Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

Институт высоких технологий

Кафедра вычислительной техники

Допускаю к защите

Руководитель \_\_\_\_\_\_Дорофеев А.С.\_\_\_\_\_\_\_\_\_\_\_\_

И.О. Фамилия

Проектирование базы данных и разработка клиент-серверного приложения

наименование темы

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту по дисциплине

**«Базы данных»**

1. 014. 00. 00 - ПЗ

обозначение документа

Выполнил студент ЭВМб-14-1 \_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_Соколова А.С. \_\_\_\_\_\_\_\_\_

шифр группы подпись И.О. Фамилия

Нормоконтроль \_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_Дорофеев А.С.\_\_\_\_\_\_\_\_\_

подпись И.О. Фамилия

Курсовой проект защищен с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Иркутск 2018г.

Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ

ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

ЗАДАНИЕ

НА КУРСОВОЕ ПРОЕКТИРОВАНИЕ

|  |  |
| --- | --- |
| По курсу | Базы данных |

|  |  |
| --- | --- |
| Студенту | Соколовой А.С. |

фамилия, инициалы

|  |  |
| --- | --- |
| Тема проекта: | Проектирование базы данных и разработка клиент-серверного приложения |

Исходные данные:

БД «Телефонный справочник организаций»

Город (телефонный код, название, страна).

Организация (название, адрес, ИНН, телефон).

Организовать поиск организации по номеру телефона или по названию.

Организовать поиск физических лиц по номеру телефона, фамилии

или адресу;

Необходимо выдавать следующие выходные документы:

1. Телефонные коды городов, отсортированные по алфавиту.

2. Телефоны физических лиц города, отсортированные по фамилии.

3. Телефоны всех организаций городов, сгруппированные по городам

и рубрикам.

4. Телефоны организаций города определенной рубрики.

**Рекомендуемая литература:**

1. Кузин А.В. Базы данных : учеб. пособие по направлению подгот. дипломир. специалистов "Информатика и вычисл. техника" / А. В. Кузин, С. В. Левонисова. - 4-е изд., стер.. - М. : Академия, 2010. - 311 с.

2. Базы данных : учеб. пособие / А. С. Дорофеев; Иркут. гос. техн. ун-т . – Иркутск: Изд-во ИрГТУ, 2008. – 99 с. : a-ил.

Графическая часть на листах

Дата выдачи задания “ 04” апреля 2017 г.

Задание получил \_\_\_\_\_\_\_\_\_ Соколова А.С.

Подпись И.О. Фамилия

Дата представления проекта руководителю “\_\_” \_\_\_\_\_ 2018 г.

Руководитель курсового проектирования:                              Дорофеев А.С.

**Содержание**

[Введение 4](#_Toc506940509)

[1 Объектная модель задачи 6](#_Toc506940510)

[1.1 Описание использованного ПО 6](#_Toc506940511)

[1.2 Разработка объектной модели задачи 6](#_Toc506940512)

[1.2.1 Диаграмма прецедентов 6](#_Toc506940513)

[1.2.2 Диаграмма классов 7](#_Toc506940514)

[1.2.3 Диаграмма последовательностей 9](#_Toc506940515)

[2 Структура инфологической модели и результаты ее нормализации 14](#_Toc506940516)

[3 Модель (логическая и физическая) данных 18](#_Toc506940517)

[3.1 Описание использованного ПО 18](#_Toc506940518)

[3.2 Построение логической модели 19](#_Toc506940519)

[3.3 Построение физической модели 19](#_Toc506940520)

[4 Описание базы данных на сервере 24](#_Toc506940521)

[4.1 Создание базы данных программе IBExpert 25](#_Toc506940522)

[4.2 Создание исключений в программе IBExpert 27](#_Toc506940523)

[4.5 Процедуры, созданные в программе IBExpert. 27](#_Toc506940524)

[4.6 Генераторы, созданные в программе IBExpert. 36](#_Toc506940544)

[4.7 Средства просмотра таблиц, созданные в программе IBExpert. 37](#_Toc506940545)

[5 Описание базы данных на клиенте 38](#_Toc506940549)

[6 Описание тестов и результаты прогона тестов 46](#_Toc506940550)

[Заключение 49](#_Toc506940551)

[Список использованных источников 50](#_Toc506940552)

# Введение

В данном курсовом проекте будет разработан телефонный справочник организаций, который включает в себя базу данных и клиентское приложение, позволяющее упростить введение списка городов и организаций. Созданный телефонный справочник будет содержать информацию о телефонах юридических лиц (организаций) и физических лиц. О городе будет храниться следующая информация: телефонный код, название и страна, на территории которой город расположен. Организация характеризуется названием, адресом и телефоном. Организация принадлежит определенной рубрике, а также организации могут иметь несколько телефонов. В базе данных необходимо предоставить возможность поиска информации:

* для физических лиц – по номеру телефона, фамилии или адресу;
* для организаций – по названию или телефону.

Необходимо также выдавать следующие выходные документы:

* Телефонные коды городов, отсортированные по алфавиту.
* Телефоны физических лиц города, отсортированные по фамилии.
* Телефоны всех организаций городов, сгруппированные по городам и рубрикам.
* Телефоны организаций города определенной рубрики.

Целью данного курсового проекта является приобретение практических навыков обследования предметной области, концептуального, логического и физического проектирования базы данных, освоение средств поддержания целостности базы данных, запросов, отчётов, процедур. Курсовой проект отражает этапы создания базы данных с помощью средства разработки IBExpert и другими программными продуктами.

# Объектная модель задачи

## 1.1 Описание использованного ПО

Rational Rose представляет собой CASE средство проектирования и разработки информационных систем и программного обеспечения для управления предприятиями. Первая версия этого продукта была выпущена компанией Rational Software . В дальнейшем Rational Rose был куплен IBM.

Принципиальное отличие Rational Rose от других средств заключается в объектно-ориентированном подходе. Графические модели, создаваемые с помощью этого средства, основаны на объектно-ориентированных принципах и языке ***UML*** (Unified Modeling Language). Инструменты моделирования Rational Rose позволяют разработчикам создавать целостную архитектуру процессов предприятия, сохраняя все взаимосвязи и управляющие воздействия между различными уровнями иерархии.

В результате разработки проекта с помощью CASE-средства Rational Rose формируются следующие документы:

* диаграммы классов;
* диаграммы состояний;
* диаграммы сценариев;
* диаграммы модулей;
* диаграммы процессов;
* спецификации классов, объектов, атрибутов и операций
* заготовки текстов программ;
* модель разрабатываемой программной системы.

Последний из перечисленных документов является текстовым файлом, содержащим всю необходимую информацию о проекте (в том числе необходимую для получения всех диаграмм и спецификаций).

Тексты программ являются заготовками для последующей работы программистов.

## 1.2 Разработка объектной модели задачи

### 1.2.1 Диаграмма прецедентов

Этот вид диаграмм позволяет создать список операций, которые выполняет система. Часто этот вид диаграмм называют диаграммой функций, потому что на основе набора таких диаграмм создается список требований к системе и определяется множество выполняемых системой функций.

Каждая такая диаграмма или, как ее обычно называют, каждый Usecase – это описание сценария поведения, которому следуют действующие лица (Actors).

Данный тип диаграмм используется при описании бизнес процессов автоматизируемой предметной области, определении требований к будущей программной системе. Отражает объекты как системы, так и предметной области и задачи, ими выполняемые.

Построение модели необходимо для того, чтобы лучше понимать разрабатываемую систему.

Перед началом построения диаграммы необходимо определить роли пользователей системы (актеров) и их функции (прецеденты). В нашем случае с системой работают четыре актера – это «Реестр физических лиц», «Реестр юридических лиц», «Министерство» и «Оператор связи».

Под «Реестром физических лиц» я подразумеваю, орган исполнительной власти, в задачи которого входит регистрация физических лиц. Это может быть, например паспортный стол, для которого требуется вести список всех лиц, кому был выдан удостоверяющий документ.

Под «Реестром юридических лиц» можно понимать налоговую инспекцию, для которой требуется регистрировать ИНН организации, ее адрес, контактные телефоны.

В функции «Министерства» входит ведение списка городов России, и ведение телефонных кодов городов.

В функции «Оператора связи» входит регистрация номера телефона в общем банке номеров. Диаграмма прецедентов представлена на рисунке 1.

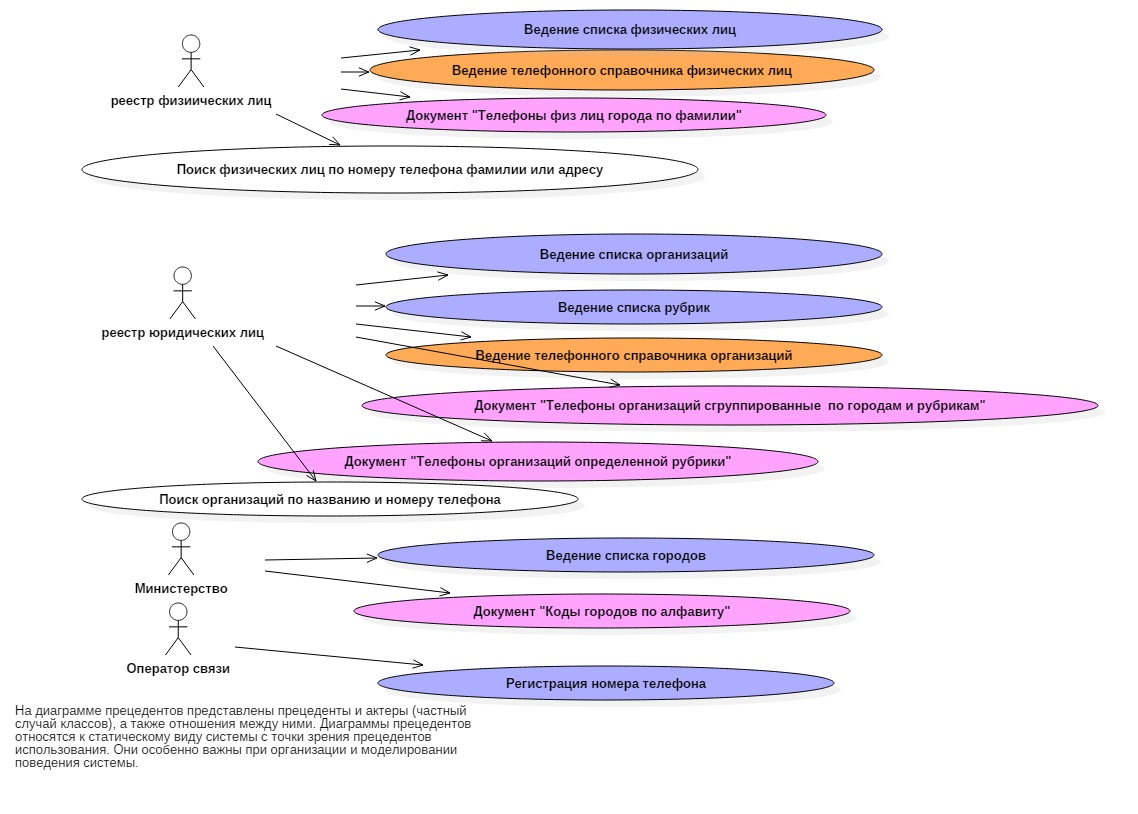


Рисунок 1 -Диаграмма прецедентов

### 1.2.2 Диаграмма классов

Диаграммы классов являются центральным звеном методологии объектно-ориентированных анализа и проектирования.

Диаграмма классов показывает классы и их отношения, тем самым представляя логический аспект проекта. Отдельная диаграмма классов представляет определенный ракурс структуры классов. На стадии анализа диаграммы классов используются, чтобы выделить общие роли и обязанности сущностей, обеспечивающих требуемое поведение системы. На стадии проектирования диаграммы классов используются, чтобы передать структуру классов, формирующих архитектуру системы.

Класс - это лишь шаблон, по которому в дальнейшем будет создан конкретный объект. При этом видимость атрибутов и операций изображается в форме специальных пиктограмм или украшений. Используемые пиктограммы для видимости изображаются перед именем соответствующего атрибута или операции. Для отдельных атрибутов выделенного класса можно задать тип данных и начальные значения атрибута. Для отдельных операций выбранного класса можно задать тип возвращаемого результата, добавить аргументы к операции, назначить для нее стереотип, а также определить протокол и размер, задать исключительные ситуации и целый ряд дополнительных свойств. Диаграммы классов интерфейса приведены на рисунке 2. Диаграмма классов данных приведена на рисунке 3.

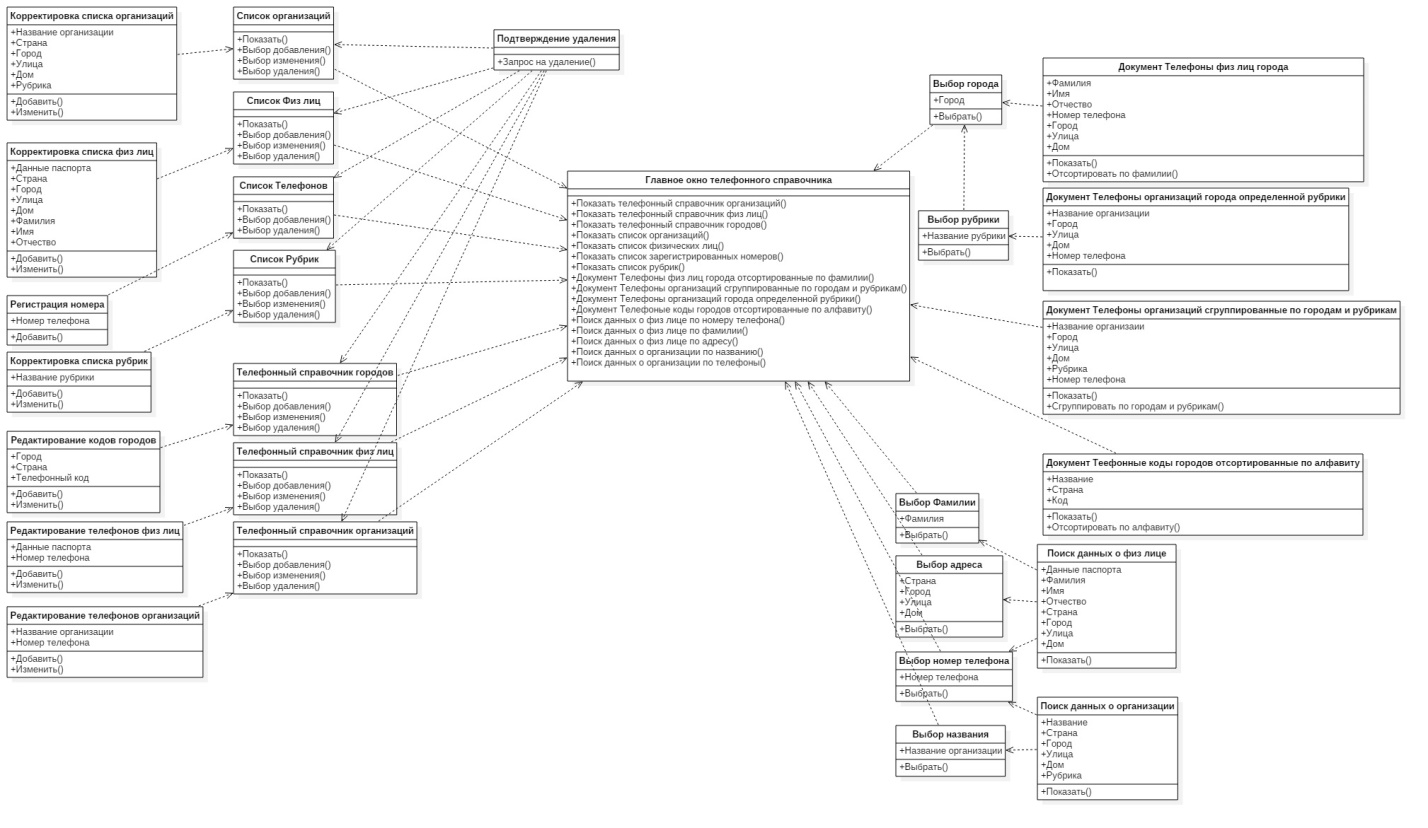


Рисунок 2 - Диаграмма классов (пакет «Интерфейс приложения»)

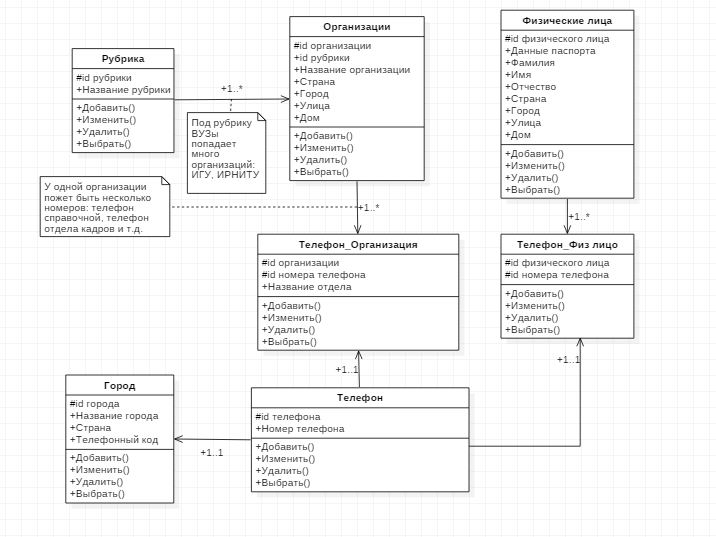


Рисунок 3 - Диаграмма классов (пакет «Сущности БД»)

### 1.2.3 Диаграмма последовательностей

**Диаграмма последовательности**  — диаграмма, на которой для некоторого набора объектов на единой временной оси показаны жизненный цикл (создание-деятельность-уничтожение) и взаимодействие (отправка запросов и получение ответов).

Основными элементами диаграммы последовательности являются обозначения объектов (прямоугольники с названиями объектов), вертикальные «линии жизни», отображающие течение времени, прямоугольники, отражающие деятельность объекта или исполнение им определенной функции (прямоугольники на пунктирной «линии жизни»), и стрелки, показывающие обмен сигналами или сообщениями между объектами.

Данный тип диаграмм позволяет отразить последовательность передачи сообщений между объектами. При добавлении сообщений на диаграмму последовательности они получают по умолчанию свой номер в последовательности.

Создадим по диаграмме последовательностей для каждой из прецедентов:

* Ведение телефонного справочника физических лиц (рисунок 4)
* Ведение телефонного справочника организаций (рисунок 5)
* Ведение справочника кодов городов (рисунок 6)
* Выходной документ : телефоны физических лиц города отсортированные по фамилии (рисунок 7)
* Выходной документ : телефоны организаций сгруппированные по городам и рубрикам (рисунок 8)
* Выходной документ : телефоны организаций города определенной рубрики» (рисунок 9)
* Выходной документ : Коды городов отсортированных по алфавиту (рисунок 10)
* Поиск: организаций по названию и номеру телефона (рисунок 11)
* Поиск: физических лиц по фамилии, номеру телефона или адресу (рисунок 12)
* Ведение списка физических лиц (рисунок 13)
* Ведение списка организаций (рисунок 14)
* Регистрация номера телефона (рисунок 15)
* Ведение списка рубрик» (рисунок 16)

Рисунок 4 - Диаграмма последовательностей для прецедента «Ведение телефонного справочника физических лиц»

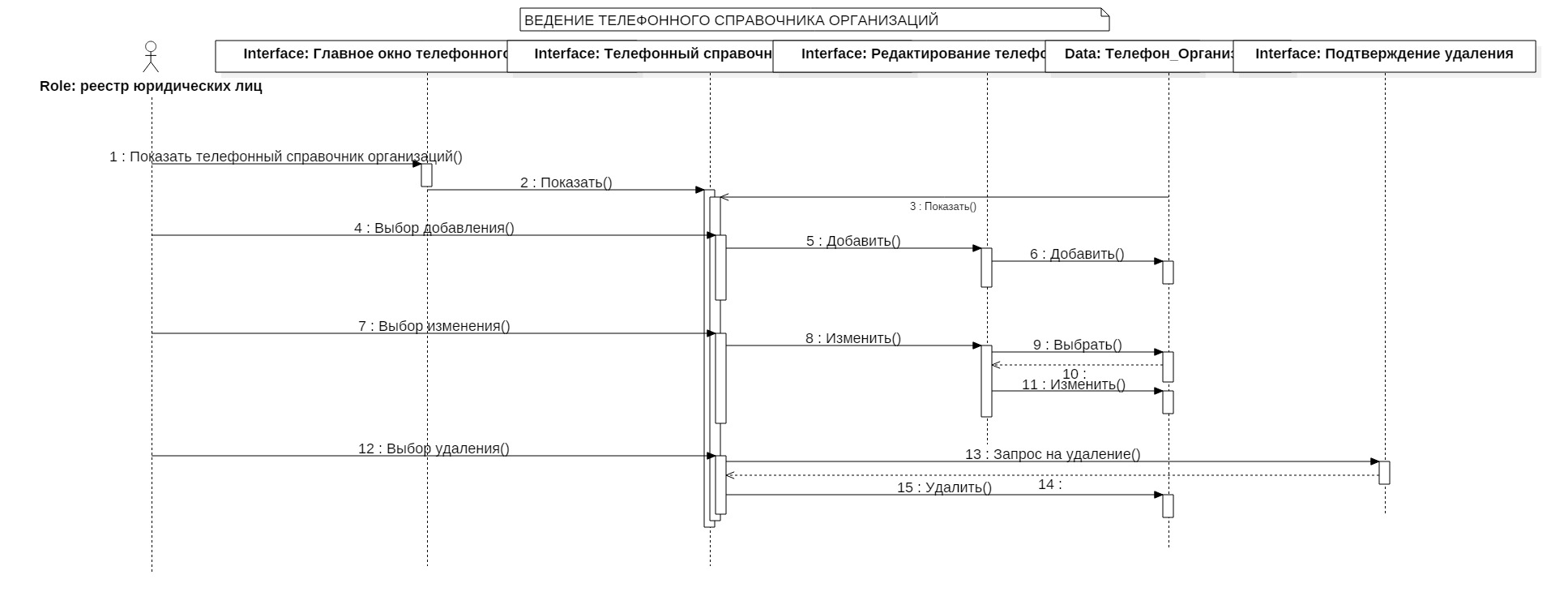
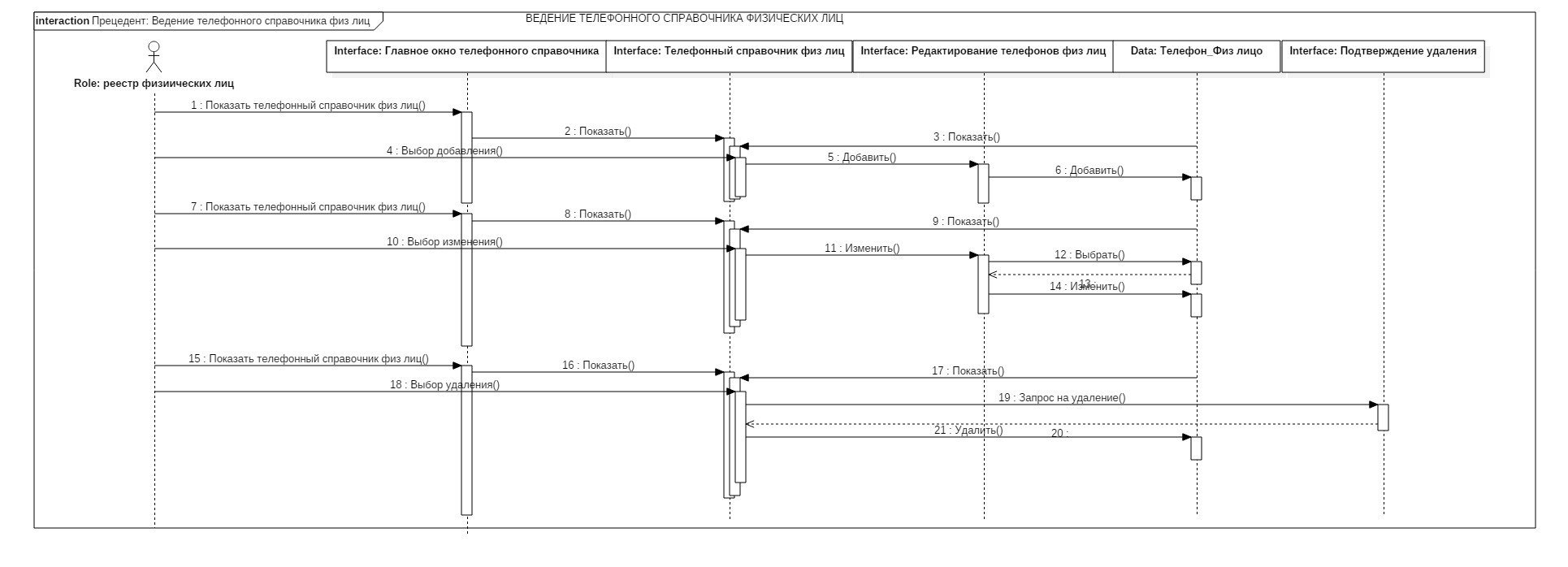


Рисунок 5 - Диаграмма последовательностей для прецедента «Ведение телефонного справочника организаций»

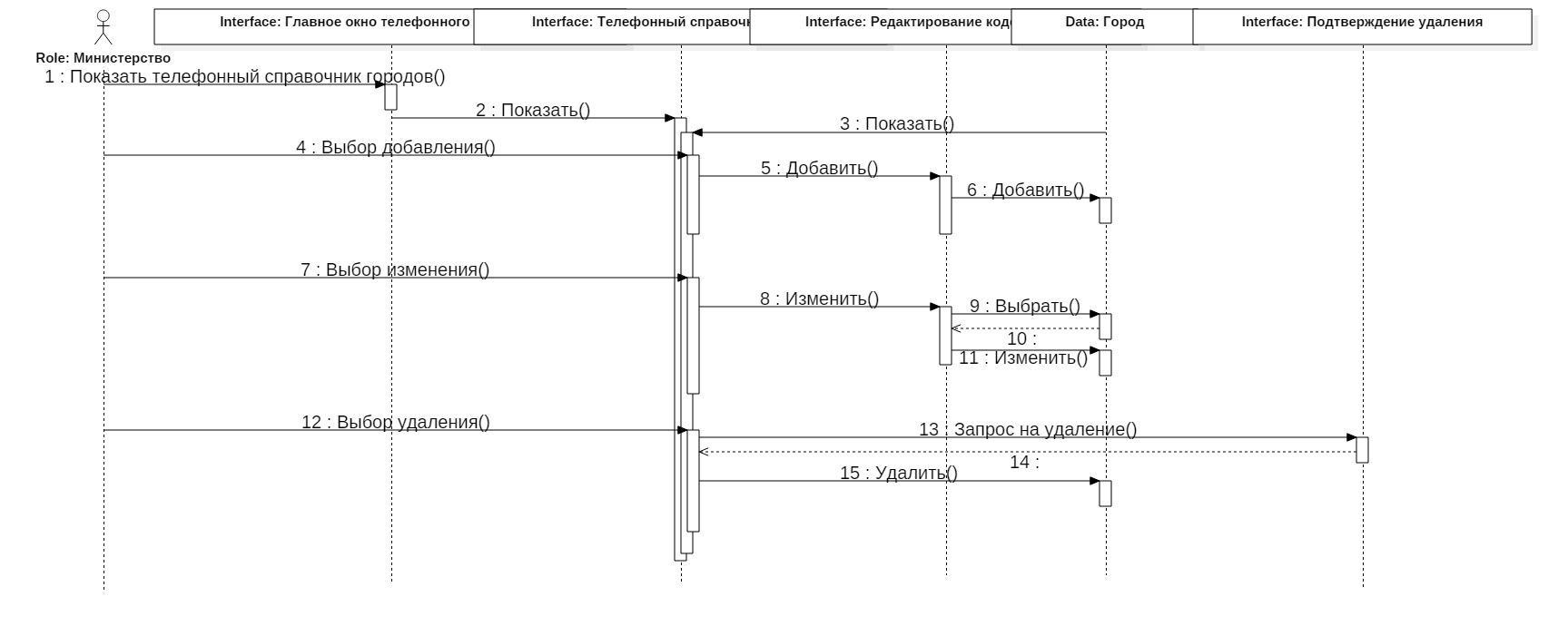


Рисунок 6 - Диаграмма последовательностей для прецедента «Ведение справочника кодов городов»

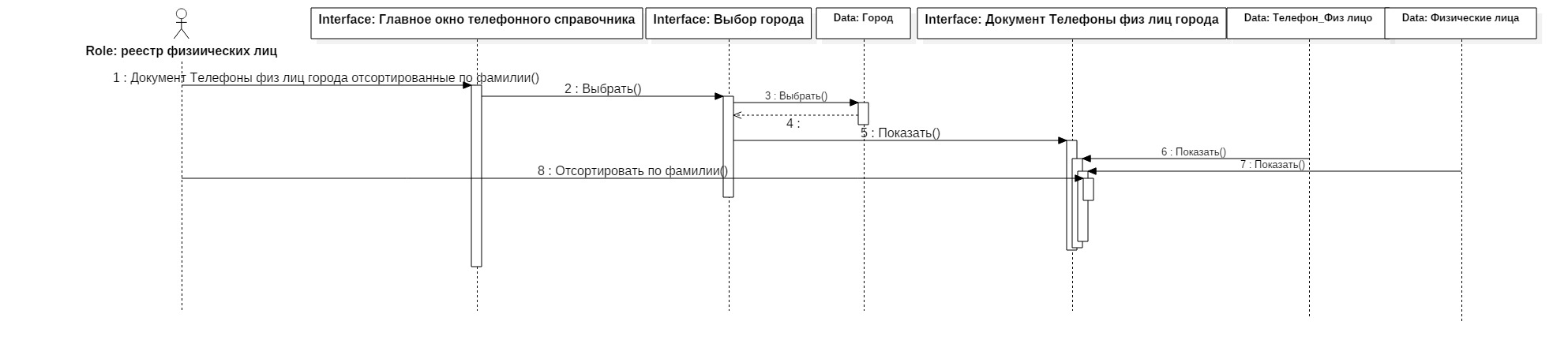


Рисунок 7 - Диаграмма последовательностей для прецедента «Документ : телефоны физ лиц города отсортированные по фамилии»

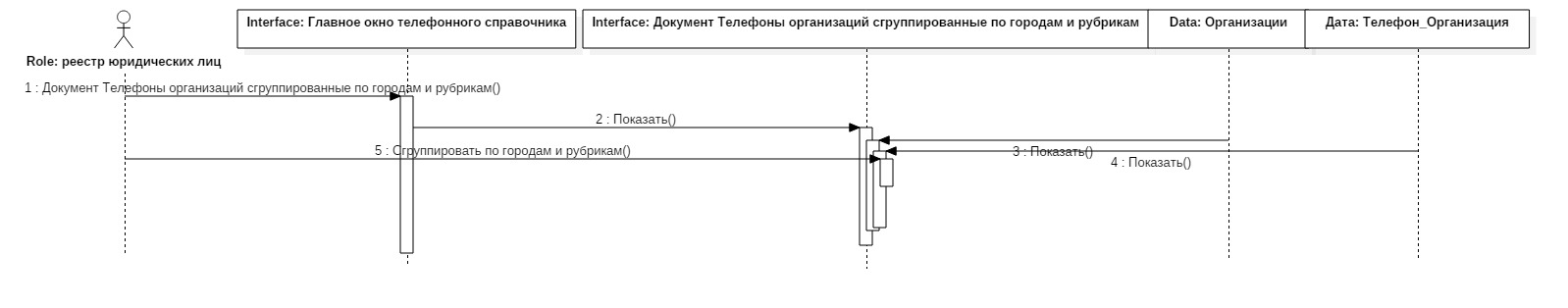


Рисунок 8 - Диаграмма последовательностей для прецедента «Документ : телефоны организаций сгруппированные по городам и рубрикам»

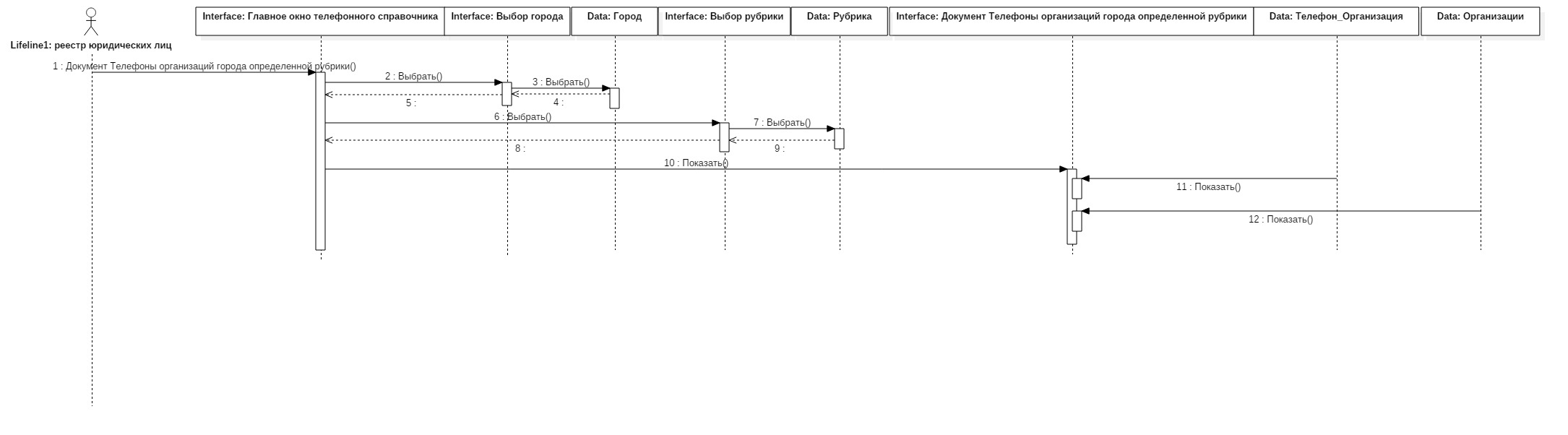


Рисунок 9 - Диаграмма последовательностей для прецедента «Документ : телефоны организаций города определенной рубрики»

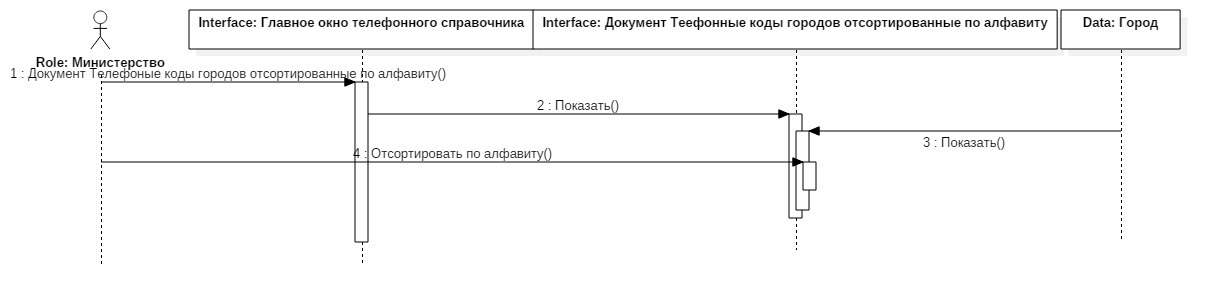


Рисунок 10 - Диаграмма последовательностей для прецедента «Документ : Коды городов отсортированных по алфавиту»

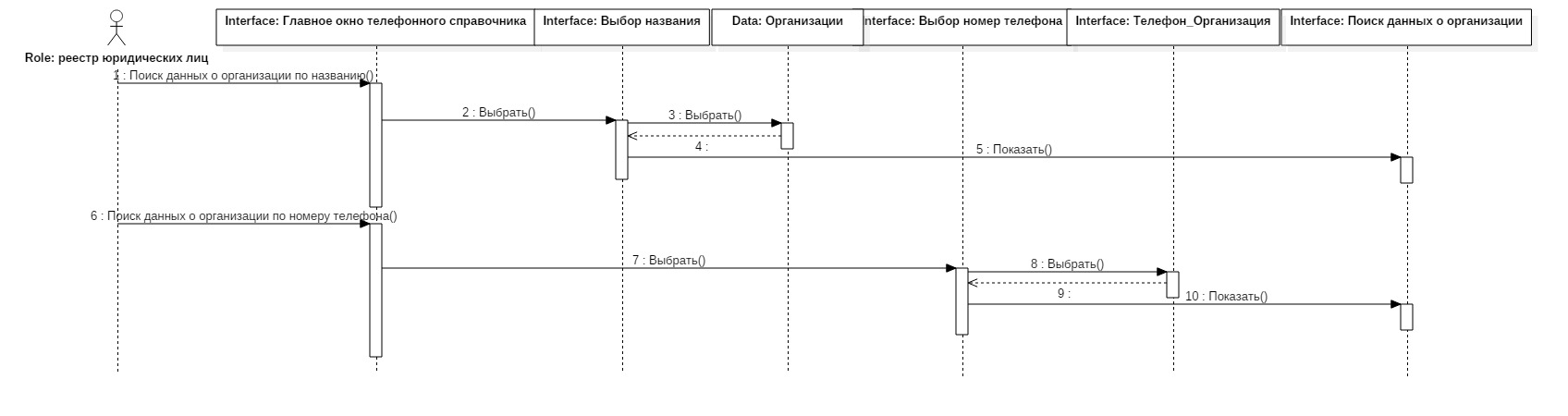


Рисунок 11 - Диаграмма последовательностей для прецедента «Поиск: организаций по названию и номеру телефона»

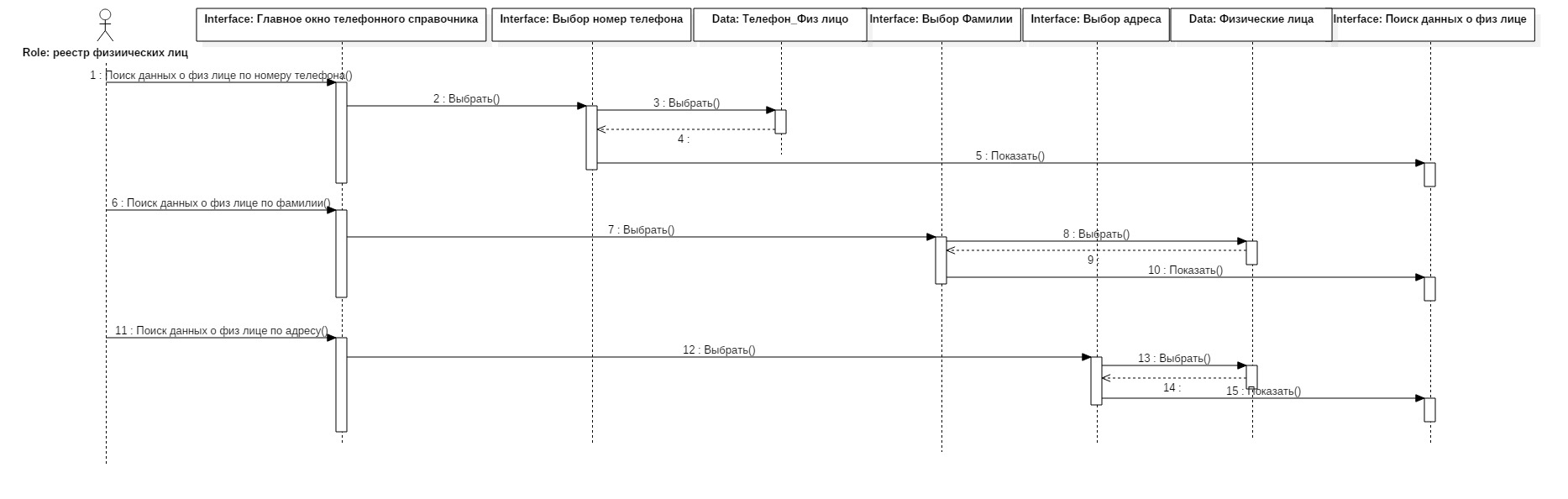


Рисунок 12 - Диаграмма последовательностей для прецедента «Поиск: физических лиц по фамилии, номеру телефона или адресу»

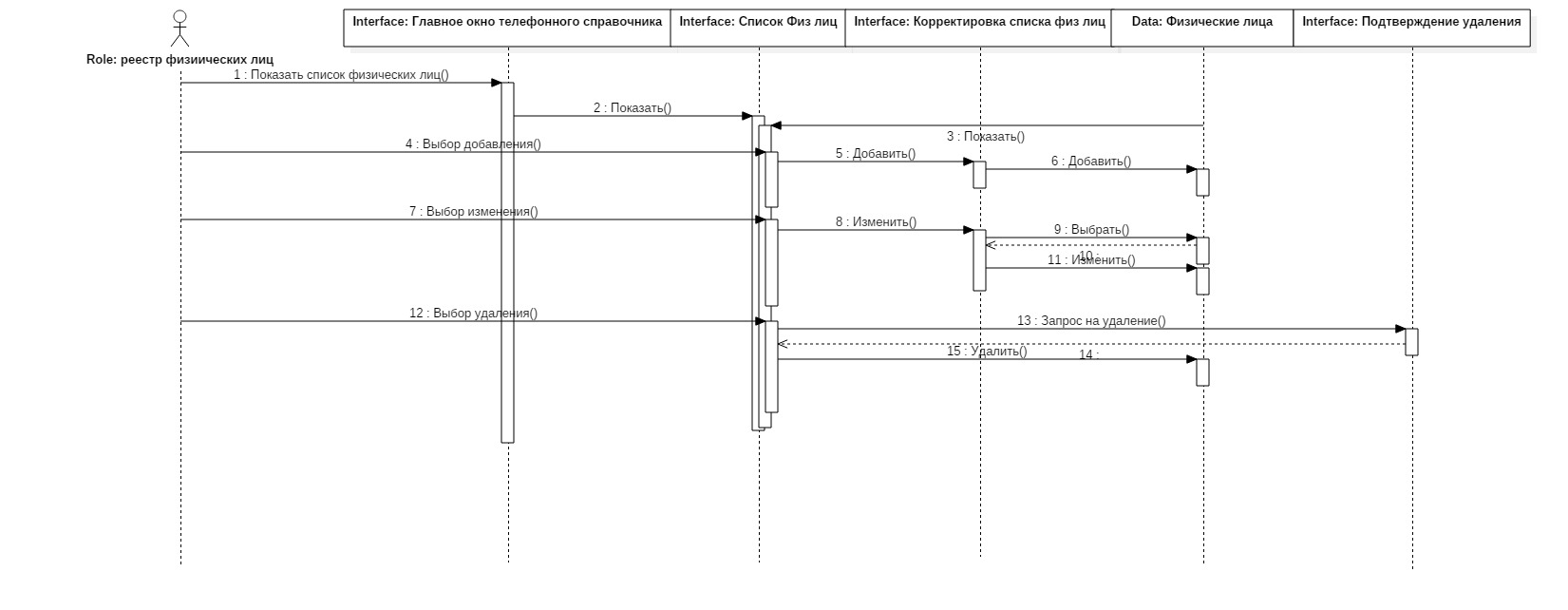


Рисунок 13 - Диаграмма последовательностей для прецедента «Ведение списка физических лиц»

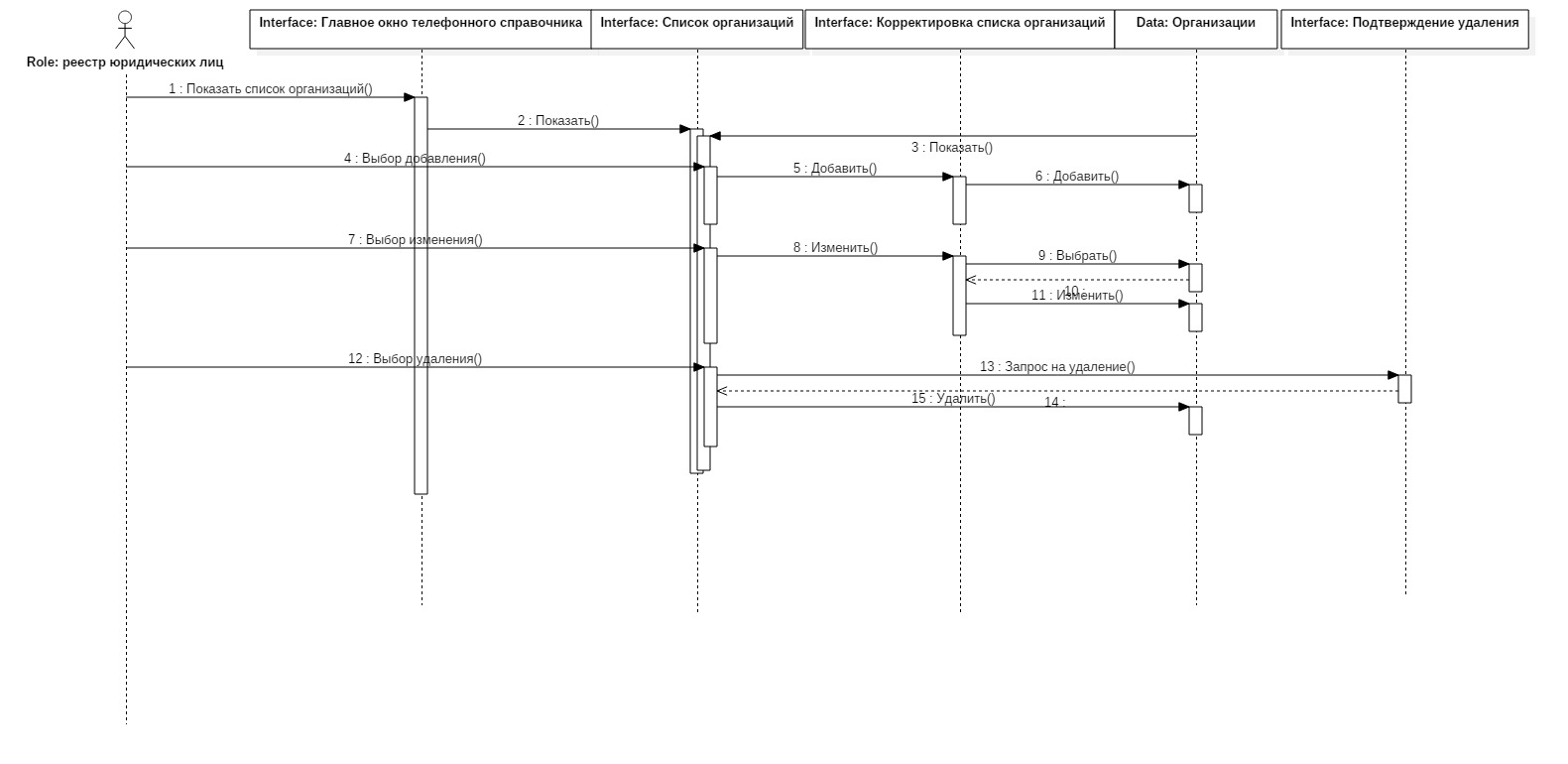


Рисунок 14 - Диаграмма последовательностей для прецедента «Ведение списка организаций»

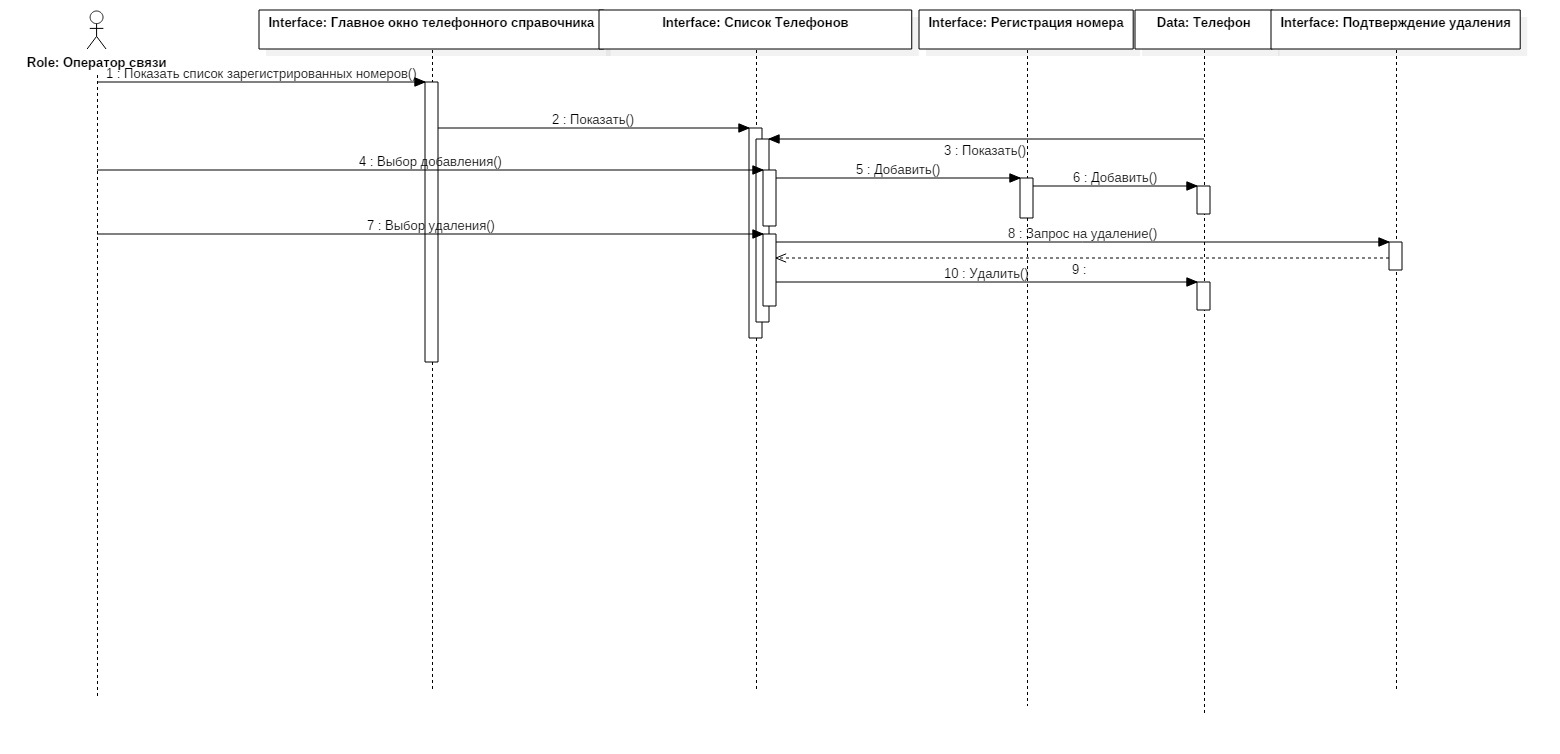


Рисунок 15 - Диаграмма последовательностей для прецедента «Регистрация номера телефона»

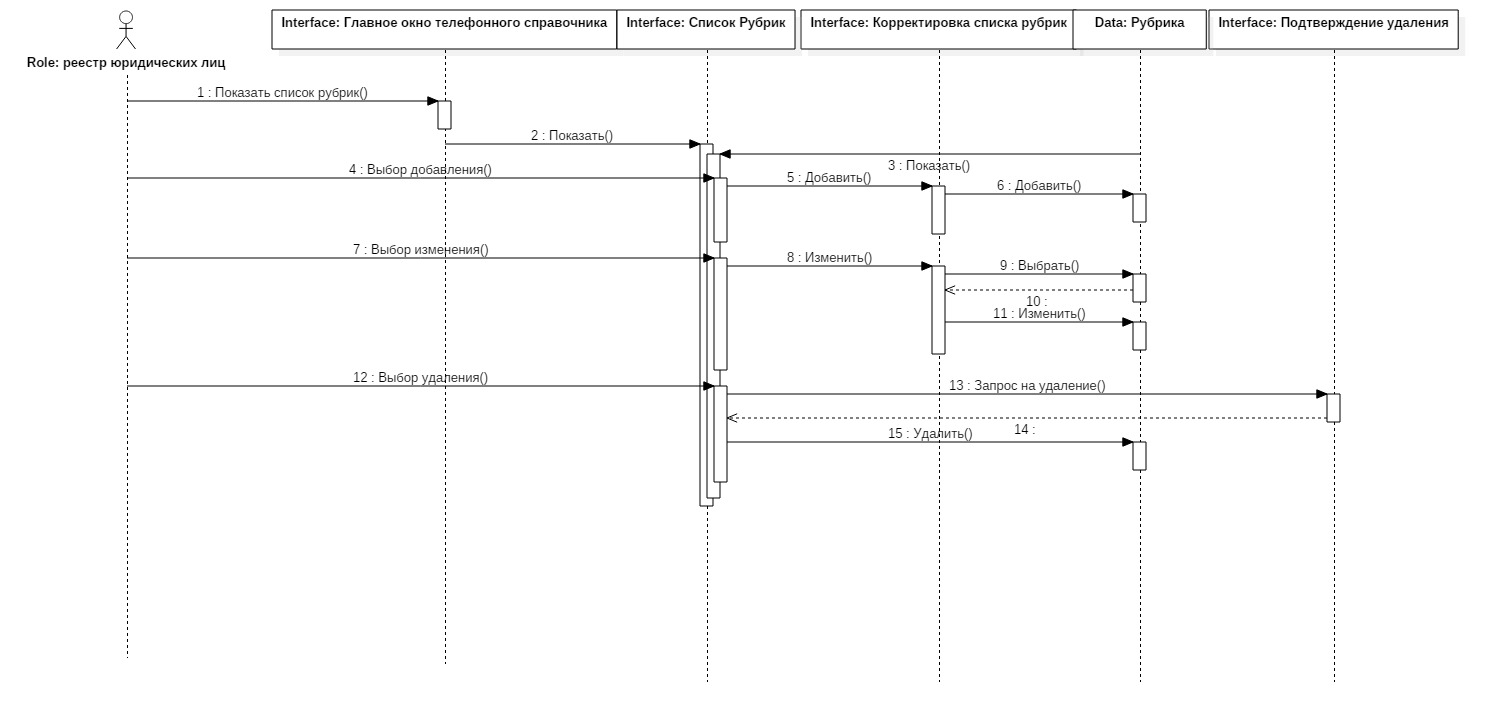


Рисунок 16 - Диаграмма последовательностей для прецедента «Ведение списка рубрик»

# Структура инфологической модели и результаты ее нормализации

В нашем случае выделяем 5 сущностей: «Города», «Организации», «Физические лица», «Телефоны», «Рубрики». Связи между ними изображены на рисунке 17.



Рисунок 17 – Инфологическая модель

Физическое лицо может иметь несколько телефонных номеров, например, один для работы, а другой для дома. При этом один и тот же номер телефона не может принадлежать нескольким физическим лицам. Рассмотрим связь между сущностями «Физическое лицо» и «Телефон» (рисунок 18):



Рисунок 18 – Связь между сущностями «Физическое лицо» и «Телефон»

**Правило 3:** Требуется построить 3 отношения: по одному на каждую сущность с соответствующими ключами и связующего отношения, ключом которого будет комбинация ключей двух сущностей.

* **Физическое лицо** (Паспортные данные(pk))
* **Телефон** (Номер телефона (pk))
* **Телефоны физ лиц** (Паспортные данные(fk,pk), Номер телефона (fk,pk))

Город может иметь лишь один единственный телефонный код. При этом телефонный код, может принадлежать лишь одному городу. Рассмотрим связь между сущностями «Город» и «Телефон» (рисунок 19):



Рисунок 19 – Связь между сущностями «Город» и «Телефон»

**Правило 2:** Требуется построить 2 отношения. Каждому отношению соответствует 1 сущность, ключ является ключом соответствующей сущности, ключ второй сущности добавляется в качестве атрибута в первое отношение (это внешний ключ)

* **Город** (название (pk), страна (pk), код города (fk))
* **Телефон** (номер телефона (pk))

Организация может иметь несколько номеров телефонов, например, один номер от отдела закупок, другой номер – для отдела продаж, а третий – это номер главного директора. При этом один и тот же номер телефона может принадлежать только одной организации. Рассмотрим связь между сущностями «Организация» и «Телефон» на рисунке 20.



Рисунок 20 – Связь между сущностями «Организация» и «Телефон»

**Правило 3:** Требуется построить 3 отношения: по одному на каждую сущность с соответствующими ключами и связующего отношения, ключом которого будет комбинация ключей двух сущностей.

* **Организация** (Название организации (pk))
* **Телефон**(Номер телефона(pk))
* **Телефоны организаций** (Название организации (pk,fk), Номер телефона (pk,fk))

Организация может принадлежать определенной конкретной рубрике. При этом к одной рубрике может относиться несколько организаций. Рассмотрим связь между сущностями «Организация» и «Рубрика» на рисунке 21.



Рисунок 21 – Связь между сущностями «Организация» и «Рубрика»

**Правило 4:** достаточным является использование 2х отношений, соответствующих сущностям, при чем ключами будут ключи сущностей, а дополнительно ключ первой сущности должен быть атрибутом второго отношения.

* **Организация** (Название (pk),Рубрика (fk))
* **Рубрика** (Название рубрики (pk))

Организация может быть расположена только в пределах одного города, при этом в одном городе может быть несколько организаций. Рассмотрим связь между сущностями «Организация» и «Город» на рисунке 22.



Рисунок 22 - Связь между сущностями «Организация» и «Город»

**Правило 4:** достаточным является использование 2х отношений, соответствующих сущностям, при чем ключами будут ключи сущностей, а дополнительно ключ первой сущности должен быть атрибутом второго отношения.

* **Организация** (Название организации(pk), Город(fk), Страна(fk))
* **Город** (Название города(pk), страна (pk))

Физическое лицо может быть зарегистрировано в одном городе, при этом в одном городе может быть зарегистрировано несколько физических лиц. Рассмотрим связь между сущностями «Физическое лицо» и «Город» на рисунке 23.



Рисунок 23 - Связь между сущностями «Физическое лицо» и «Город»

**Правило 4:** достаточным является использование 2х отношений, соответствующих сущностям, при чем ключами будут ключи сущностей, а дополнительно ключ первой сущности должен быть атрибутом второго отношения.

* **Город** (Название (pk), страна(pk))
* **Физические лицо (**Серия и номер паспорта (pk), Город (fk), Страна(fk))

**Нормальная форма №1 (НФ1)** требует, чтобы каждое поле таблицы было атомарным (неделимым) и отсутствовали повторяющиеся группы атрибутов.

* Так как нам придется телефоны физ лиц сортировать по фамилиям, то нам требуется разбить поле ФИО на Фамилию, Имя, Отчество
* Так как нам придется телефоны физ лиц группировать по городам, то адрес придется разбить на Страна, Город, Улица, Дом, Квартира
* Так как нам придется телефоны организаций группировать тоже по городам, то нужно адрес «Организация» тоже разбить на составляющие.

Результаты нормализации по НФ1 приведены на рисунке 24.



Рисунок 24 – Нормальная форма №1

**Нормальная форма №2 (НФ2):** Отношение находится во 2ой НФ, если оно удовлетворяет НФ1, первичный ключ pk не является избыточным (там нет лишних атрибутов) и все не ключевые атрибуты функционально полно зависят ПФЗ от составного pk.

Таким образом, для телефонного справочника городов, код города зависит от названия города и от страны:

Для телефонного справочник организаций название отдела зависит от названия организации и от номера телефона:

Для телефонного справочника физических лиц назначение телефона зависит от номера телефона и физического лица:

Под назначением я имею в виду: рабочий телефон или домашний или сотовый

**Нормальная форма №3 (НФ3):** Отношение находится в НФ3, если

* оно соответствует НФ2 и
* все не ключевые атрибуты взаимно независимы, то есть не существует транзитивных зависимостей между не ключевыми атрибутами:

Моя модель удовлетворяет НФ3.

**БКНФ :** Требует независимости ключевых атрибутов от не ключевых

**Нормальная форма № 4 (НФ4):** Требует отсутствия многозначных зависимостей между атрибутами.

После нормализации базы данных, можно приступать к построению логической и физической моделей.

# Модель (логическая и физическая) данных

## 3.1 **Описание использованного ПО**

**ERwin** (**AllFusion ERwin Data Modeler** ) — CASE-средство для проектирования и документирования баз данных, которое позволяет создавать, документировать и сопровождать базы данных, хранилища и витрины данных. Модели данных помогают визуализировать структуру данных, обеспечивая эффективный процесс организации, управления и администрирования таких аспектов деятельности предприятия, как уровень сложности данных, технологий баз данных и среды развертывания.

AllFusion ERwin Data Modeler предназначен для всех компаний, разрабатывающих и использующих базы данных, для администраторов баз данных, системных аналитиков, проектировщиков баз данных, разработчиков, руководителей проектов. AllFusion ERwin Data Modeler позволяет управлять данными в процессе корпоративных изменений, а также в условиях стремительно изменяющихся технологий.

AllFusion ERwin Data Modeler позволяет наглядно отображать сложные структуры данных. Удобная в использовании графическая среда AllFusion ERwin Data Modeler упрощает разработку базы данных и автоматизирует множество трудоёмких задач, уменьшая сроки создания высококачественных и высокопроизводительных транзакционных баз данных и хранилищ данных. Данное решение улучшает коммуникацию организации, обеспечивая совместную работу администраторов и разработчиков баз данных, многократное использование модели, а также наглядное представление комплексных активов данных в удобном для понимания и обслуживания формате.

## 3.2 Построение логической модели

Логическая модель является основой базы данных, она должна отображать взаимосвязи между реляционными таблицами. Это набор схем отношений, обычно с указанием первичных ключей, а также «связей» между отношениями, представляющих собой внешние ключи. Логическая модель описывает всю базу данных - как единое целое (см. рисунок 25).



Рисунок 25 - Логическая модель

Для номера дома выбран тип данных VARCHAR (строка) потому что иногда пишут «дом 183а»

Подразумевается, что организация расположена в одном городе и филиалов в других городах у нее нет. А телефонов у одной организации много, так как много различных отделов организации (отдел работы с кадрами, отдел бухгалтерии и т.п.).

## 3.3 **Построение физической модели**

Физическая модель содержит всю информацию, необходимую для реализации конкретной БД, а также способ и средства организации эффективного доступа к ним. (см. рисунок 26).



Рисунок 26 - Физическая модель

После создания физической модели, сгенерируем скрипт для создания базы данных. Получим:

/\*Создание таблицы Рубрика\*/

CREATE TABLE Rubrics (

ID\_rubric INTEGER NOT NULL,

Name\_of\_rubric VARCHAR(30) NOT NULL,

PRIMARY KEY (ID\_rubric) /\* установка первичного ключа \*/

);

/\*---------------------------------------\*/

/\*Создание таблицы Телефон\*/

CREATE TABLE Phones (

ID\_number INTEGER NOT NULL,

Phone\_number INTEGER NOT NULL,

Minutes INTEGER NOT NULL,

Tariff NUMERIC(4,2) NOT NULL,

Pay INTEGER NOT NULL,

PRIMARY KEY (ID\_number) /\* установка первичного ключа \*/

);

/\*---------------------------------------\*/

/\*Создание таблицы Город\*/

CREATE TABLE Town (

ID\_number INTEGER NOT NULL,

ID\_town INTEGER NOT NULL,

Town\_name VARCHAR(30) NOT NULL,

Country VARCHAR(20),

PRIMARY KEY (ID\_town), /\* установка первичного ключа \*/

FOREIGN KEY (ID\_number) REFERENCES Phones /\* установка внешнего ключа для связи с таблицей Телефоны \*/

);

/\*---------------------------------------\*/

/\*Создание таблицы Физические лица\*/

CREATE TABLE Person (

ID\_person INTEGER NOT NULL,

Serias\_and\_number\_passport VARCHAR(20) NOT NULL,

Surname VARCHAR(20) NOT NULL,

Name VARCHAR(20) NOT NULL,

Patronymic VARCHAR(20) NOT NULL,

Street VARCHAR(30) NOT NULL,

House VARCHAR(10) NOT NULL,

ID\_town INTEGER NOT NULL,

PRIMARY KEY (ID\_person), /\* установка первичного ключа \*/

FOREIGN KEY (ID\_town) REFERENCES Town

/\* установка внешнего ключа для связи с таблицей Город \*/

);

/\*---------------------------------------\*/

/\*Создание таблицы Телефоны\_Физ\_лиц\*/

CREATE TABLE Person\_Phones (

ID\_number\_of\_person INTEGER NOT NULL,

ID\_person INTEGER NOT NULL,

ID\_number INTEGER NOT NULL,

Phone\_assignment VARCHAR(20),

PRIMARY KEY (ID\_number\_of\_person),

/\* установка первичного ключа \*/

FOREIGN KEY (ID\_person) REFERENCES Person, /\* установка внешнего ключа для связи с таблицей «Физ лица»\*/

FOREIGN KEY (ID\_number) REFERENCES Phones

/\* установка внешнего ключа для связи с таблицей «Телефон»\*/

);

/\*---------------------------------------\*/

/\*Создание таблицы Организации\*/

CREATE TABLE Organization (

ID\_organization INTEGER NOT NULL,

Name\_of\_organization VARCHAR(20) NOT NULL,

Street VARCHAR(30) NOT NULL,

House VARCHAR(10) NOT NULL,

ID\_rubric INTEGER NOT NULL,

ID\_town INTEGER NOT NULL,

PRIMARY KEY (ID\_organization),

/\* установка первичного ключа \*/

FOREIGN KEY (ID\_town) REFERENCES Town,

/\* установка внешнего ключа для связи с таблицей «Город» \*/

FOREIGN KEY (ID\_rubric) REFERENCES Rubrics /\* установка внешнего ключа для связи с таблицей «Рубрика»\*/

);

/\*---------------------------------------\*/

/\*Создание таблицы Телефоны Организации\*/

CREATE TABLE Organization\_Phones (

ID\_number\_of\_org INTEGER NOT NULL,

ID\_organization INTEGER NOT NULL,

ID\_number INTEGER NOT NULL,

Department\_name VARCHAR(20),

PRIMARY KEY (ID\_number\_of\_org), /\* установка первичного ключа \*/

FOREIGN KEY (ID\_number) REFERENCES Phones, /\* установка внешнего ключа для связи с таблице «Телефон»\*/

FOREIGN KEY (ID\_organization) REFERENCES Organization /\* установка внешнего ключа для связи с таблице «Телефон»\*/

);

Этот скрипт требуется доработать: добавить поля для вычислений, установить правила обновления данных в родительских таблицах, при изменении данных в дочерних таблицах. Также добавить записи, для указания значений по умолчанию. В результате редактирования получаем измененный sql скрипт:

/\*Создание таблицы Рубрика\*/

CREATE TABLE Rubrics (

ID\_rubric INTEGER NOT NULL,

Name\_of\_rubric VARCHAR(30) DEFAULT 'Без названия',

PRIMARY KEY (ID\_rubric) /\* установка первичного ключа \*/

);

/\*Создание индекса по ключу для таблицы Рубрика\*/

CREATE UNIQUE INDEX XPKRubrics ON Rubrics (

ID\_rubric

);

/\*---------------------------------------\*/

/\*Создание таблицы Организация\*/

CREATE TABLE Organization (

ID\_organization INTEGER NOT NULL,

Name\_of\_organization VARCHAR(20) DEFAULT 'Без названия',

Street VARCHAR(30) DEFAULT 'Улица',

House VARCHAR(10) DEFAULT 'Дом',

ID\_rubric INTEGER NOT NULL,

ID\_town INTEGER NOT NULL,

PRIMARY KEY (ID\_organization),

/\* установка первичного ключа \*/

FOREIGN KEY (ID\_town) REFERENCES Town

/\* установка внешнего ключа для связи с таблицей «Города» \*/

/\* нельзя удалить город с организациями\*/

ON DELETE NO ACTION

/\*каскадное изменение\*/

ON UPDATE CASCADE,

FOREIGN KEY (ID\_rubric) REFERENCES Rubrics

/\* установка внешнего ключа для связи с таблицей «Рубрика» \*/

/\* нельзя удалить рубрику с организациями\*/

ON DELETE NO ACTION

/\*каскадное изменение\*/

ON UPDATE CASCADE

);

/\*Создание индекса по ключу для таблицы Организация\*/

CREATE UNIQUE INDEX XPKOrganization ON Organization (

ID\_organization

);

/\*Создание индекса по внешнему ключу id\_rubric для таблицы Организация\*/

CREATE UNIQUE INDEX XIF1Organization ON Organization (

ID\_rubric

);

/\*Создание индекса по внешнему ключу id\_town для таблицы Организация\*/

CREATE UNIQUE INDEX XIF2Organization ON Organization (

ID\_town

);

/\*---------------------------------------\*/

/\*Создание таблицы Физическое лицо\*/

CREATE TABLE Person (

ID\_person INTEGER NOT NULL,

Serias\_and\_number\_passport VARCHAR(20) NOT NULL,

Surname VARCHAR(20) DEFAULT 'Иванов',

Name VARCHAR(20) DEFAULT 'Иван',

Patronymic VARCHAR(20) DEFAULT 'Иванович',

Street VARCHAR(30) DEFAULT 'Улица',

House VARCHAR(10) DEFAULT 'Дом',

ID\_town INTEGER NOT NULL,

PRIMARY KEY (ID\_person),

FOREIGN KEY (ID\_town)

REFERENCES Town

/\* нельзя удалить город с физическими лицами\*/

ON DELETE NO ACTION

ON UPDATE CASCADE

);

CREATE UNIQUE INDEX XPKPerson ON Person (

ID\_person

);

CREATE UNIQUE INDEX XIF3Person ON Person (

ID\_town

);

/\*---------------------------------------\*/

/\*Создание таблицы телефоны\*/

CREATE TABLE Phones (

ID\_number INTEGER NOT NULL,

Phone\_number INTEGER NOT NULL,

Minutes INTEGER DEFAULT 0,

Tariff NUMERIC(4,2) DEFAULT 0,

Pay COMPUTED BY (Minutes \* Tarrif),

PRIMARY KEY (ID\_number)

);

CREATE UNIQUE INDEX XPKPhones ON Phones (

ID\_number

);

/\*---------------------------------------\*/

/\*Создание таблицы Физическое Телефоны организаций\*/

CREATE TABLE Organization\_Phones (

ID\_number\_of\_org INTEGER NOT NULL,

ID\_organization INTEGER NOT NULL,

ID\_number INTEGER NOT NULL,

Department\_name VARCHAR(20) DEFAULT 'Без названия',

PRIMARY KEY (ID\_number\_of\_org),

FOREIGN KEY (ID\_number)

REFERENCES Phones

ON DELETE CASCADE

ON UPDATE CASCADE,

FOREIGN KEY (ID\_organization)

REFERENCES Organization

ON DELETE CASCADE

ON UPDATE CASCADE

);

CREATE UNIQUE INDEX XPKOrganization\_Phones ON Organization\_Phones (

ID\_number\_of\_org

);

CREATE UNIQUE INDEX XIF4Organization\_Phones ON Organization\_Phones (

ID\_organization

);

CREATE UNIQUE INDEX XIF5Organization\_Phones ON Organization\_Phones (

ID\_number\_of\_org

);

CREATE TABLE Person\_Phones (

ID\_number\_of\_person INTEGER NOT NULL,

ID\_person INTEGER NOT NULL,

ID\_number INTEGER NOT NULL,

Phone\_assignment VARCHAR(20) DEFAULT 'Рабочий',

PRIMARY KEY (ID\_number\_of\_person),

FOREIGN KEY (ID\_person)

REFERENCES Person

ON DELETE CASCADE

ON UPDATE CASCADE,

FOREIGN KEY (ID\_number)

REFERENCES Phones

ON DELETE CASCADE

ON UPDATE CASCADE

);

CREATE UNIQUE INDEX XPKPerson\_Phones ON Person\_Phones (

ID\_number\_of\_person

);

CREATE UNIQUE INDEX XIF6Person\_Phones ON Person\_Phones (

ID\_person

);

CREATE UNIQUE INDEX XIF7Person\_Phones ON Person\_Phones (

ID\_number

);

CREATE TABLE Town (

ID\_town INTEGER NOT NULL,

Town\_name VARCHAR(30) DEFAULT 'Город без названия',

ID\_number INTEGER NOT NULL,

Country VARCHAR(20) DEFAULT 'Страна без названия',

PRIMARY KEY (ID\_town),

FOREIGN KEY (ID\_number)

REFERENCES Phones

ON DELETE NO ACTION

ON UPDATE CASCADE

);

CREATE UNIQUE INDEX XPKTown ON Town (

ID\_town

);

CREATE UNIQUE INDEX XIF8Town ON Town (

ID\_number

);

# Описание базы данных на сервере

IBExpert — GUI-оболочка, предназначенная для разработки и администрирования баз данных InterBase и Firebird, а также для выбора и изменения данных, хранящихся в базах.

Как основные достоинства IBExpert разработчики указывают:

* поддержка InterBase версий 4.х, 5.х, 6.х, 7.х, 2007 и 2009; Firebird 1.х, 2.x, 3.x; Yaffil 1.х;
* работа одновременно с несколькими базами данных;
* отдельные редакторы для всех объектов БД с синтаксической подсветкой;
* мощный SQL-редактор с историей запросов и возможностью их фонового выполнения;
* автозавершение кода SQL (название таблиц, полей, и т. п.);
* отладчик хранимых процедур и триггеров;
* поиск в метаданных;
* полное и частичное извлечение данных и метаданных;
* анализатор зависимостей объектов баз данных;
* отчёты по метаданным;
* менеджеры пользователей и пользовательских привилегий;
* экспорт данных в различные форматы.

IBExpert обладает множеством облегчающих работу компонентов: визуальный редактор для всех объектов базы данных, редактор SQL и исполнитель скриптов, отладчик для хранимых процедур и триггеров, построитель области, инструмент для импорта данных из различных источников, собственный скриптовый язык, а также дизайнер баз данных и т. д.

## 4.1 Создание базы данных программе IBExpert

При вызове на выполнение IBExpert появляется окно, в котором, прежде всего, следует выполнить пункт меню - Создать базу данных, необходимо задать путь для размещения файла базы данных. Проводим регистрацию базы данных и выбираем версию сервера. После чего в указанном каталоге будет создан файл базы данных с расширением gdb.

Созадание базы данных проиллюстрировано на рисунке 27. При регистрации использовалась версия сервера InterBase 7.5, пароль от базы данных «masterkey».

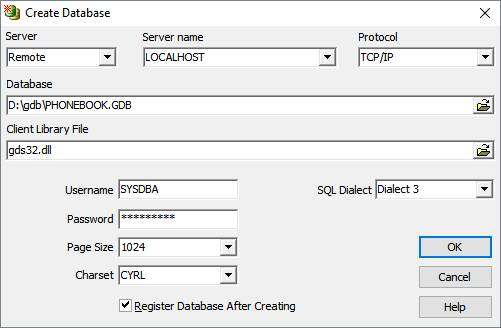
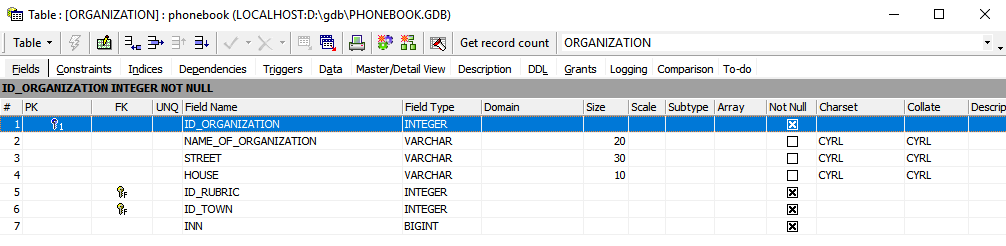
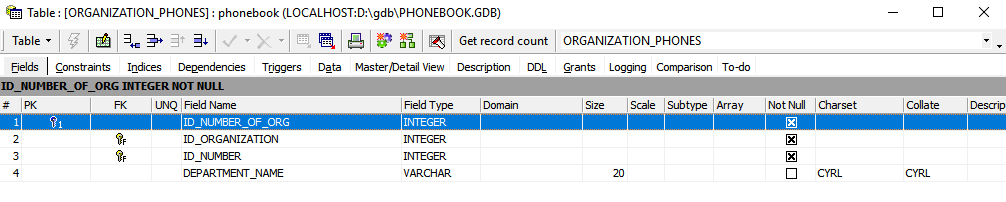
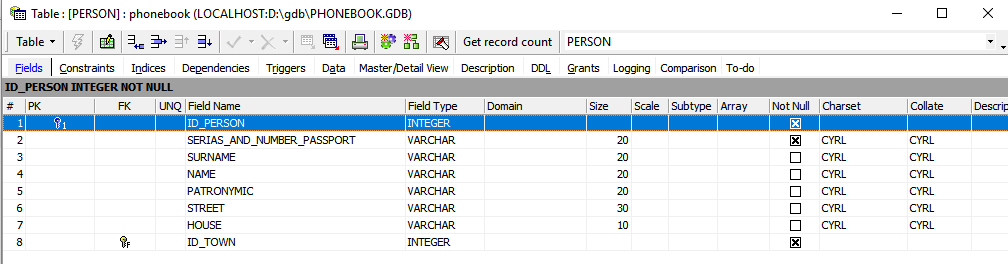


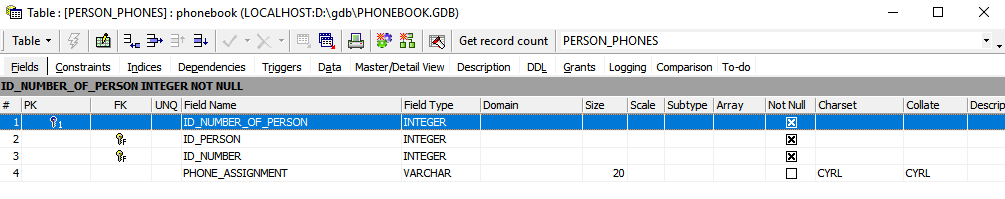
Рисунок 27 – Создание базы данных

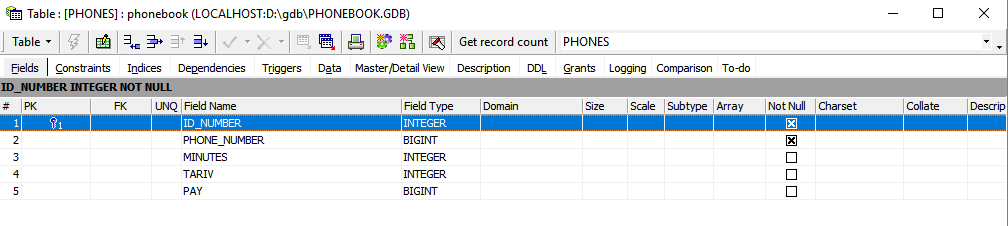
После того, как БД создана, с ней можно установить связь. Также следует выполнить сгенерированный ранее скрипт БД, для этого следует загрузить файл \*.sql и выполнить его. После успешного выполнения операторов SQL скрипта создаются указанные в нем объекты (рисунок 28).

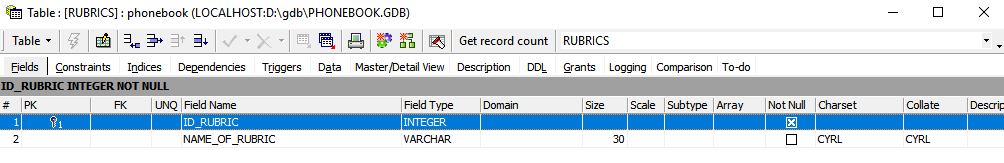












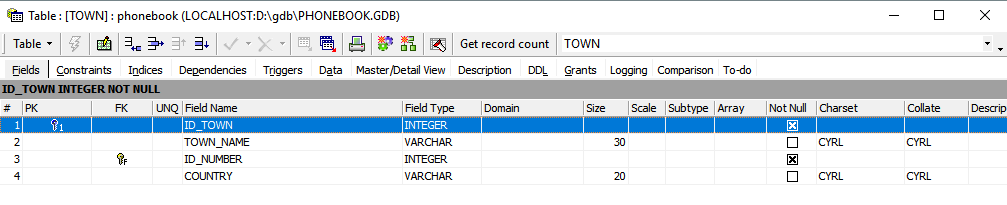


Рисунок 28 – Созданные таблицы на сервере

## 4.2 Создание исключений в программе IBExpert

**Исключение**– это именованное сообщение, которое вызывается из хранимой процедуры или триггера при возникновении какой-либо ситуации. Созданные исключения приведены на рисунке 29.

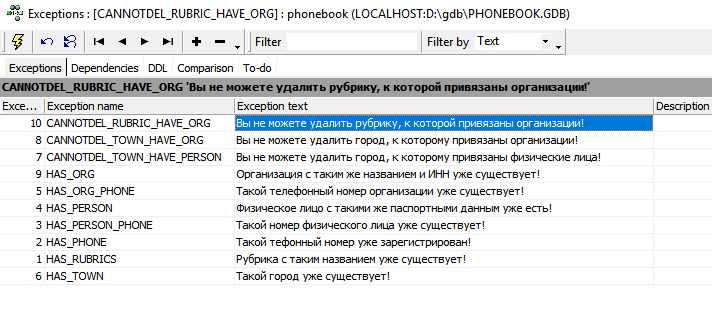


Рисунок 29 - Исключения

## 4.5 Процедуры, созданные в программе IBExpert.

Для созданных таблиц необходимо создать хранимые процедуры (добавление, редактирование и удаление данных)

**Хранимая процедура** – это модуль, написанный на процедурном языке сервера БД и хранящийся в базе данных как метаданные (то есть данные о данных). Хранимую процедуру можно вызвать из приложения.

Существует 2 разновидности хранимых процедур:

* Процедуры выбора – возвращают более одного значения
* Процедура действия – вообще может не возвращать данных и используется для реализации каких-либо действий

В нашей базе данных будут следующие процедуры:

* Процедура добавления записи в таблицу «Организации»
* Процедура добавления записи в таблицу «Физические лица»
* Процедура удаления записи из таблицы «Город»
* Процедура добавления номера телефона в справочник
* Процедура добавления рубрики
* Процедура добавления записи в таблицу «Город»
* Процедура добавления записи в таблицу «Телефоны организаций»
* Процедура добавления записи в таблицу «Телефоны физических лиц»
* Процедура обновления данных в таблице «Города»
* Процедура обновления данных в таблице «Рубрики»
* Процедура обновления данных в таблице «Телефоны»
* Процедура обновления данных в таблице «Телефоны физических лиц»
* Процедура обновления данных в таблице «Физические лица»
* Процедура обновления данных в таблице «Телефоны организаций»
* Процедура обновления данных в таблице «Организации»
* Процедура выходного документа: Телефонные коды городов, отсортированные по алфавиту
* Процедура выходного документа: Телефоны физических лиц города, отсортированные по фамилии.
* Процедура выходного документа: Телефоны всех организаций городов, сгруппированные по городам и рубрикам.
* Процедура выходного документа: Телефоны организаций города определенной рубрики.

### Процедура добавления записи в таблицу «Организации» INS\_ORG

Процедура проверяет, нет ли в таблице организации с таким ИНН и с таким же названием организации, если нет, то в таблицу добавляется запись об организации, если подобная запись есть, то вызывается исключение has\_org, которое выводит на экран сообщение: «Такая организация уже существует!». То есть в данном случае подразумевается, что может быть так, что у двух разных по названию организаций будет одинаковое ИНН, что будет означает, что одна из данных организаций является дочерней.

Согласно п. 7 ст. 84 НК РФ идентификационный номер налогоплательщика (ИНН) присваивается именно налогоплательщику, т.е. самой организации (юридическому лицу). Приложением N 1 к приказу МНС РФ от 03.03.2004 N БГ-3-09/178 утвержден Порядок и условия присвоения, применения, а также изменения ИНН и форм документов, используемых при постановке на учет, снятии с учета юридических и физических лиц (далее - Порядок).   
Согласно п. 2.1.4 Порядка код причины постановки на учет (КПП) присваивается организации в том числе при постановке на учет по месту нахождения обособленных подразделений на основании: - заявления о постановке на учет; - одновременно представленных в одном экземпляре заверенных в установленном порядке копий свидетельства о постановке на учет в налоговом органе организации по месту ее нахождения; - документов, подтверждающих создание обособленного подразделения (при их наличии).

Из вышеизложенного следует, что ИНН присваивается организации в целом, т.е. ИНН филиала и головной компании соответствуют друг другу. При этом КПП филиала может отличаться от КПП самой головной организации (юридического лица).

CREATE PROCEDURE INS\_ORG (

param\_org\_name varchar(20),

param\_street varchar(30),

param\_house varchar(10),

param\_id\_rubric integer,

param\_id\_town integer,

param\_inn integer)

as

declare variable k integer; /\* count of org where eaqul inn \*/

begin

select count(\*) from organization where INN=:param\_inn and NAME\_OF\_ORGANIZATION=:param\_org\_name INTO K;

/\*or with such inn have not in table\*/

if (:K=0) then

insert into organization (ID\_ORGANIZATION, NAME\_OF\_ORGANIZATION, STREET, HOUSE, ID\_RUBRIC, ID\_TOWN, INN)

values (gen\_id(ORG\_GEN,1), :param\_org\_name, :param\_street, :param\_house, :param\_id\_rubric, :param\_id\_town, :param\_inn);

else

exception has\_org;

end

### Процедура добавления записи в таблицу «Физические лица» INS\_PERSON

Процедура проверяет, нет ли в таблице физического лица с такими же паспортными данными, если нет, то в таблицу добавляется запись об физическом лице, если подобная запись есть, то вызывается исключение has\_person, которое выводит на экран сообщение: «Такое физическое лицо уже существует!».

CREATE PROCEDURE INS\_PERSON (

param\_passport\_number varchar(20),

param\_surname varchar(20),

param\_name varchar(20),

param\_patronymic varchar(20),

param\_street varchar(30),

param\_house varchar(10),

param\_id\_town integer)

as

declare variable k integer; /\* count of person where eaqul passpot number \*/

begin

select count(\*) from PERSON where SERIAS\_AND\_NUMBER\_PASSPORT=:param\_passport\_number INTO K;

/\*or with such inn have not in table\*/

if (:K=0) then

insert into PERSON (ID\_PERSON, SERIAS\_AND\_NUMBER\_PASSPORT, SURNAME, NAME, PATRONYMIC, STREET, HOUSE, ID\_TOWN)

values (gen\_id(PERSON\_GEN,1), :param\_passport\_number,:param\_surname, :param\_name, :param\_patronymic, :param\_street, :param\_house, :param\_id\_town);

else

exception has\_person;

end

### Процедура удаления записи из таблицы «Город» DEL\_TOWN

Здесь вызывается исключение CANNOTDEL\_TOWN\_HAVE\_PERSON, если в данном городе содержатся физические лица, а также вызывается исключение CANNOTDEL\_TOWN\_HAVE\_ORG, если в данном городе содержатся юридические лица (организации).

CREATE PROCEDURE DEL\_TOWN (

param\_id\_town integer)

as

declare variable k integer;

declare variable l integer;

begin

select count(\*) from person where ID\_TOWN=:param\_id\_town into :K;

if (:K=0) then

delete from TOWN where ID\_TOWN=:param\_id\_town;

else exception CANNOTDEL\_TOWN\_HAVE\_PERSON;

/\*-------------------------------------------------------------------\*/

select count(\*) from organization where ID\_TOWN=:param\_id\_town into :L;

if (:L=0) then

delete from TOWN where ID\_TOWN=:param\_id\_town;

else exception CANNOTDEL\_TOWN\_HAVE\_ORG;

end

### Процедура добавления номера телефона в справочник INS\_PHONE

Здесь выдается количество записей из таблицы «Телефонные номера», где номер телефона совпадает с тем номером, который хочет зарегистрировать пользователь. Если такой номер уже существует в таблице, то вызывается исключение HAS\_PHONE, иначе запись заносится в таблицу.

CREATE PROCEDURE INS\_PHONE (

param\_phone\_number integer,

param\_minutes integer,

param\_tariv integer)

as

declare variable k integer; /\* count of phones where eaqul phone\_number \*/

begin

select count(\*) from phones where phone\_number=:param\_phone\_number INTO K;

/\*phone with such number have not in table\*/

if (:K=0) then

insert into phones (id\_number, phone\_number, minutes, tariv)

values (gen\_id(PHONES\_GEN,1), :param\_phone\_number, :param\_minutes, :param\_tariv);

else

exception HAS\_PHONE;

end

### **Процедура добавления рубрики** INS\_RUBRIC

Здесь выдается количество записей из таблицы «Рубрики», где название рубрики совпадает с тем названием новой рубрики, которое хочет зарегистрировать пользователь. Если такая рубрика уже существует в таблице, то вызывается исключение HAS\_RUBRICS, иначе запись заносится в таблицу.

CREATE PROCEDURE INS\_RUBRICS (

param\_name\_of\_rubric varchar(30))

as

declare variable k integer; /\* count of rubrics where eaqul name\_of\_rubrics \*/

begin

select count(\*) from rubrics where name\_of\_rubric=:param\_name\_of\_rubric INTO K;

/\*rubrics with such name have not in table\*/

if (:K=0) then

insert into rubrics (id\_rubric, name\_of\_rubric)

values (gen\_id(RUBRICS\_GEN,1), :param\_name\_of\_rubric);

else

exception HAS\_RUBRICS;

end

### Процедура добавления записи в таблицу «Город» INS\_TOWN

Здесь выдается количество записей из таблицы «Город», где название города и страна совпадает с тем городом и страной, которые хочет зарегистрировать пользователь. Если такой город в такой стране уже существует в таблице, то вызывается исключение HAS\_TOWN, иначе запись заносится в таблицу. То есть здесь предусматривается, что два города с одинаковыми названиями могут находиться в разных странах.

CREATE PROCEDURE INS\_TOWN (

param\_town\_name varchar(30),

param\_id\_number integer,

param\_country varchar(20))

as

declare variable k integer; /\* count of town where eaqul town\_name and country \*/

begin

select count(\*) from town where town\_name=:param\_town\_name and country=:param\_country INTO K;

/\*town with such name have not in table\*/

if (:K=0) then

insert into town (id\_town, town\_name, id\_number, country)

values (gen\_id(TOWN\_GEN,1), :param\_town\_name, :param\_id\_number, :param\_country);

/\*values ('3', :param\_town\_name, :param\_id\_number, :param\_country); \*/

else

exception HAS\_TOWN;

end

### **Процедура добавления записи в таблицу «Телефоны организаций»** INS\_ORG\_PHONES

Если в таблице уже существует телефон для данной организации, то вызывается исключение HAS\_ORG\_PHONES, иначе запись добавляется в таблицу.

CREATE PROCEDURE INS\_ORG\_PHONES (

param\_id\_organization integer,

param\_id\_number integer,

param\_department\_name varchar(20))

as

declare variable k integer; /\* count of org\_phons where eaqul id\_org and id\_number \*/

begin

select count(\*) from organization\_phones

where id\_organization=:param\_id\_organization and id\_number=:param\_id\_number INTO K;

/\*org\_phones with such id\_org and id\_number have not in table\*/

if (:K=0) then

insert into organization\_phones (id\_number\_of\_org, id\_organization, id\_number, department\_name)

values (gen\_id(ORG\_PNONES\_GEN,1), :param\_id\_organization, :param\_id\_number, :param\_department\_name);

else

exception HAS\_ORG\_PHONE;

end

### Процедура добавления записи в таблицу «Телефоны физических лиц» INS\_PERSON\_PHONES

Если в таблице уже существует телефон физического лица, то вызывается исключение HAS\_PERSON\_PHONES, иначе запись добавляется в таблицу.

CREATE PROCEDURE INS\_PERSON\_PHONES (

param\_id\_person integer,

param\_id\_number integer,

param\_phone\_assignment varchar(20))

as

declare variable k integer; /\* count of person\_phones where eaqul id\_person and id\_number \*/

begin

select count(\*) from person\_phones

where id\_person=:param\_id\_person and id\_number=:param\_id\_number INTO K;

/\*person\_phones with such id\_person and id\_number have not in table\*/

if (:K=0) then

insert into person\_phones (id\_number\_of\_person, id\_person, id\_number, phone\_assignment)

values (gen\_id(PERSON\_PHONES\_GEN,1), :param\_id\_person, :param\_id\_number, :param\_phone\_assignment);

else

exception HAS\_PERSON\_PHONE;

end

### **Процедура обновления данных в таблице «Города»** UPD\_TOWN

CREATE PROCEDURE UPD\_TOWN (

p\_id\_town integer,

p\_town\_name varchar(30),

p\_id\_number integer,

p\_country varchar(20))

as

declare variable k integer;

begin

select count(\*) from TOWN

Where town\_name=:p\_town\_name and country=:p\_country into :k;

if (:k=0) then

UPDATE TOWN

SET town\_name=:p\_town\_name, id\_number=:p\_id\_number, country=:p\_country

WHERE id\_town=:p\_id\_town;

else

exception HAS\_TOWN;

suspend;

end

### Процедура обновления данных в таблице «Рубрики» UPD\_RUBRICS

CREATE PROCEDURE UPD\_RUBRICS (

pid\_rubric integer,

prubric\_name varchar(30))

as

declare variable k integer;

begin

select count(\*) from RUBRICS

Where name\_of\_rubric=:prubric\_name into :k;

if (:k=0) then

UPDATE RUBRICS

SET name\_of\_rubric=:prubric\_name

WHERE id\_rubric=:pid\_rubric;

else

exception HAS\_RUBRICS;

suspend;

end

### Процедура обновления данных в таблице «Телефоны» UPD\_PHONE

CREATE PROCEDURE UPD\_PHONE (

p\_id\_number integer,

p\_phone\_number numeric(11,0),

p\_minutes integer,

p\_tariv integer)

as

declare variable k integer;

begin

/\*Если записей с таким же телефонным номером не было найдено то просто обновляем\*/

/\*Если записей с таким же телефонным номером не было найдено 1 то возможные ситуации:\*/

/\*Пользователь изменил в записи номер телефон на тот, который уже есть в таблице\*/

/\*Пользователь изменил у текущий записи другие поля, но не номер телефона\*/

select count(\*) from PHONES

Where phone\_number=:p\_phone\_number and id\_number<>:p\_id\_number into :k;

if (:k=0) then

UPDATE PHONES

SET phone\_number=:p\_phone\_number,minutes=:p\_minutes, tariv=:p\_tariv

WHERE id\_number=:p\_id\_number;

else

exception HAS\_PHONE;

suspend;/\*Возврат НД по одной записи по мере обработки\*/

end

### Процедура обновления данных в таблице «Телефоны физических лиц» UPD\_PERSON\_PHONE

CREATE PROCEDURE UPD\_PERSON\_PHONES (

p\_id\_number\_of\_person integer,

p\_id\_person integer,

p\_id\_number integer,

p\_phone\_assignment varchar(20))

as

declare variable k integer;

begin

select count(\*) from PERSON\_PHONES

Where id\_person=:p\_id\_person and id\_number=:p\_id\_number into :k;

if (:k=0) then

UPDATE PERSON\_PHONES

SET id\_person=:p\_id\_person, id\_number=:p\_id\_number, phone\_assignment=:p\_phone\_assignment

WHERE id\_number\_of\_person=:p\_id\_number\_of\_person;

else

exception HAS\_PERSON\_PHONE;

suspend;

end

### Процедура обновления данных в таблице «Физические лица» UPD\_PERSON

CREATE PROCEDURE UPD\_PERSON (

p\_id\_person integer,

p\_serias\_and\_number\_passport varchar(20),

p\_surname varchar(20),

p\_name varchar(20),

p\_patronymic varchar(20),

p\_street varchar(30),

p\_house varchar(10),

p\_id\_town integer)

as

declare variable k integer;

begin

select count(\*) from PERSON

Where serias\_and\_number\_passport=:p\_serias\_and\_number\_passport AND

id\_person<>:p\_id\_person into :k;

if (:k=0) then

UPDATE PERSON

SET serias\_and\_number\_passport=:p\_serias\_and\_number\_passport,

surname=:p\_surname,

name=:p\_name,

patronymic=:p\_patronymic,

street=:p\_street,

house=:p\_house,

id\_town=:p\_id\_town

WHERE id\_person=:p\_id\_person;

else

exception HAS\_PERSON;

suspend;

end

### Процедура обновления данных в таблице «Телефоны организаций» UPD\_ORGANIZATION\_PHONES

CREATE PROCEDURE UPD\_ORGANIZATION\_PHONES (

p\_id\_number\_of\_org integer,

p\_id\_organization integer,

p\_id\_number integer,

p\_department\_name varchar(20))

as

declare variable k integer;

begin

select count(\*) from ORGANIZATION\_PHONES

Where id\_organization=:p\_id\_organization and

id\_number=:p\_id\_number and

id\_number\_of\_org<>:p\_id\_number\_of\_org into :k;

if (:k=0) then

UPDATE ORGANIZATION\_PHONES

SET id\_organization=:p\_id\_organization,

id\_number=:p\_id\_number,

department\_name=:p\_department\_name

WHERE id\_number\_of\_org=:p\_id\_number\_of\_org;

else

exception HAS\_ORG\_PHONE;

suspend;

end

### Процедура обновления данных в таблице «Организации» UPD\_ORGANIZATION

CREATE PROCEDURE UPD\_ORGANIZATION (

p\_id\_organization integer,

p\_name\_of\_organization varchar(20),

p\_street varchar(30),

p\_house varchar(10),

p\_id\_rubric integer,

p\_id\_town integer,

p\_inn numeric(11,0))

as

declare variable k integer;

begin

select count(\*) from ORGANIZATION

Where inn=:p\_inn and name\_of\_organization=:p\_name\_of\_organization and id\_organization<>:p\_id\_organization into :k;

if (:k=0) then

UPDATE ORGANIZATION

SET name\_of\_organization=:p\_name\_of\_organization,

street=:p\_street,

house=:p\_house,

id\_rubric=:p\_id\_rubric,

id\_town=:p\_id\_town,

inn=:p\_inn

WHERE id\_organization=:p\_id\_organization;

else

exception HAS\_ORG;

suspend;

end

### Процедура выходного документа: Телефонные коды городов, отсортированные по алфавиту.

CREATE PROCEDURE TOWN\_ORDER\_BY\_ALFAVIT

returns (

p\_town\_name varchar(30),

p\_country varchar(20),

p\_phone\_number integer)

as

begin

For select t.town\_name, t.country, p.phone\_number

from TOWN t, PHONES p

where t.id\_number=p.id\_number

order by t.town\_name into :p\_town\_name, :p\_country, :p\_phone\_number

do

suspend;

end

### Процедура выходного документа: Телефоны физических лиц города, отсортированные по фамилии.

CREATE PROCEDURE PP\_ORDER\_BY\_ALF (

p\_id\_town integer)

returns (

p\_surname varchar(20),

p\_name varchar(20),

p\_patronymic varchar(20),

l\_phone\_assignment varchar(20),

h\_phone\_number integer,

t\_town\_name varchar(30),

t\_country varchar(20))

as

begin

for select p.surname, p.name, p.patronymic,

l.phone\_assignment,

h.phone\_number,

t.town\_name, t.country

from PERSON p, PERSON\_PHONES l, PHONES h, TOWN t

where p.id\_town=t.id\_town and p.id\_town=:p\_id\_town and p.id\_person=l.id\_person and l.id\_number=h.id\_number

order by p.name into :p\_surname, :p\_name , :p\_patronymic,

:l\_phone\_assignment,

:h\_phone\_number,

:t\_town\_name,:t\_country

do

suspend;

end

### Процедура выходного документа: Телефоны всех организаций городов, сгруппированные по городам и рубрикам.

CREATE PROCEDURE PO\_GROUP\_BY\_TANDR

returns (

pname\_of\_organization varchar(20),

pname\_rubric varchar(30),

ptown\_name varchar(30),

pphone\_number numeric(11,0))

as

begin

for select name\_of\_organization, town.town\_name , RUBRICS.name\_of\_rubric,PHONES.phone\_number

from organization, town, RUBRICS, PHONES, ORGANIZATION\_PHONES

where organization.id\_town=town.id\_town and organization.id\_rubric=RUBRICS.id\_rubric

and organization.id\_organization=ORGANIZATION\_PHONES.id\_organization

and ORGANIZATION\_PHONES.id\_number=PHONES.id\_number

group by town\_name, name\_of\_rubric, name\_of\_organization, phone\_number

into :pname\_of\_organization, :ptown\_name,:pname\_rubric,:pphone\_number

do

suspend;

end

### Процедура выходного документа: Телефоны организаций города определенной рубрики.

CREATE PROCEDURE PO\_KT\_KR (

p\_id\_town integer,

p\_id\_rubric integer)

returns (

p\_name\_of\_organization varchar(20),

p\_country varchar(20),

p\_town\_name varchar(30),

p\_street varchar(30),

p\_house varchar(10),

p\_inn integer,

p\_name\_of\_rubric varchar(30),

p\_number\_of\_phone numeric(11,0))

as

begin

for select name\_of\_organization, country, town\_name, street, house, inn, name\_of\_rubric, phone\_number

from ORGANIZATION o, organization\_phones op, RUBRICS r, PHONES h, TOWN t

where o.id\_town=t.id\_town and o.id\_rubric=r.id\_rubric and o.id\_organization=op.id\_organization and op.id\_number=h.id\_number

and o.id\_town=:p\_id\_town and o.id\_rubric=:p\_id\_rubric

order by name\_of\_organization

into

:p\_name\_of\_organization ,

:p\_country ,

:p\_town\_name ,

:p\_street ,

:p\_house ,

:p\_inn ,

:p\_name\_of\_rubric ,

:p\_number\_of\_phone

do

suspend;

end

## 4.6 Генераторы, созданные в программе IBExpert.

**Генератор** – механизм для создания уникальных значений первичных ключей при добавлении строк к таблице. Созданные генераторы приведены на рисунке 30.

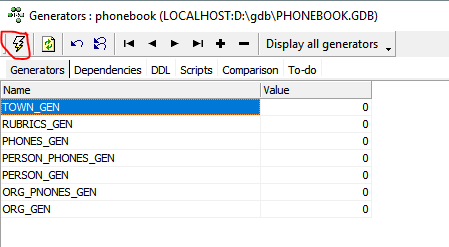


Рисунок 30 - Генераторы

## 4.7 Средства просмотра таблиц, созданные в программе IBExpert.

**Просмотр (view)** – это виртуальная таблица, которая не сохраняется в БД физически, но выглядит как реальная таблица. Просмотр может содержать выборочные данные из одной или нескольких просмотров и используется для сохранения наиболее часто используемых запросов к БД.

### Просмотр списка телефонов организаций

CREATE VIEW PHONE\_ORG(

ID\_ORGANIZATION,

ID\_RUBRIC,

ID\_TOWN,

ID\_NUMBER\_OF\_ORG,

ID\_NUMBER,

NAME\_OF\_ORGANIZATION,

STREET,

HOUSE,

INN,

DEPARTMENT\_NAME,

PHONE\_NUMBER,

COUNTRY,

TOWN\_NAME)

AS

SELECT

id\_organization,

id\_rubric,

id\_town,

id\_number\_of\_org,

id\_number,

name\_of\_organization,

street,

house,

inn,

department\_name,

phone\_number,

country,

town\_name

FROM ORGANIZATION\_PHONES, PHONES, ORGANIZATION, TOWN

WHERE ORGANIZATION\_PHONES.ID\_NUMBER=PHONES.ID\_NUMBER AND

ORGANIZATION\_PHONES.ID\_ORGANIZATION=ORGANIZATION.ID\_ORGANIZATION AND

ORGANIZATION.ID\_TOWN=TOWN.ID\_TOWN

;

### Просмотр телефонов физических лиц

CREATE VIEW PHONE\_PERSONS(

SERIAS\_AND\_NUMBER\_PASSPORT,

SURNAME,

NAME,

PATRONYMIC,

COUNTRY,

TOWN\_NAME,

STREET,

HOUSE,

PHONE\_NUMBER,

PHONE\_ASSIGNMENT)

AS

select

p.serias\_and\_number\_passport, p.surname, p.name, p.patronymic, t.country, t.town\_name, p.street, p.house,

h.phone\_number,

pp.phone\_assignment

from person p, person\_phones pp, phones h, town t

where p.id\_person=pp.id\_person and

h.id\_number=pp.id\_number and

p.id\_town=t.id\_town

;

### Просмотр телефонных кодов городов

CREATE VIEW PHONE\_TOWNS(

TOWN\_NAME,

COUNTRY,

PHONE\_NUMBER)

AS

select town\_name, country, phone\_number from TOWN, phones

where TOWN.id\_number=phones.id\_number

;

# 5 Описание базы данных на клиенте

Создав проект, создав в нем DataModule, поместим в него необходимые компоненты для работы с ранее созданной базой данных. Окно модуля примет вид как на рисунке 31.

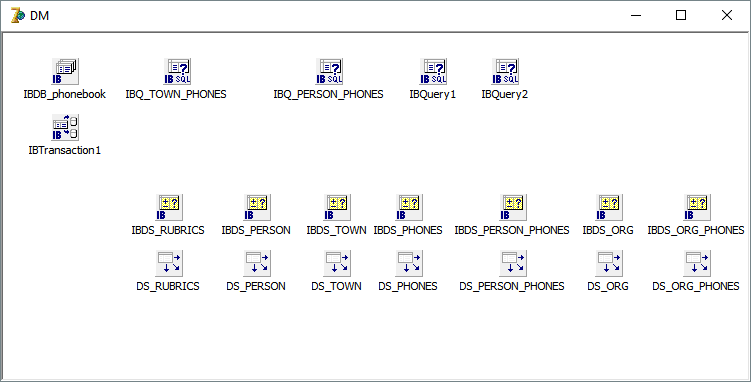


Рисунок 31– Модуль данных

Настроив компоненты модуля данных главное окно программы примет вид в соответствии с рисунком 32. Подпункты меню представлены на рисунках 33,34,35.

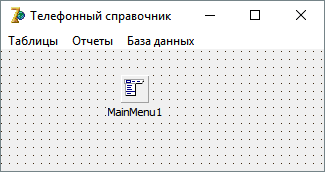


Рисунок 32 – Главное окно программы

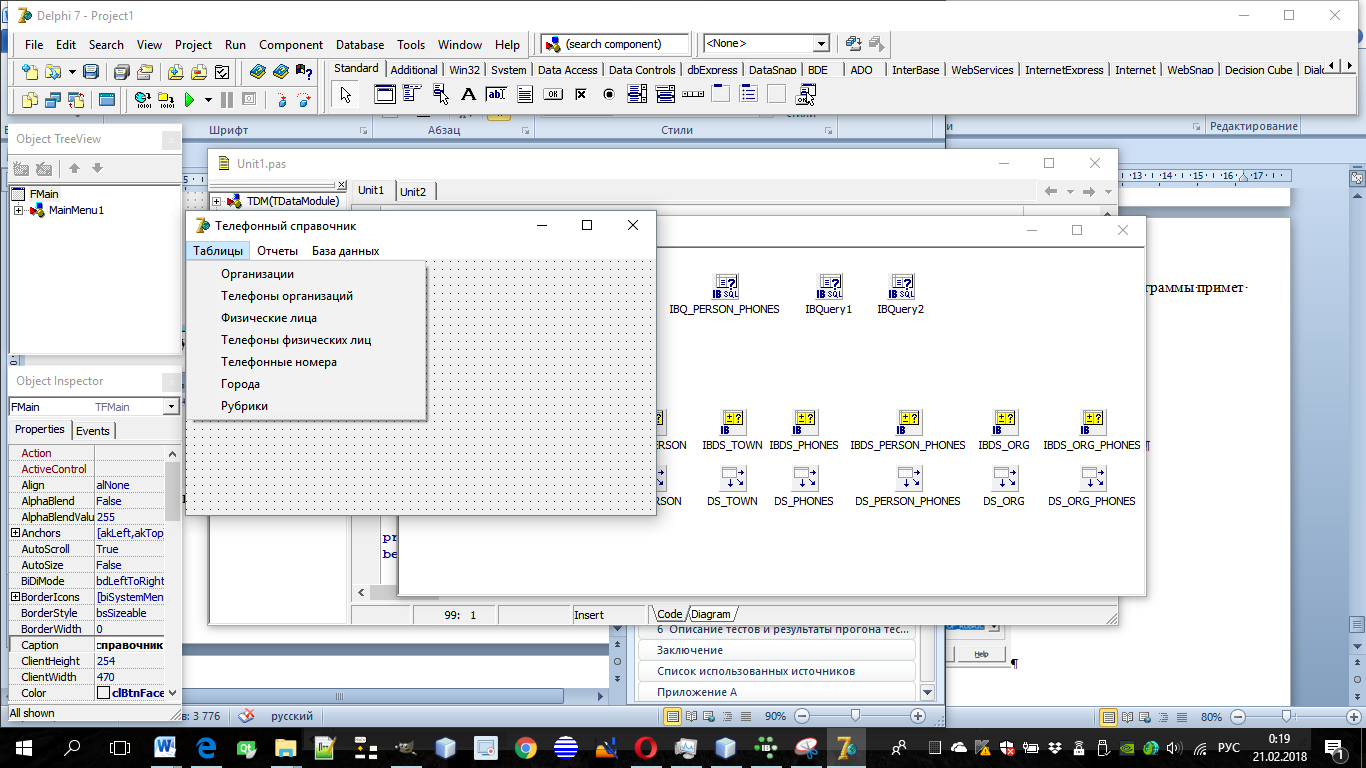


Рисунок 33 – Пункт меню «Таблицы»

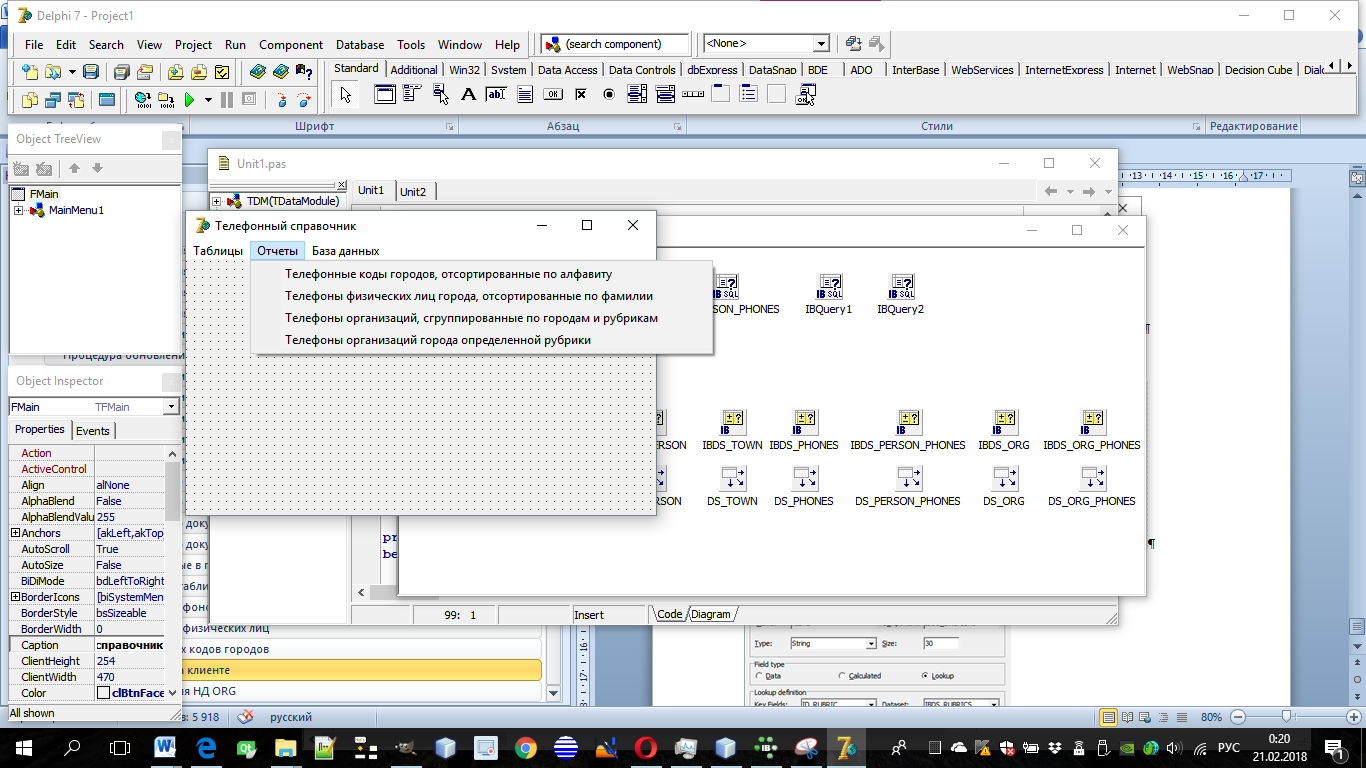


Рисунок 34 – Пункт меню «Отчеты»

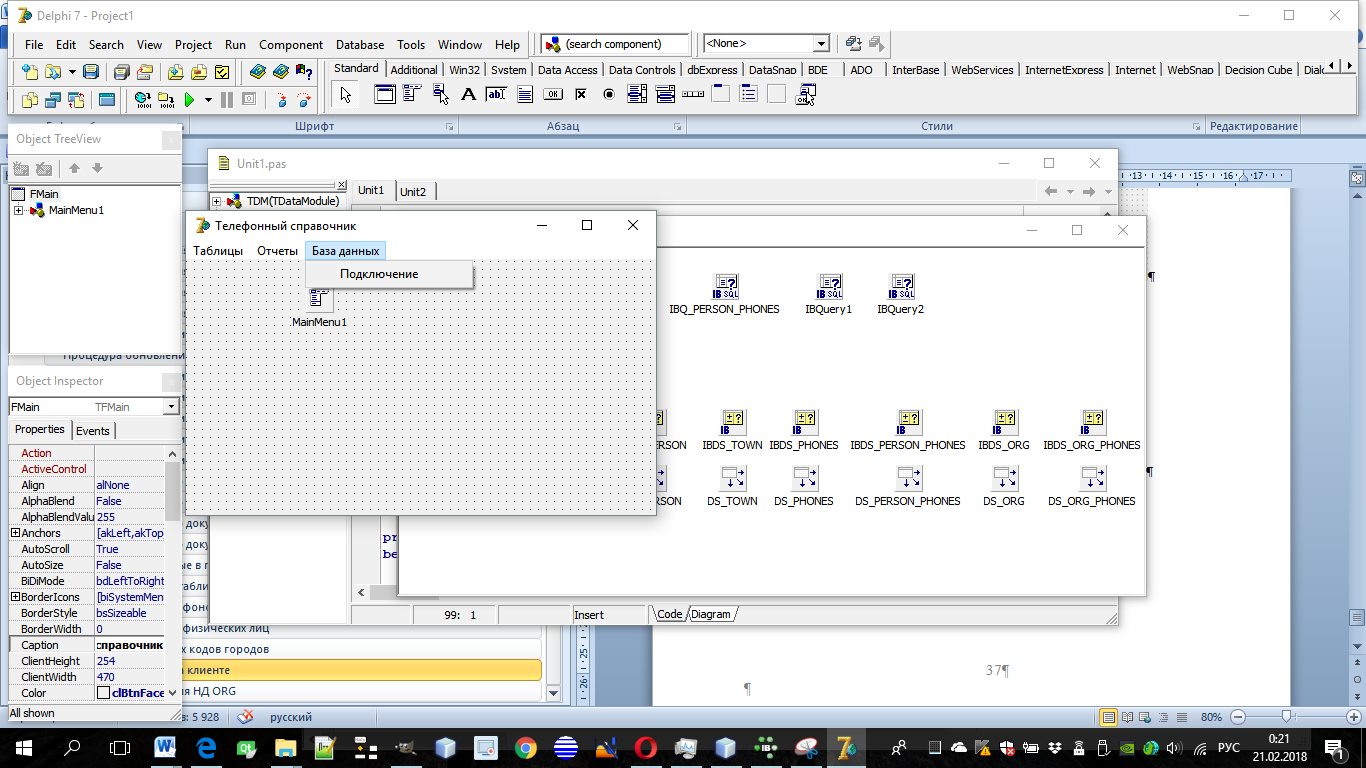


Рисунок 35 – Пункт меню «База данных»

Так же, из главного меню, щелкнув по пункту «Таблицы» откроется список из которого можно перейти в форму в которой содержится информация о городах (см. рисунок 36), о организациях (см. рисунок 37), и в форму с данными о телефонах организаций. (см. рисунок 38), в форму с данными о физических лицах (рисунок 39), в форму с данными о телефонах физических лиц (рисунок 40), в форму с данными о рубриках (рисунок 41), в форму с данными о зарегистрированные телефонных номерах (рисунок 42).

С помощью соответствующей панели в формах можно производить добавление/удаления записей.

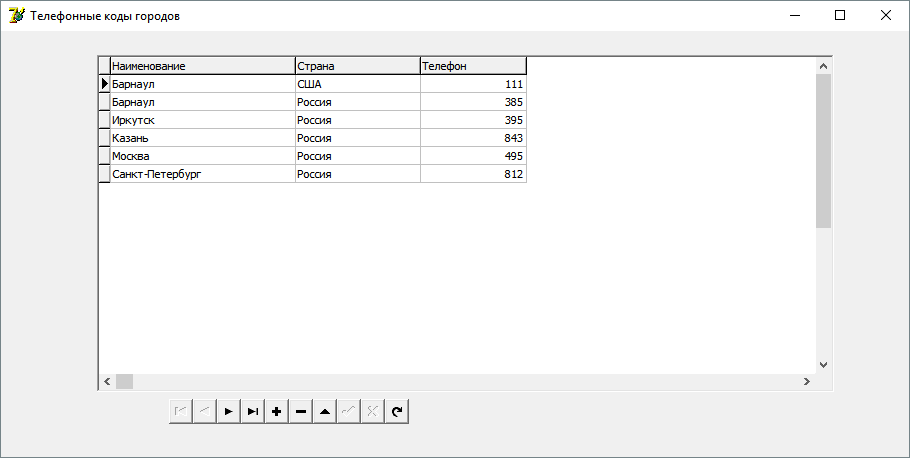


Рисунок 36 – Форма с таблицей «Города»

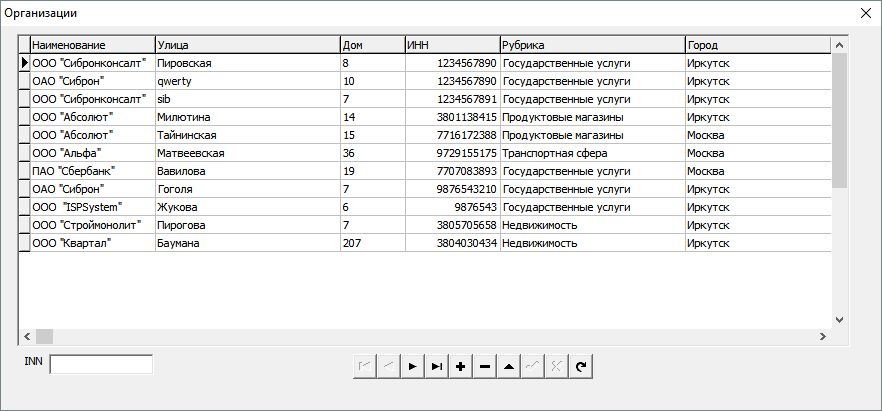


Рисунок 37 – Форма с таблицей «Организации»

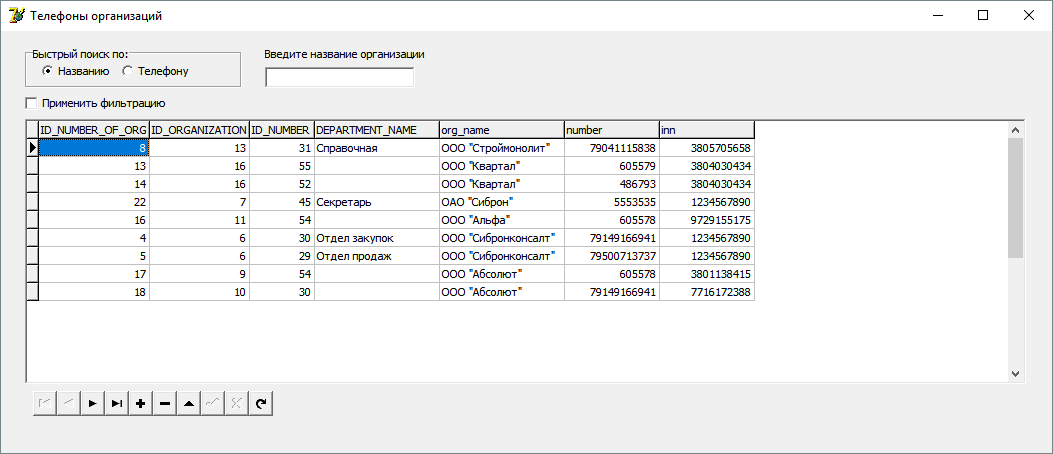


Рисунок 38 – Форма с таблицей «Телефоны организаций»

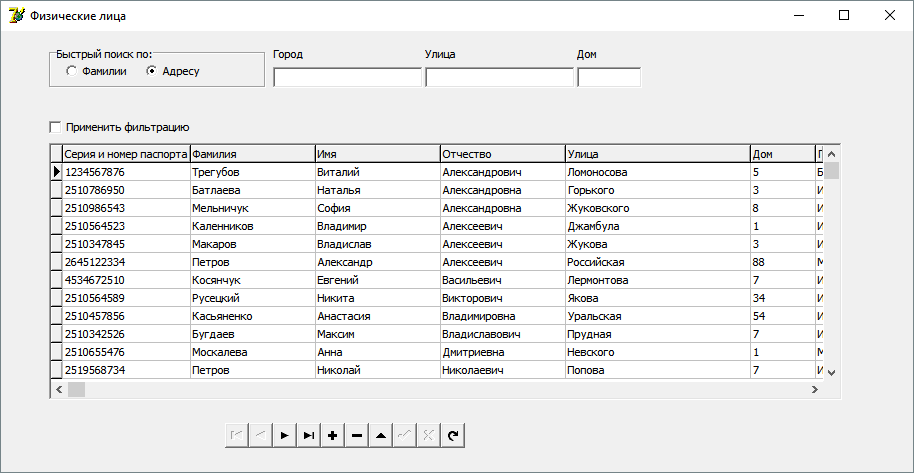


Рисунок 39 – Форма с таблицей «Физические лица»

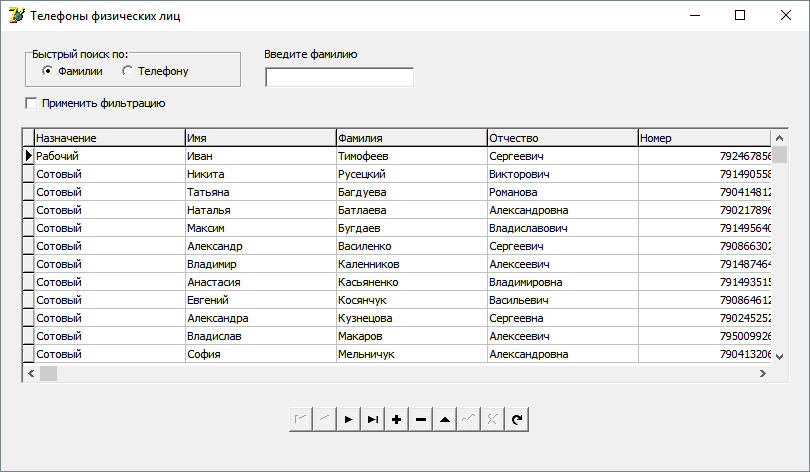


Рисунок 40 – Форма с таблицей «Телефоны физических лиц»

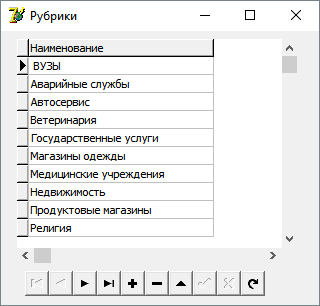


Рисунок 41 – Форма с таблицей «Рубрики»

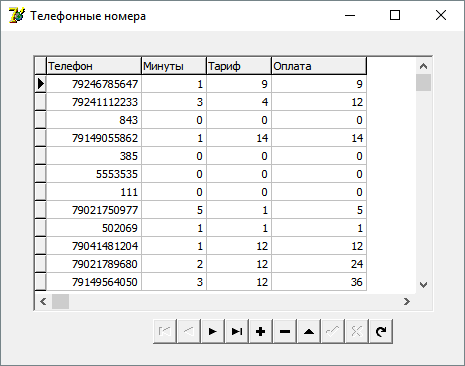


Рисунок 42 – Форма с таблицей «Телефонные номера»

Так же на форме «Организации» расположен компонент Edit1 для поиска организации по ИНН. На форме с таблицей «Телефонные номера организаций» есть компонент Edit1, для поиска организации по названию или по номеру телефона. На форме с таблицей «Физические лица» реализована возможность поиска по адресу (город, улица и дом) и по фамилии. На форме с таблицей «Телефоны физических лиц» реализована возможность поиска по фамилии и по номеру телефона.

Выбрав в главном окне из меню «Отчеты» пункт «Отсортированные коды городов оп алфавиту» откроется форма отчета, в которой будут выведены все имеющиеся коды городов, отсортированные по названию города в алфавитном порядке (рисунок 43)

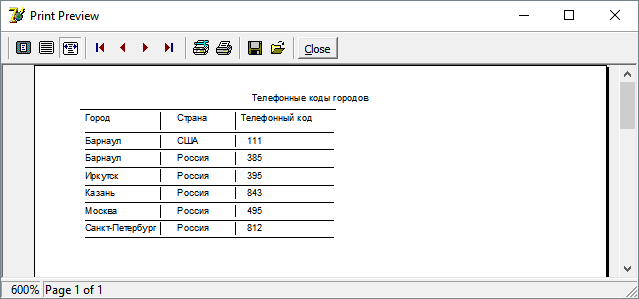


Рисунок 43 – Выходной документ: «Коды городов, отсортированные по алфавиту»

Выбрав в главном окне из меню «Отчеты» пункт «Телефоны физических лиц города, отсортированные по фамилии» на экране появится форма, в которой (см. рисунок 44), можно выбрать определенный город для которого после нажатия кнопки «ОК» откроется форма отчета в которой будет выведены все имеющиеся телефоны физических лиц из данного города (см. рисунок45).

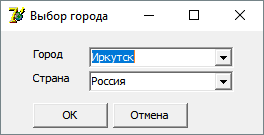


Рисунок 44 – Окно выбора города для отчета

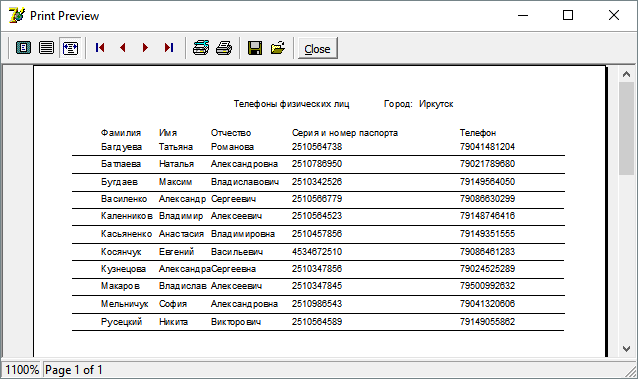


Рисунок 45 – Выходной документ: «Телефоны физических лиц города, отсортированные по алфавиту»

Выбрав в главном окне из меню «Отчеты» пункт «Телефоны организаций, сгруппированные по городам и рубрикам» на экране появится форма отчета в которой будет выведены все имеющиеся организации, сгруппированные по городам и рубрикам (см. рисунок46).

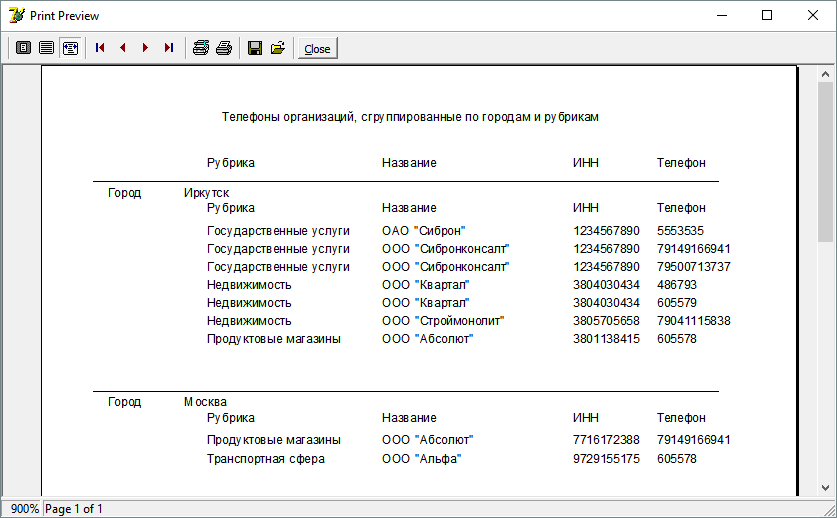


Рисунок 46 – Выходной документ: «Телефоны организаций, сгруппированные по городам и рубрикам»

Выбрав в главном окне из меню «Отчеты» пункт «Телефоны организаций города, определенной рубрики » на экране появится форма, в которой (см. рисунок 47), можно выбрать определенный город и рубрику. После нажатия кнопку «ОК» откроется форма отчета в которой будет выведены все имеющиеся телефоны организаций для выбранного города и соответствующей рубрике (см. рисунок48).

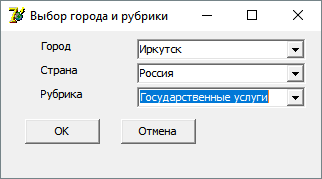


Рисунок 47 – Форма выбора города и рубрики

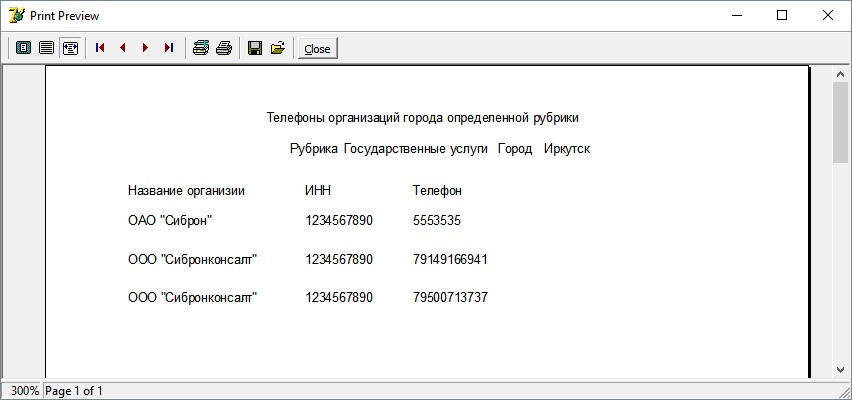


Рисунок 48 – Выходной документ: «Телефоны организаций города, определенной рубрики»

# 6 Описание тестов и результаты прогона тестов

Результаты тестов приведены в таблице 1 и на рисунках 49-55.

Таблица 1 – Результаты тестов

|  |  |  |  |
| --- | --- | --- | --- |
| № | Описание | Результат | Причина |
| 1 | Форма «Рубрики». В поле «Название рубрики» введено «ВУЗЫ» | Выдача сообщения об ошибке «Рубрика с таким название уже существует» | Такая рубрика уже есть в базе |
| 2 | Форма «Города». В поле «Название» введено «Барнаул», а в поле «Страна» введено «Россия» | Выдача сообщения об ошибке «Такой город уже существует» | Город с таким названием и принадлежащий этой же стране уже существует |
| 3 | Форма «Телефонные номера» в поле «Телефон» введено «111» | Выдача сообщения об ошибке «Такой телефонный номер уже существует» | Телефон с таким же номером уже зарегистрирован |
| 4 | Форма «Телефонные номера физических лиц» : для Тимофеева Ивана Сергеевича задан номер 79246785647 | Выдача сообщения об ошибке «Такой телефонный номер для физического лица уже существует» | Телефон с таким же номером уже зарегистрирован для данного пользователя |
| 5 | Форма «Физические лица» создается пользователь «Ануфриева Анна Дмитриевна с ИНН 1234567890» | Выдача сообщения об ошибке «Физическое лицо с таким ИНН уже существует» | Физическое лицо с таким же ИНН уже есть в базе |
| 6 | Форма «Телефонные номера организаций» для организации ООО «Строймонолит» задается телефон 790411115838 | Выдача сообщения об ошибке «Телефонный номер для данной организации уже существует» | Запись с таким же названием организации и с таким же номером телефона уже имеется. |
| 7 | Форма «Организации»: пытаемя зарегистрировать организацию ООО «Сибронконсалт» с ИНН 1234567890 | Выдача сообщения об ошибке «Организация с таким ИНН уже существует» | В БД уже имеется запись об этой организации |

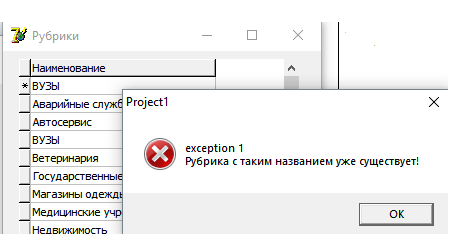


Рисунок 49 – Результаты теста №1

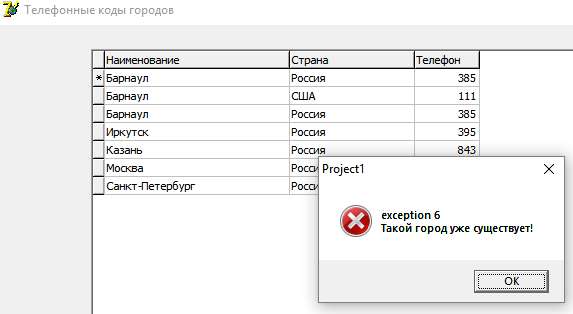


Рисунок 50 – Результаты теста №2

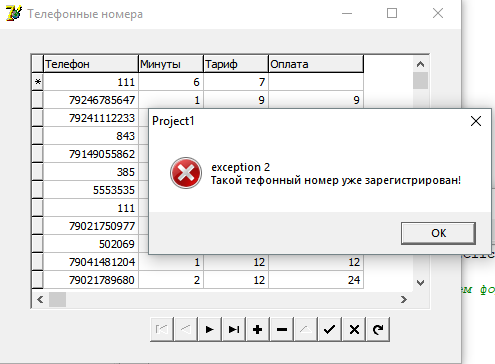


Рисунок 51 – Результаты теста №3

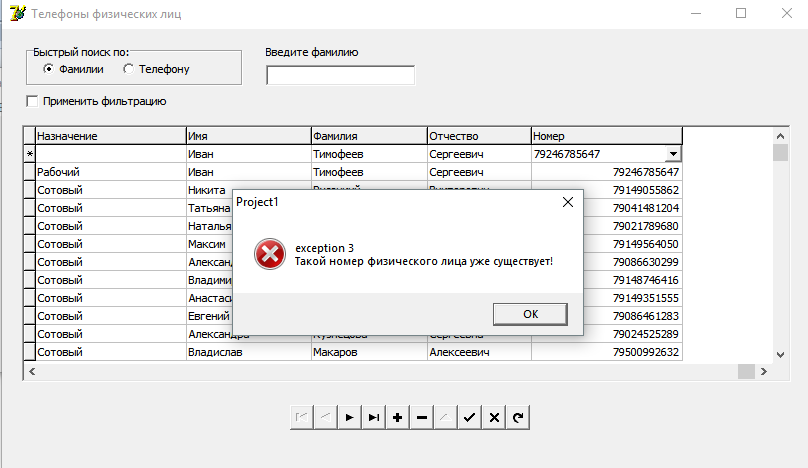


Рисунок 52 – Результаты теста №4

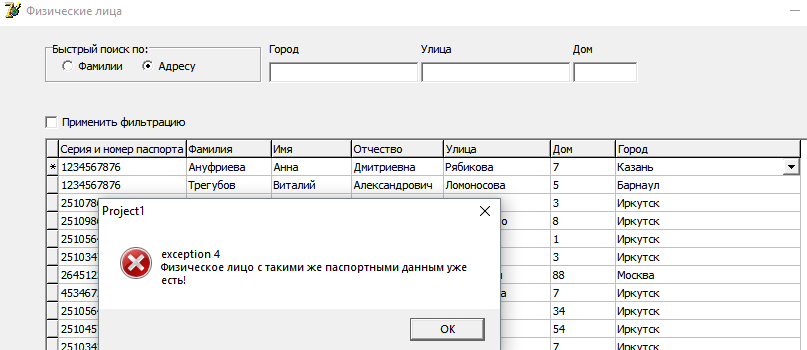


Рисунок 53 – Результаты теста №5

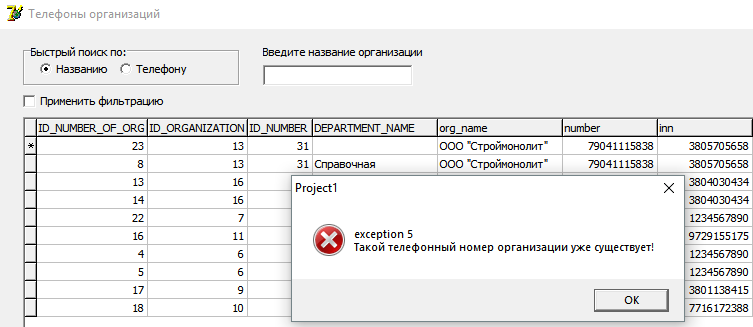


Рисунок 54 – Результаты теста №6

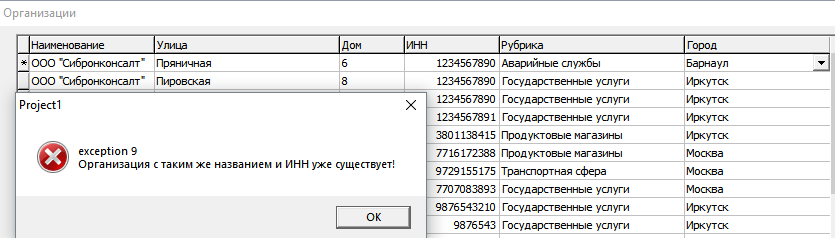


Рисунок 55 – Результаты теста №7

# Заключение

В ходе выполнения курсовой работы была спроектирована база данных «Телефонный справочник» и разработано клиент-серверное приложение, содержащие необходимые данные. База данных спроектирована и разработана с помощью основных программ, таких как StarUML, CA ERwin Data Modeler, IBExpert, Borland Delphi.

В данной базе данных были созданы таблицы, выделенные при формализации предметной области. А также произведено их наполнение. Были спроектированы и реализованы SQL–запросы, указанные в задании на курсовую работу. Разработан программный продукт, обеспечивающий добавление и редактирование данных, а также поиск в базе.

# Список использованных источников

1. Агальцов В.П. Базы данных : учеб. для студентов вузов спец. 230100 «Информатика и вычислительная техника» / В. П. Агальцов. – М.: Форум, Б.г.2011-Кн.2 Распределенные и удаленные базы данных . – Б.м.: Б.и., 2011. – 270 с.
2. Базы данных : учеб. пособие / А. С. Дорофеев; Иркут. гос. техн. ун-т . – Иркутск: Изд-во ИрГТУ, 2008. – 99 с. : a-ил.
3. Дейт К. Дж. Введение в системы баз данных = Introduction to Database Systems. — 8-е изд. — М.: Вильямс, 2005. — 1328 с
4. Кузин А.В. Базы данных : учеб. пособие по направлению подгот. дипломир. специалистов "Информатика и вычисл. техника" / А. В. Кузин, С. В. Левонисова. - 4-е изд., стер.. - М. : Академия, 2010. - 311 с.
5. Сосинская С.С. Управление данными : учеб. пособие / С. С. Сосинская. - Иркутск: Изд-во ИрГТУ, 2006. - 139 с.
6. Хомоненко А.Д. Базы данных: Учебник для ВУЗов / Хомоненко А.Д. Изд-во: "Корона-Век", 2010.
7. Delphi и базы данных [Электронный ресурс] // Работа с базами данных с использованием Delphi [URL: http://www.cyberforum.ru/delphi-database/](URL:http://www.opengl.org.ru/) (Дата обращения 10.08.17)
8. Гарант – информационно правовой портал [Электронный ресурс]// Аспекты регистрации юридических лиц URL: <http://www.garant.ru/ia/opinion/author/nikanorova/549829/> (дата обращения 22.10.17)