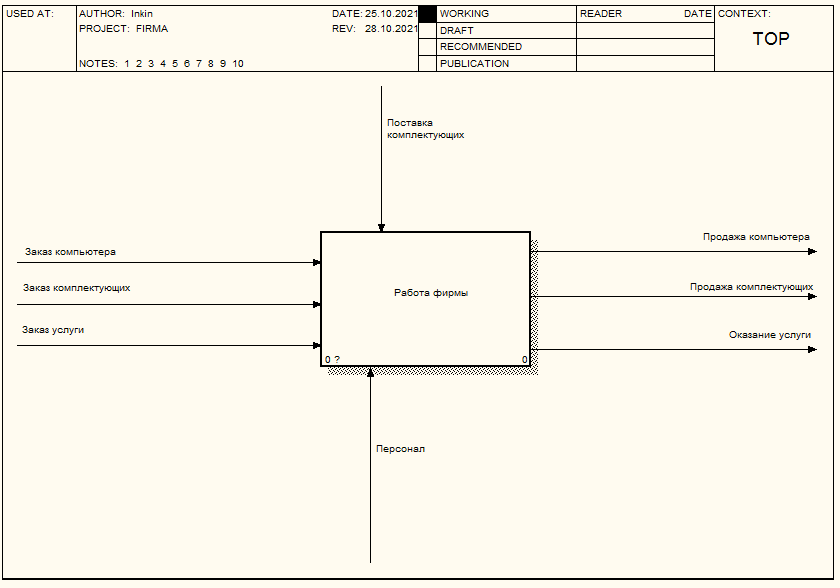


Рисунок 1. Диаграмма главного процесса работы фирмы.

Составим декомпозицию главного рабочего процесса, он состоит из трех основных процессов: сборка компьютера для продажи, продажа комплектующих для самостоятельного ремонта и оказание услуг по ременту, установке и настройке ПО, доступа к сети Интернет.

Диаграмма декомпозиции главного процесса представлена на рисунке 2.

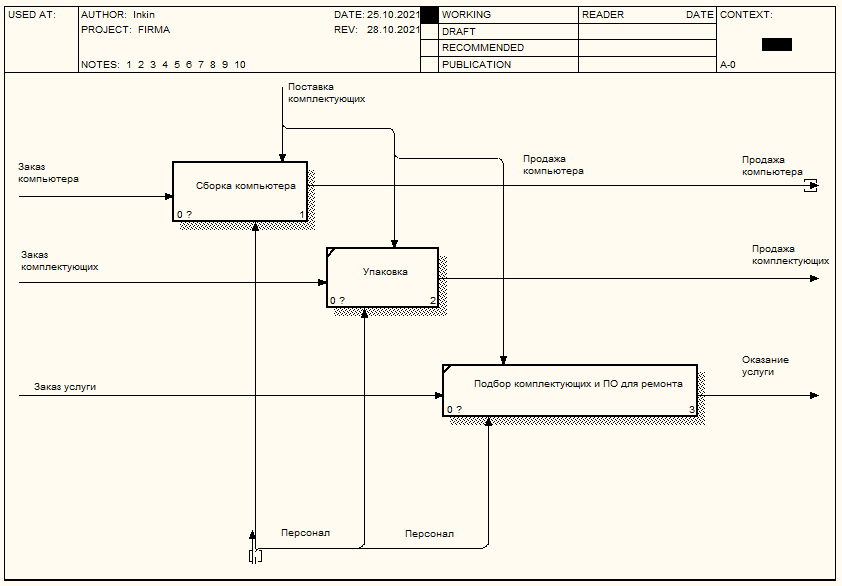
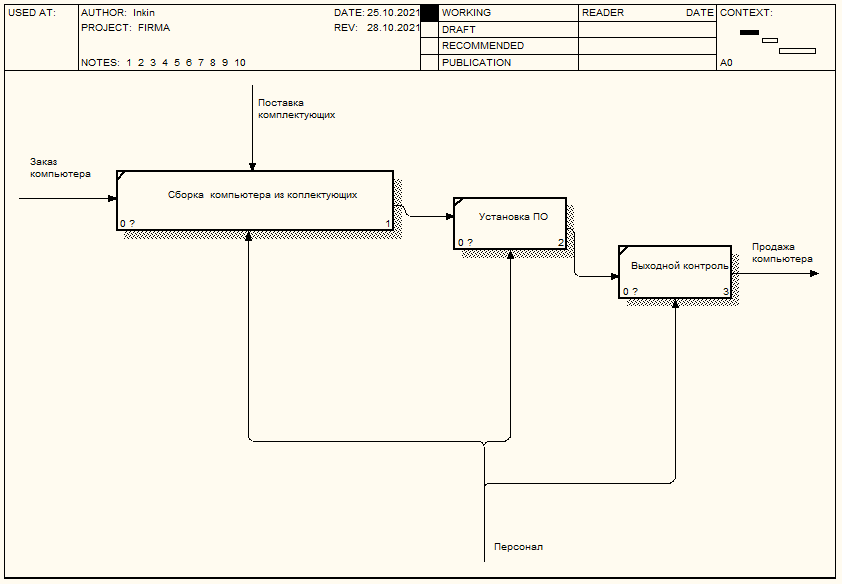


Рисунок 2. Диаграмма декомпозиции главного процесса работы фирмы.

Каждый из этих процессов может быть так же подвергнут декомпозиции. Сборка компьютера состоит из нескольких технологических процессов. Декомпозиция процесса представлена на рисунке 3.



# Рисунок 3. Декомпозиция процесса сборки компьютера.

Декомпозиция процесса оказания услуги представлена на рисунке 4.

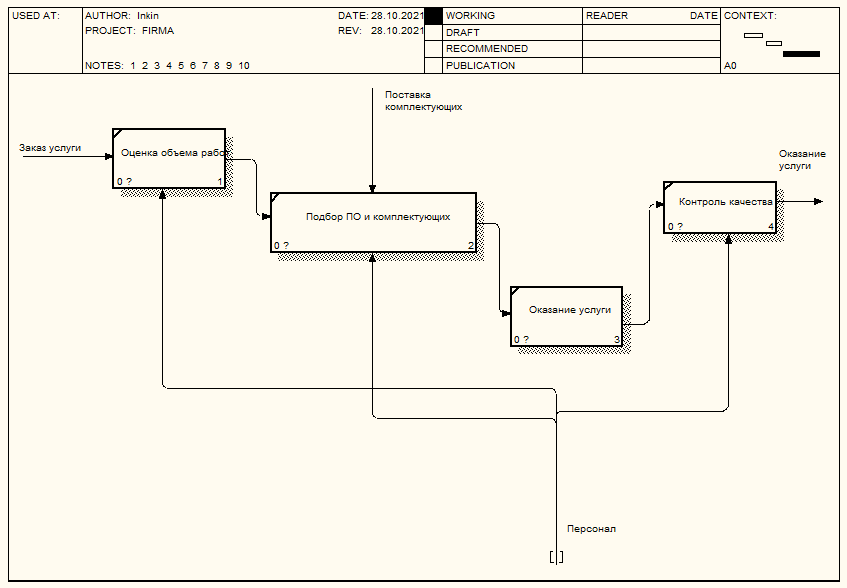


Рисунок 4. Декомпозиция процесса оказания услуги.

**53 Программа Справочник драгоценных и полудрагоценных камней.**

**Пользователь может ввести:**

* **название камня и получить его описание;**
* **знак зодиака и получить список подходящих камней;**
* **знак зодиака и получить список противопоказанных камней.**

**Результаты пользователь может получить как на экране, так и в виде отчета.**

Создание контекстной диаграммы.

Контекстная диаграмма отражает отношение системы с внешней средой. Она должна содержать только один процесс, называемый общей фразой, обозначающий в целом деятельность всей моделируемой системы. В данном случае это процесс будет называться «Обработать данные».

Процесс - функция системы, набор действий, или элементарное действие. В названии процесса обязательно должен присутствовать глагол. Обозначается прямоугольником с закругленными углами. Детализируется при помощи диаграмм нижних уровней.

На контекстной диаграмме изображены все внешние сущности, то есть объекты, поставляющие информацию в систему или получающие ее. В данном примере их две - пользователь и администратор. Обозначается внешняя сущность как прямоугольник с выделенными более ярко двумя границами.

Потоки данных между процессом и внешними сущностями отображают получаемую и передаваемую информацию. В данном примере:

От администратора к процессу идут три потока данных - Данные о пользователях, Данные о камнях, Данные о зодиаках. Это новая информация для базы данных.

От пользователя к процессу идут два потока данных - Название камня, Знак зодиака. Эти потоки необходимы для получения нужных пользователю сведений.

От процесса к пользователю идут три потока данных – Список подходящих камней, Список противопоказанных камней и Описание камня. Эти потоки соответствуют данным, выдаваемым на монитор компьютера и отчетам, напечатанным на принтере.

От процесса к администратору идет один поток данных – Данные о пользователях.

Этих данных достаточно, чтобы построить контекстную диаграмму. Диаграмма приведена на рисунке 5.

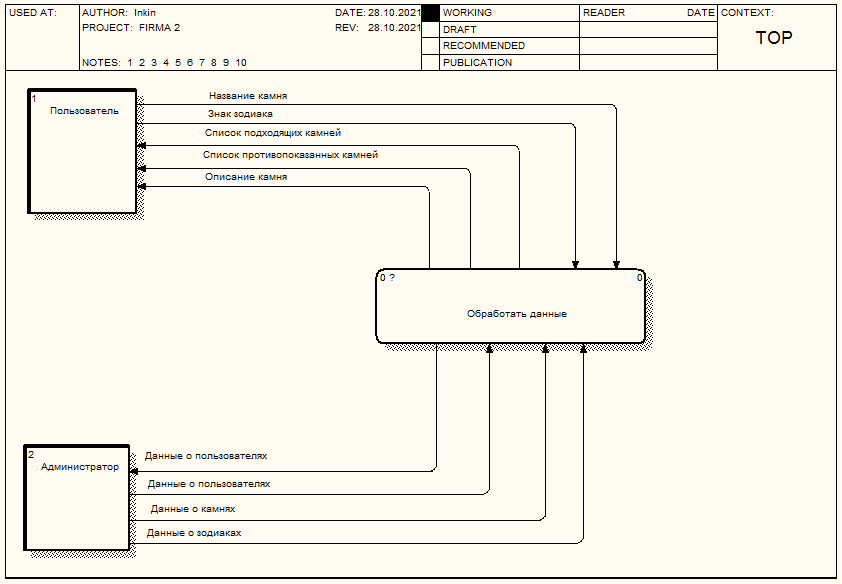


Рисунок 5. Контекстная диаграмма.

На диаграмме первого уровня в данной задаче необходимо разместить пять процессов, соответствующие основным функциям справочной системы.

Эти процессы можно назвать так:

1 Процесс 1 - «Внести новые данные» - носит в базу данных сведения вводимые администратором.

2 Процесс 2 - «Найти камень» - выполняет поиск камня по введенному названию.

3 Процесс 3 - «Найти список подходящих камней» - по введенному знаку зодиака выполняет поиск и выдает описание подходящих камней.

4 Процесс 4 - «Найти список противопоказанных камней» - по введенному знаку зодиака выполняет поиск и выдает описание противопоказанных камней.

5 Процесс 5 - «Распечатать отчеты» - распечатывает для пользователя отчеты из найденных данных.

6 Процесс 6 - «Получить список пользователей» - выводит для администратора список пользователей системы.

Теперь надо распределить информационные потоки между процессами. Потоки на контекстной диаграмме направленные от процесса к пользователю на диаграмме первого уровня направлены стрелкой вовне, то есть от процессов. В дальнейшем они будут обозначаться как исходящие потоки. Потоки, направленные на контекстной диаграмме от пользователя к процессу, на диаграмме первого уровня направлены стрелкой к процессам. В дальнейшем они будут обозначаться как входящие потоки.

Процесс «Найти камень» должен получить данные для поиска. Это название камня. Поэтому надо связать его и входящий поток данных «Название камня». Это процесс выдает пользователь результат поиска - описание найденного камня. Следовательно, исходящий поток данных «Описание камня» надо связать с данным процессом.

Кроме того, данный процесс из найденных данных формирует отчет о данном камне для печати. Следовательно, надо создать новый поток данных и направить его от процесса «Найти камень» к процессу «Распечатать отчеты».

Чтобы получить данные процесс должен направить к базе данных запрос. Из базы данных он получает результат поиска. Надо создать два новых потока. Один от базы данных к процессу, второй от процесса к базе данных.

Аналогичным образом потоки данных распределяются между остальными процессами.

Диаграмма первого уровня представлена на рисунке 6.

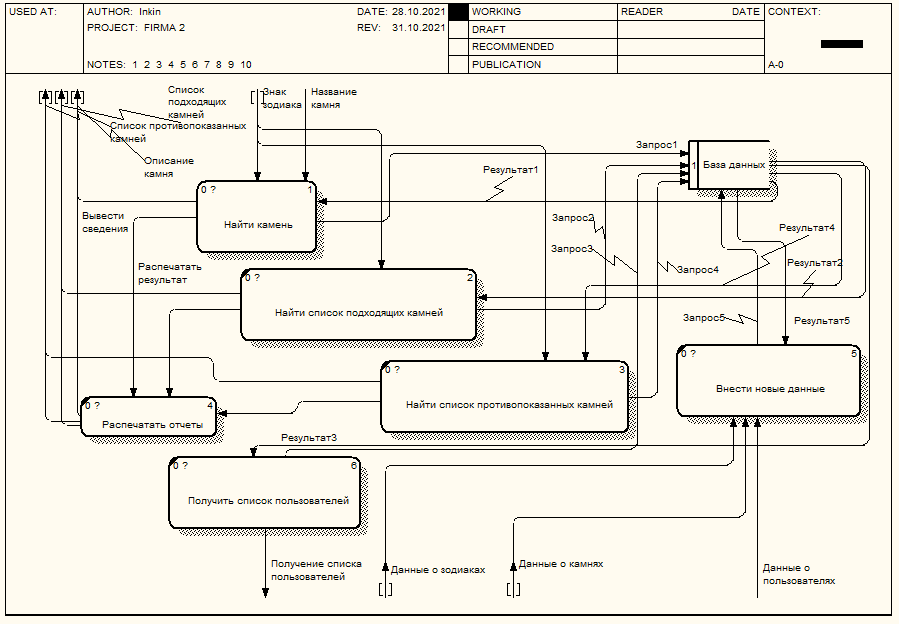


Рисунок 6. Диаграмма первого уровня.

На диаграмме второго уровня будет четыре процесса. Эти процессы детализируют работу процесса «Найти сведения о камне». Названия этих процессов будут следующими:

«Сформировать запрос» - получает потоки «Название камня» и «Знак зодиака», выдает поток данных, направленный к процессу 3.

«Направить запрос в базу данных» - получает поток данных, и направляет его в базу данных.

«Сформировать отчет» - получает поток данных «Результат1» и выдает потоки для вывода данных на экран и печать.

«Выдать результат на экран» - получает поток данных и выдает «Описание камня» на экран.

Диаграмма второго уровня представлена на рисунке 7.

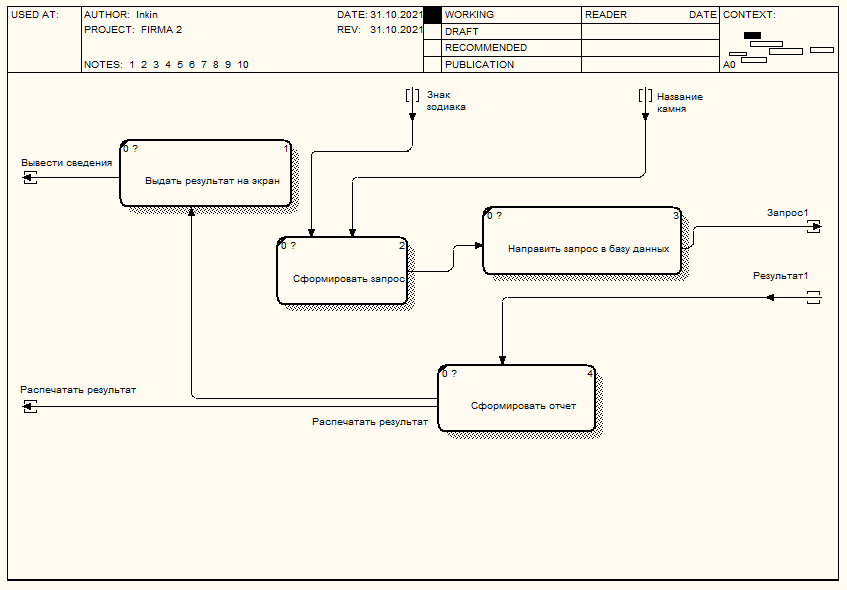


Рисунок 7. Диаграмма второго уровня.

**65 База данных «Студенты колледжа»**

**Хранится информация о студенте (Номер студенческого, ФИО, Адрес, Номер группы), о группе (Номер группы, Куратор, Закрепленные аудитории), поощрениях и наказаниях (Номер студенческого, дата, мероприятие). Студент входит только в одну группу, в группе может быть много студентов. Студент может не получить наказание или поощрение.**

Создадим три сущности: Студенты, Группы и ПоощНак, в которых будем хранить соответствующие им атрибуты.

Пример окна создания сущности «Студенты» приведен на рисунке 8.

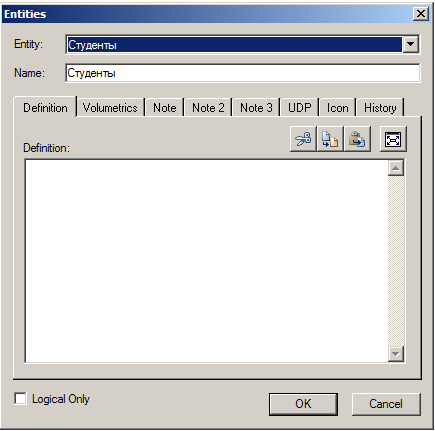


Рисунок 8. Создание сущности «Студенты».

Заполним атрибуты сущности «Студенты». Окно заполнения атрибутов представлено на рисунке 9.

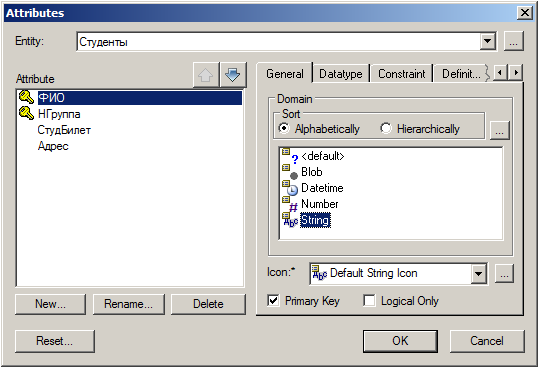


Рисунок 9. Заполнение атрибутов сущности.

После создания сущностей и добавления их атрибутов необходимо установить между ними связи. Окно создания связей представлено на рисунке 10.

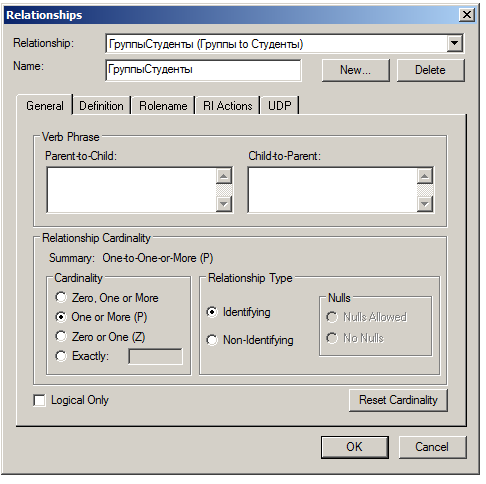


Рисунок 10. Создание связей между сущностями.

При создании связей необходимо верно выставить свойство Cardinality, таким образом, чтобы оно соответствовало заданию.

Диаграмма логического уровня базы данных после указания Сущностей, их Атрибутов и Связей представлена на рисунке 11.

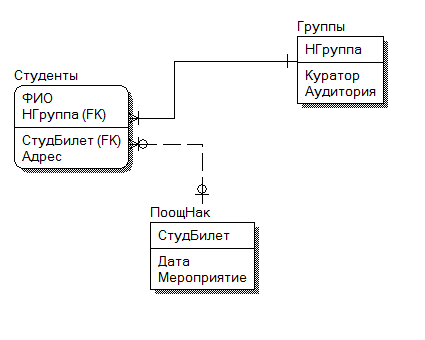


Рисунок 11. Диаграмма логического уровня базы данных.

Диаграмма физического уровня базы данных на рисунке 12.

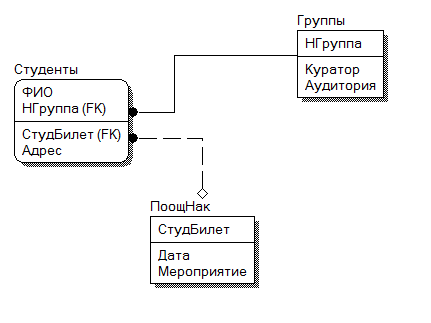


Рисунок 12. Диаграмма физического уровня базы данных.

**85 Разрабатывается программная система, автоматизирующая работу с заказами в парикмахерской. Проведены работы по сбору требований к системе. Согласно собранным сведениям, в системе должны работать администратор и заведующий парикмахерской.**

**Администратор должен выполнять с помощью системы следующие функции: записать клиента на услугу, изменить данные записи, удалить предварительную запись по требованию клиента.**

**Заведующий парикмахерской изменяет данные об услугах и персонале.**

**Предусмотрите идентификацию пользователей и защиту паролями.**

Так как в магазине работает заведующий парикмахерской и администратор, необходима регистрация в системе.

Для построения диаграммы вариантов использования выделены следующие исполнители: Администратор и Заведующий.

Для Администратора выделены следующие варианты использования:

* Зарегистрироваться в системе;
* Записать на услугу;
* Изменить данные записи;
* Удалить запись.

Для Заведующего выделены следующие варианты использования:

* Зарегистрироваться в системе;
* Изменить данные об услугах;
* Изменить данные о персонале.

Разработанная диаграмма вариантов использования представлена на рисунке 13.

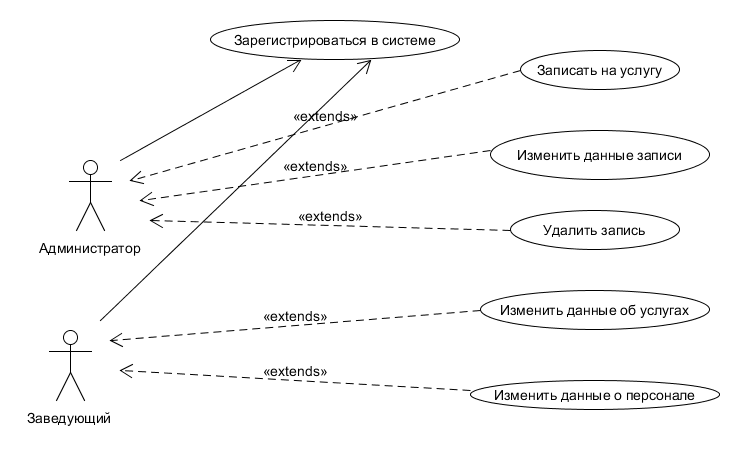


Рисунок 13. Диаграмма вариантов использования

Для решения задачи автоматизации учета продаж были разработаны 3 таблицы.

Диаграмма классов представлена на рисунке 14. На диаграмме выделены для каждой таблицы атрибуты и операции, которые возможно осуществить со справочником. Указаны ключевые поля, созданы связи и указан их тип.

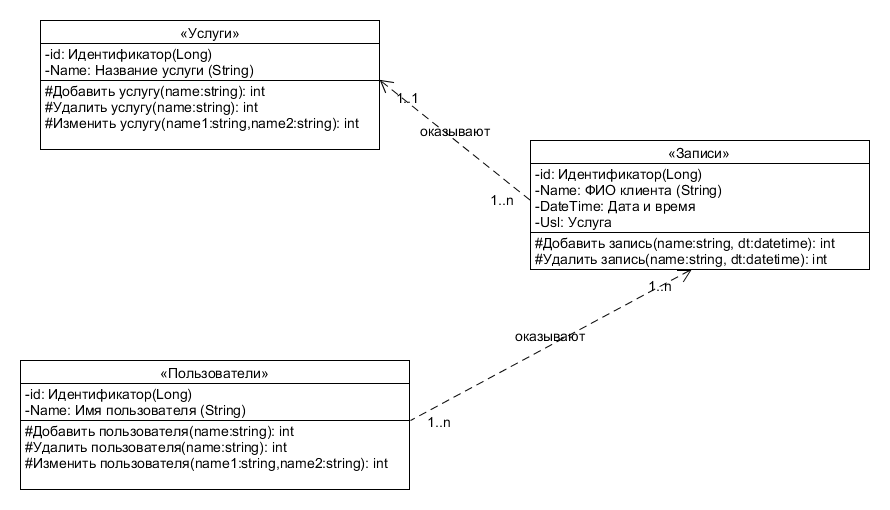


Рисунок 14. Диаграмма классов

**Список используемых источников**

1 ГОСТ 19.201-78. ЕСПД. Техническое задание. Требования к содержанию и оформлению.

2 Калянов, Г.Н. CASE. Структурный системный анализ (автоматизация и применение) / Г.Н.Калянов. - М.:,1996.

3 Купер, А. Алан Купер об интерфейсе. Основы проектирования взаимодействия / А.Купер; пер. с англ. - СПб., 2009.

4 Леоненков, А. Самоучитель UML / А.Леоненков. - 2-е изд. - СПб., 2004.

5 Липаев, В.В. Системное проектирование сложных программных средств для информационных систем / В.В. Липаев. - М., 1999.

6 Маклаков, С.В. BPwin и ERwin. CASE-средства разработки информационных систем / В.В.Липаев. - М.:,1999.

7 Мартин, Р. Принципы, паттерны и методики гибкой разработки на языке C# / Р.Мартин, М.Мартин; пер. с англ. - СПб., 2009.