

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

Направление подготовки: 09.04.01 «Информатика и вычислительная техника»

Магистерская программа: «Вычислительные машины, комплексы, системы и сети»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА МАГИСТРА

на тему

Ведение и актуализация классификаторов по

пирамидальному принципу в сложных

информационных системах

Студент Гачи 03.06.2020 г. Полковникова Светлана Андреевна
(подпись, дата) (ФИО полностью)

Руководитель Р.Н. Федюнин 03.06.2020 г.
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр А.В. Кучин
(подпись, дата) (фамилия, инициалы)

Рецензент А.В. Васильков
Зам. н.з. НТБ АО «НПТ «Рубин» (должность, место работы) (подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 03.06.2020 № 12)

Заведующий кафедрой М.А. Митрохин
(подпись) (фамилия, инициалы)

Работа защищена с отметкой отлично (протокол заседания ГЭК от
15.06.2020 № 3)

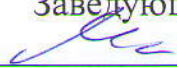
Секретарь ГЭК Е.И. Гурин
(подпись) (фамилия, инициалы)

Пенза 2020

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
 М.А. Митрохин
«30» 09 2019 г.

**ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
МАГИСТРА**

1. Студент Полковникова Светлана Андреевна гр. 18ВВм1 факультета вычислительной техники
2. Тема работы «Ведение и актуализация классификаторов по пирамидальному принципу в сложных информационных системах».
Тема утверждена приказом ПГУ № 1039/о от " 30 " сентября 2019 г.
3. Руководитель магистерской диссертации к.т.н., доцент каф. ВТ ПГУ Федюнин Р.Н.
4. Задание на работу (назначение разработки, исходные данные и т.п.)
Назначение разработки: ведение и актуализация классификаторов по пирамидальному принципу в сложных информационных системах.
Разработать программные модули формирования пакетов актуализации и загрузки сформированных пакетов данных в базу данных.
5. Перечень подлежащих разработке вопросов
 1. Анализ автоматизированных систем ведения классификаторов.
 2. Рассмотрение основных способов поддержания данных в актуальном состоянии.
 3. Создание модулей выгрузки и загрузки пакета актуализации классификаторов и нормативно-справочной информации.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Анализ предметной области	17	22.03.2020	
2.	Рассмотрение основных способов поддержания данных в актуальном состоянии	15	06.04.2020	
3.	Анализ требований на разработку	15	19.04.2020	
4.	Разработка модулей выгрузки и загрузки пакета актуализации	15	07.05.2020	
5.	Тестирование	15	07.05.2020	
6.	Отладка приложения на ЭВМ	11	08.05.2020	
7.	Оформление выпускной работы	12	10.06.2020	

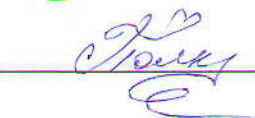
Дата выдачи задания "26" сентября 2019 г.

Руководитель магистерской диссертации
(подпись, дата)



Федюнин Р.Н.

Задание к исполнению принял студент
(подпись, дата)



Полковникова С.А.

Работу к защите допустить
Декан факультета
(подпись, дата)



Фионова Л.Р.

РЕФЕРАТ

Выпускная квалификационная работа, 98 листов, 26 рисунков, 2 таблицы, 5 источников, 4 приложения.

ПРОГРАММНЫЙ МОДУЛЬ, ПАКЕТЫ АКТУАЛИЗАЦИИ, ЗАГРУЗКА, ВЫГРУЗКА, КЛАССИФИКАТОРЫ, НОРМАТИВНО-СПРАВОЧНАЯ ИНФОРМАЦИЯ.

Цель работы – изучение навыков ведения классификаторов и создание модулей выгрузки и загрузки пакета актуализации классификаторов и нормативно-справочной информации.

Проведение исследований по предложенной мной теме считаю действительно актуальным и новым, так как при создании данной системы можно сократить количество рассматриваемых одновременно объектов, упорядочить процесс рассмотрения объектов, а также выделить общие характеристики.

Объектом разработки являются программный модуль для формирования пакета актуализации классификаторов и модуль загрузки пакета актуализации в БД.

В процессе разработки была выявлена важность поддержания классификаторов в актуальном состоянии, обеспечения полноты информации и ее распространения среди пользователей.

В результате проектирования получена подсистема актуализации, включающая в себя два программных модуля (ПМ) – модуль формирования пакета актуализации и модуль загрузки пакета актуализации в базу данных (БД). Программный проект содержит 2 программных модуля, разработанные на языке Си++ с использованием библиотеки Qt.

Результаты данной работы состоят во внедрении разработанных модулей загрузки и выгрузки пакетов актуализации данных в систему ведения классификаторов и НСИ на предприятии АО «НПП «Рубин».

Содержание

ВВЕДЕНИЕ.....	4
1 Анализ предметной области и требований.....	6
1.1 Анализ предметной области.....	6
1.1.1 Анализ программ-аналогов.....	7
1.1.1.1 АСУ НСИ 2.0.....	8
1.1.1.2 Программа 1С.....	8
1.1.1.3 Недостатки и методы устранения.....	9
1.1.2 Выбор инструментальных средств.....	10
1.1.2.1 Обзор используемых языков программирования.....	10
1.1.2.2 Обзор используемой среды разработки.....	11
1.2 Анализ требований.....	11
2 Проектирование программного обеспечения.....	12
2.1 Пользовательский интерфейс.....	12
2.2.1 Сообщение оператору.....	15
2.2 Описание базы данных.....	17
3 Разработка программного обеспечения.....	26
3.1 Описание основных классов.....	27
3.2 Схема алгоритма.....	32
4 Тестирование программного обеспечения.....	34
Заключение.....	39
Список использованных источников.....	40
ПРИЛОЖЕНИЕ А. ФОРМИРОВАНИЕ ПАКЕТОВ АКТУАЛИЗАЦИИ.....	41
ПРИЛОЖЕНИЕ Б. ЗАГРУЗКА СФОРМИРОВАННЫХ ПАКЕТОВ АКТУАЛИЗАЦИИ.....	48
ПРИЛОЖЕНИЕ В. СОЗДАНИЕ ФУНКЦИЙ БАЗЫ ДАННЫХ.....	68
ПРИЛОЖЕНИЕ Г. ПРЕЗЕНТАЦИЯ.....	92

ВВЕДЕНИЕ

Ввиду огромного роста количества информации повсеместно, а особенно в интернете, все большую роль играет возможность классифицировать её.

Информацию можно классифицировать по значению:

- актуальная информация — информация, ценная в данный момент времени;
- достоверная информация — информация, полученная без искажений;
- понятная информация — информация, выраженная на языке, понятном тому, кому она предназначена;
- полная информация — информация, достаточная для принятия правильного решения или понимания;
- полезная информация — полезность информации определяется субъектом, получившим информацию в зависимости от объёма возможностей её использования.

Существует множество способов решения проблемы переработки большого объема информации, один из них — это создание классификаторов. В классификаторах информация разделяется на подмножества по сходству или различию в соответствии с принятыми методами.

Процесс поддержания классификаторов в актуальном состоянии, обеспечения полноты информации и ее распространения среди пользователей называется ведением классификаторов. При этом средства и соответствующая организационная структура составляют систему ведения классификаторов [1].

В процессе ведения классификаторов совершенствуются их состав и содержание, но не нарушаются принципы их построения и целостная структура.

Ведение классификатора заключается в подготовке и внесении в него изменений в части включения, аннулирования или изменения кодов и наименований классификационных группировок, или иной информации, являющейся его частью.

В данной выпускной квалификационной работе требуется разработать программный модуль для формирования пакета актуализации классификаторов и модуль загрузки пакета актуализации в БД.

Таким образом, можно сделать вывод, что тема выпускной квалификационной работы является актуальной.

1 Анализ предметной области и требований

Анализ предметной области

Классификатор – это сгруппированный, по различным признакам, перечень наименованных объектов, где каждому из них присвоен уникальный код, в соответствии с их общими признаками или различиями. Классификаторы применяются для упрощения работы по сбору информации и систематизации данных, а также используются для указания кодов в различных унифицированных документах [2].

Классификаторы разрабатываются как на уровне отдельных предприятий (организаций), так и на уровне государств. Существуют следующие уровни классификаторов (рисунок 1):

- международные — стандартные классификаторы, используемые по всему миру;
- межгосударственные — классификаторы, используемые в рамках экономических союзов и других межгосударственных объединений: например, классификаторы, используемые в ЕС, СНГ и т. д.
- национальные, или межотраслевые — классификаторы, используемые в пределах государства. Не должны противоречить международным классификаторам;
- отраслевые — классификаторы, используемые в рамках одной отрасли;
- системные — классификаторы, принятые отдельным предприятием (организацией) для применения в рамках своей автоматизированной системы. Они содержат информацию, необходимую для решения задач в конкретной АС и отсутствующую в национальном или отраслевом классификаторе.



Рисунок 1 – Уровни классификаторов

Классификаторы подразделяют на виды в зависимости от объекта систематизации. В РФ применяются широкий круг общероссийских классификаторов.

В классификаторах применяется два метода классификации: иерархический и фасетный. Выбор между этими двумя методами зависит от особенностей конкретной предметной области.

Классификация позволяет выделить из всего многообразия объектов группы с определенными свойствами и позволяет установить наследственные связи между объектами. Общими словами, классификация нужна, чтобы упорядочить, разложить "по полочкам" информацию, однотипные данные.

Разрабатываемая программа позволяет автоматизировано вести классификаторы, словари и нормативно-справочную информацию (НСИ) в отдельной (своей) базе данных (БД). А разрабатываемый мной программный модуль позволяет формировать пакеты актуализации с измененной информацией в классификаторах и НСИ, а также загружать актуализированную информацию в БД.

1.1.1 Анализ программ-аналогов

В качестве аналогов разрабатываемой программы рассмотрим «АСУ НСИ 2.0» и программу 1С.

1.1.1.1 АСУ НСИ 2.0

АСУ НСИ 2.0 – это автоматизированная система управления НСИ, предназначена для поддержки и ведения уже существующего справочника материально-технических ресурсов (МТР). АСУ НСИ 2.0 состоит из справочника МТР, дополнительных справочников (ОКВЭД2, ОКПД2, справочника производителей и т.д.), реестра запросов.

Функции АСУ НСИ 2.0:

- Отображение данных справочника МТР;
- Управление данными справочника МТР;
- Интеграция справочника МТР с другими системами;
- Отображение аналитики по справочнику МТР и работе в АСУ НСИ 2.0.

Основным недостатком программы АСУ НСИ 2.0 является отсутствие ведения классификаторов (только ведение НСИ).

1.1.1.2 Программа 1С

«Платформа 1С» – это единая технологическая среда, с помощью которой разработчики создают программы (прикладные решения).

В 1С классификатором называют систематизированный перечень установленных элементов для всех компаний. В типовых конфигурациях 1С могут присутствовать различные классификаторы, необходимые для учета тех или иных показателей

Есть только 2 способа загрузки классификатора в программу 1С – с диска 1С ИТС и с сайта (например, сайт банка, если это классификатор банков).

Основным недостатком программы 1С является отдельная разработка конфигурации для ведения классификаторов и НСИ, а также отсутствие кроссирования данных.

1.1.1.3 Недостатки и методы устранения

На таблице 1 отображен результат сравнения информации данных программных комплексов и разрабатываемой системы ведения и актуализации классификаторов.

Таблица 1 – Анализ информации об аналогах разрабатываемой системы

	АСУ НСИ 2.0	1С	КНСИ
1. Поддерживаемые ОС	Windows	Windows Linux	Windows Linux
2. Стоимость	Стоимость аренды мощностей у поставщика облачных вычислений + стоимость развертывания системы + стоимость администрирования и предоставления доступа к системе	Лицензия оплачивается каждый год + стоимость обслуживания	Лицензия оплачивается 1 раз + стоимость обслуживания
3. Ведение классификаторов, НСИ, кроссов	Только ведение НСИ	Ведение классификаторов и НСИ (но требует разработки конфигурации)	Ведение классификаторов, НСИ, кроссов
4. Форматы выгрузки	doc, docx	xl	zip, xml, xmlx, json, jsonx.
5. Форма выгрузки	Выгружает только справочники	Выгружает отдельные классификаторы	Возможна выгрузка как отдельных классификаторов, так и всех сразу + одновременная выгрузка НСИ и кроссов

Разрабатываемая система ведения классификаторов – это универсальная система, предназначенная для ведения классификаторов, НСИ и кроссов, которая

позволяет выгружать данные в форматах zip, xml, xmlx, json, jsonx. С помощью разрабатываемых мной программных модулей производится формирование пакетов актуализации данных в системе, а также загрузка сформированных пакетов данных.

1.1.2 Выбор инструментальных средств

1.1.2.1 Обзор используемых языков программирования

В качестве языка программирования для реализации данного дипломного проекта был выбран язык C++, т.к. приложения, написанные на нём, обладают высокими скоростными характеристиками [3]. Язык C++ явился мощным и стремительным рывком в развитии программирования. C++ и по сей день занимает господствующее положение среди языков программирования в мире. Огромное множество профессиональных программистов использует именно его при разработке разного рода проектов. Также важную роль в выборе языка программирования сыграло наличие качественных свободных библиотек и реализации пользовательского интерфейса(Qt).

Библиотека Qt представляет собой набор классов C++ и инструментов разработки программ для Windows, Linux, Mac OS X и встраиваемых систем (Embedded Linux). На всех платформах библиотека Qt использует свой собственный набор визуальных элементов. В результате приложения, созданные на её основе, во всех системах выглядят и работают одинаково.

Библиотека Qt является лидером среди имеющихся средств разработки межплатформенных программ на языке C++.

Разработка будет осуществляться в операционной среде Linux на языке C++ с использованием библиотеки Qt.

1.1.2.2 Обзор используемой среды разработки

В качестве среды разработки был выбран Qt Creator.

Qt Creator – кроссплатформенная интегрированная среда для разработки на C, C++.

При работе в Qt Creator используется несколько режимов, которым соответствуют вкладки на панели, также существует быстрое переключение между режимами. Для чтения документации в интегрированной среде разработки (IDE) предусмотрен отдельный режим справки [4].

Qt Creator использует файлы в формате .pro, поэтому в IDE легко импортируются старые проекты, созданные вручную. Также доступен мастер, при помощи которого можно создавать заготовку нового проекта.

Анализ требований

В данной части работы будут описаны функциональные, программные и технические требования к программным модулям.

Программные модули должны удовлетворять следующим требованиям:

1. Формирование пакетов актуализации классификаторов;
2. Формирование пакетов актуализации НСИ;
3. Загрузка сформированных пакетов актуализации классификаторов;
4. Загрузка сформированных пакетов актуализации НСИ;
5. Поддержка операционных систем Windows и Linux.

Результатом работы является программный модуль, который позволяет сформировать пакеты актуализации классификаторов и НСИ, и программный модуль, который позволяет загружать сформированные пакеты актуализации.

2 Проектирование программного обеспечения

2.1 Пользовательский интерфейс

Элементы пользовательского интерфейса представлены на рисунках 2 – 7.

На рисунке 2 отображено главное окно программы.

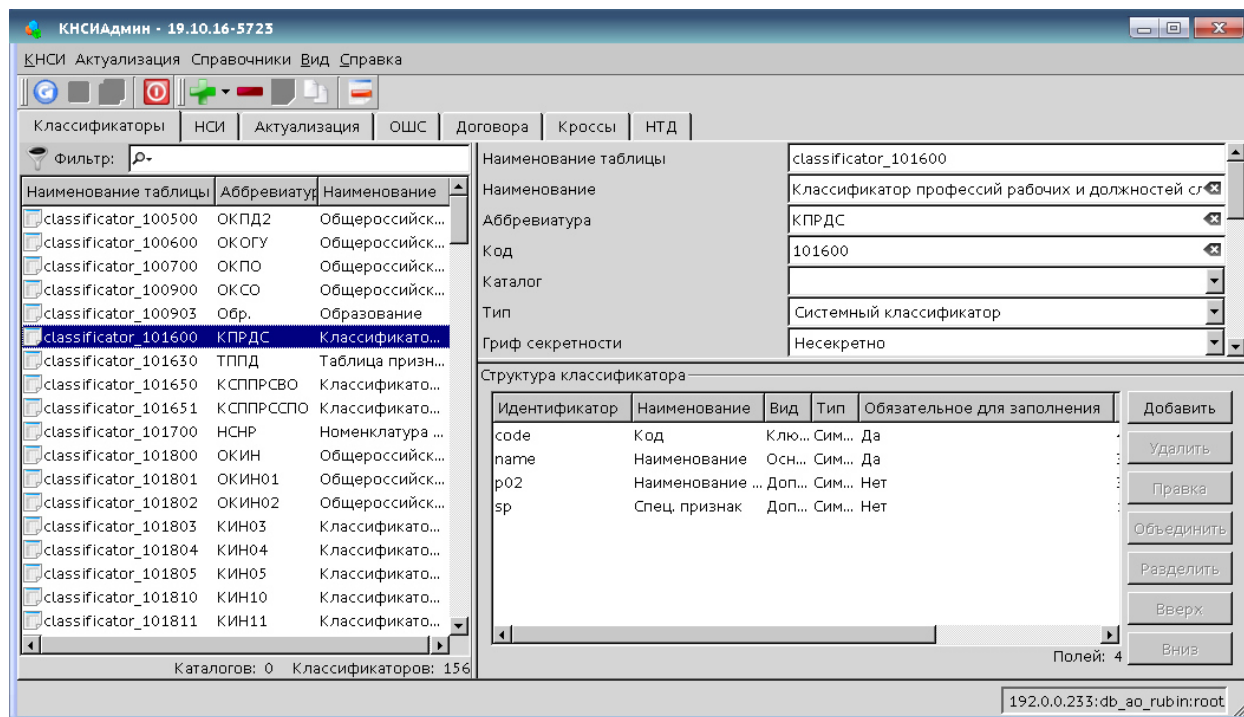


Рисунок 2 – Главное окно программы

Для выгрузки пакета актуализации классификаторов необходимо выбрать последовательно пункты меню программы КНСИ «Актуализация/Выгрузка», в результате чего появится окно, представленное на рисунке 3.

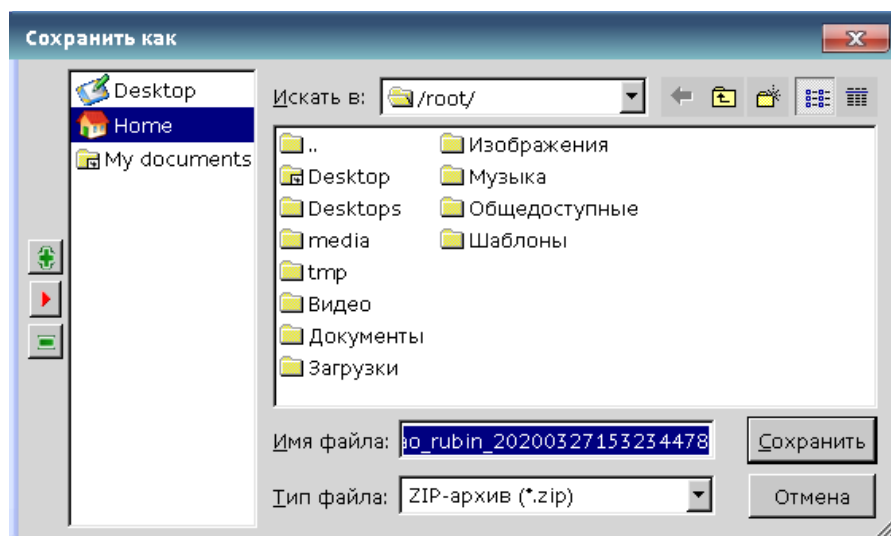


Рисунок 3 – Путь сохранения пакета актуализации

Выберите путь сохранения пакета актуализации и нажмите на кнопку «Сохранить». В результате появится окно с настройками содержимого пакета (рисунок 4), выберите, что необходимо выгрузить и нажмите на кнопку «Принять».

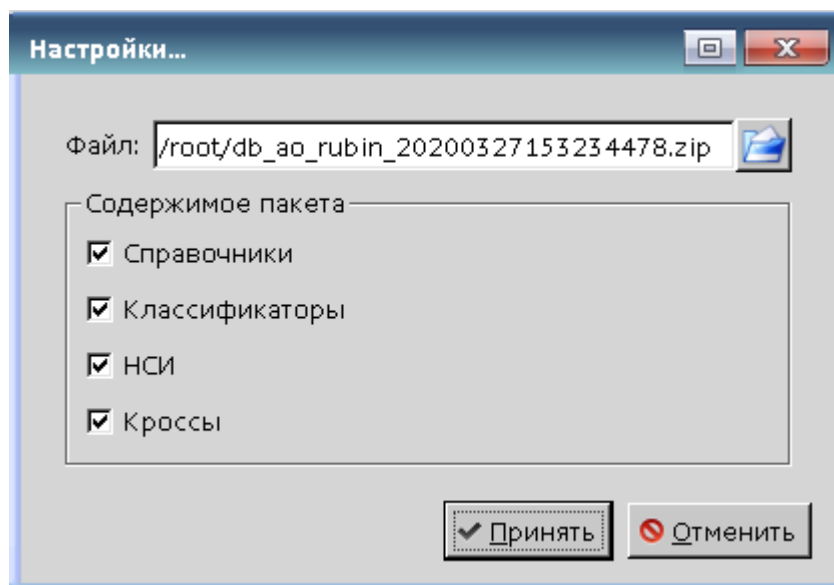


Рисунок 4 – Настройка выгрузки пакета актуализации

Откроется окно выгрузки файла актуализации, представленное на рисунке 5.

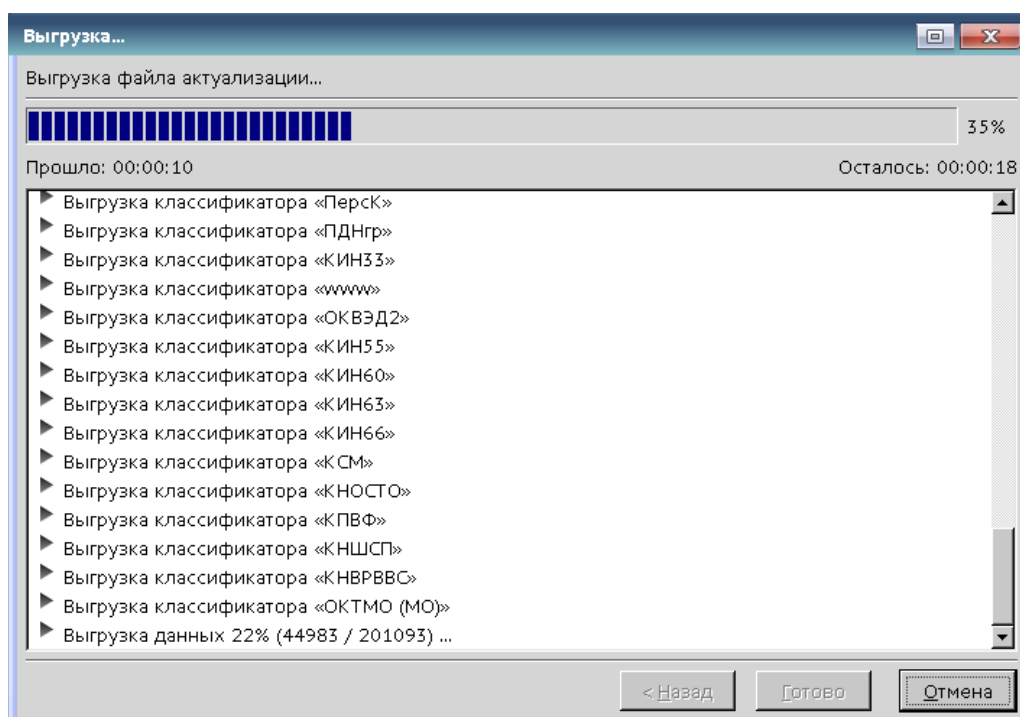


Рисунок 5 – Окно выгрузки файла актуализации

Кнопка «Отмена» служит для отмены процесса выгрузки файла.

Кнопка «Назад» позволяет вернуться к настройке содержимого пакета выгрузки (см. рисунок 4).

После успешного завершения выгрузки станет доступна кнопка «Готово» и файл будет сохранен в указанную папку (см. рисунок 3).

Программу для загрузки файла актуализации назвала knsi_ilo_sync.

Для загрузки актуализации классификаторов в базу данных выберите программу knsi_ilo_sync и нажмите на клавишу «Enter». В результате появится окно, приведенное на рисунке 6.

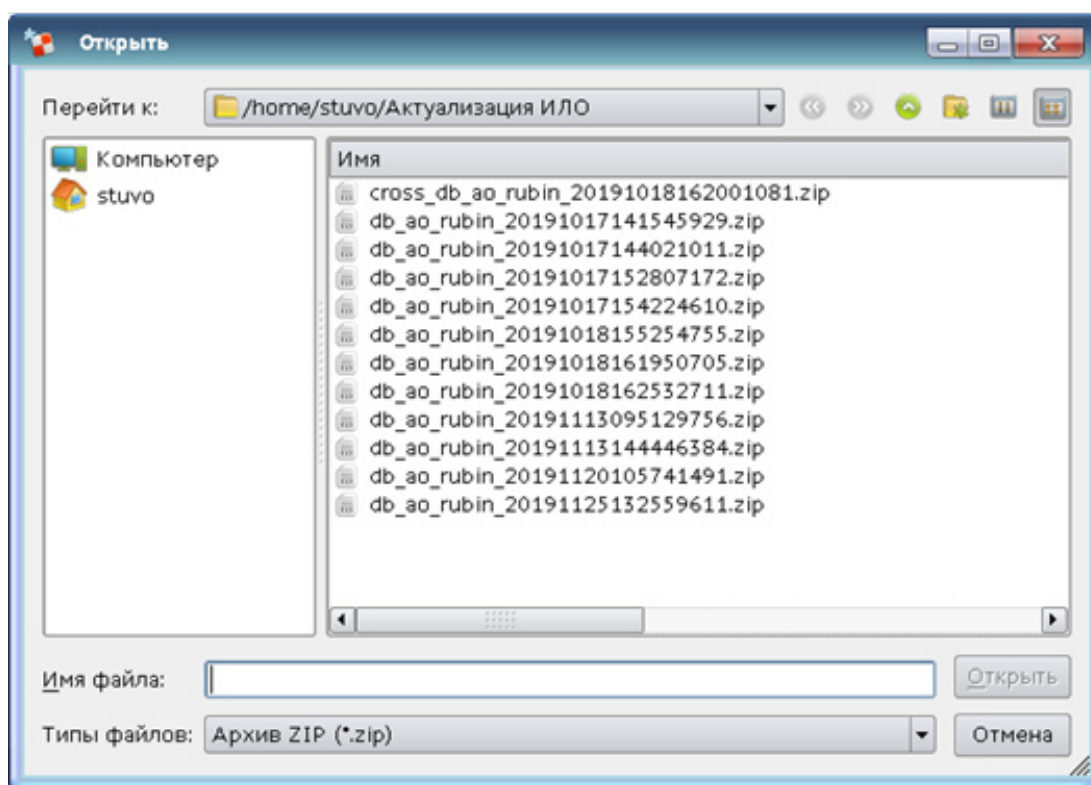


Рисунок 6 – Выбор файла актуализации для загрузки

Выберите необходимый файл актуализации классификаторов (ilo.zip) и нажмите кнопку . Появится окно загрузки файла (рисунок 7).

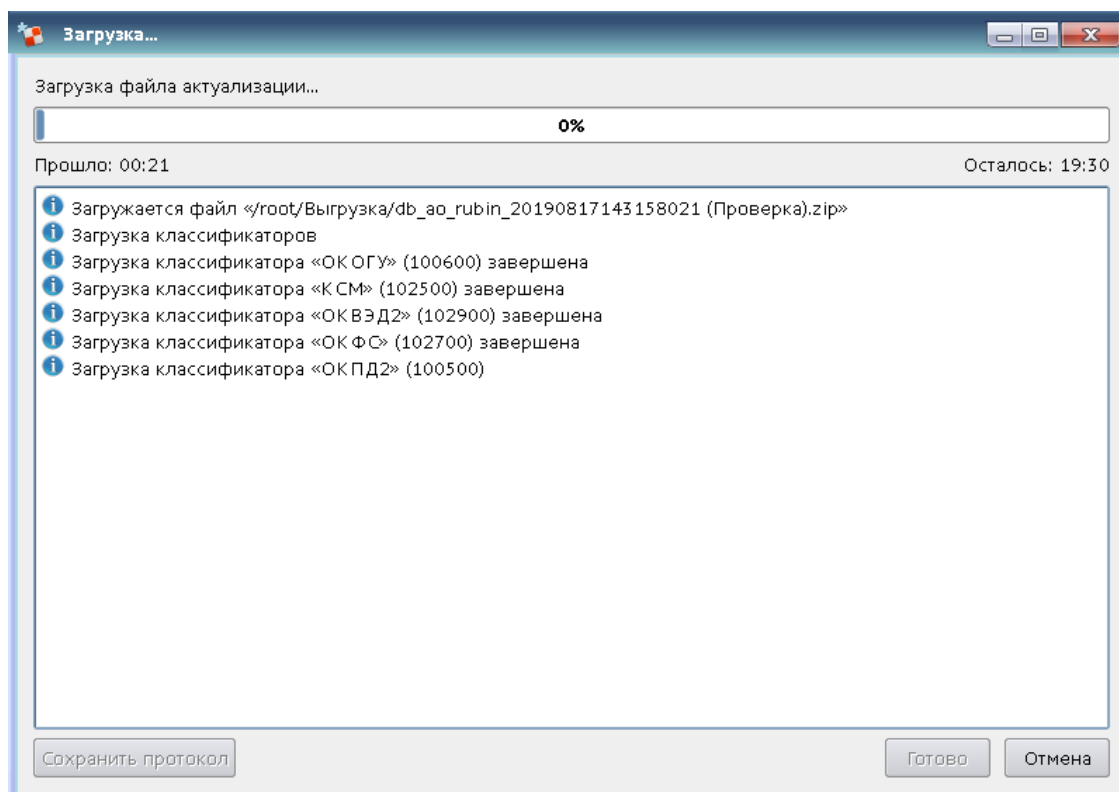


Рисунок 7 – Процесс загрузки файла актуализации

Дождитесь завершения процесса загрузки файла актуализации и нажмите кнопку «Готово».

Кнопка «Сохранить протокол» служит для сохранения журнала загрузки в файл.

2.2.1 Сообщение оператору

В ходе выполнения программного модуля пользователю могут выдаваться ошибочные сообщения.

Ошибки, выдаваемые на экран монитора в процессе решения задач, приведены в таблице 2.

Таблица 2 – Перечень сообщений

<i>Ошибка</i>	<i>Причина</i>	<i>Действие</i>
Ошибка открытия файла	Не найден указанный файл в заданном каталоге	Указать правильный путь и имя файла; установить полномочия на файл текущему пользователю

Продолжение таблицы 2

Ошибка структуры файла	Структура заданного файла не соответствует структуре принятых в программе данных	Проверить корректность структуры файла
Ошибка чтения файла	Возникла ошибка при чтении заданного файла	Проверить работоспособность накопителя данных
Ошибка подключения к БД	Невозможно установить подключение к БД.	Проверить состояние сетевого подключения, запущен ли сервер БД на удалённом компьютере.
Файл не является zip-архивом	Возникла ошибка при загрузке пакета актуализации	Выбрать корректный файл
Синтаксическая ошибка в xml-файле	Возникла ошибка при загрузке данных	Выбрать корректный файл

При выдаче сообщений об ошибках, необходимо следовать указаниям в описании сообщений, либо обратиться к администратору системы.

2.2 Описание базы данных

База данных разработана на языке PostgreSQL [5]. Она включает в себя набор таблиц и набор функций. На рисунке 8 представлена логическая схема базы данных, на основе которой создается физическая схема.

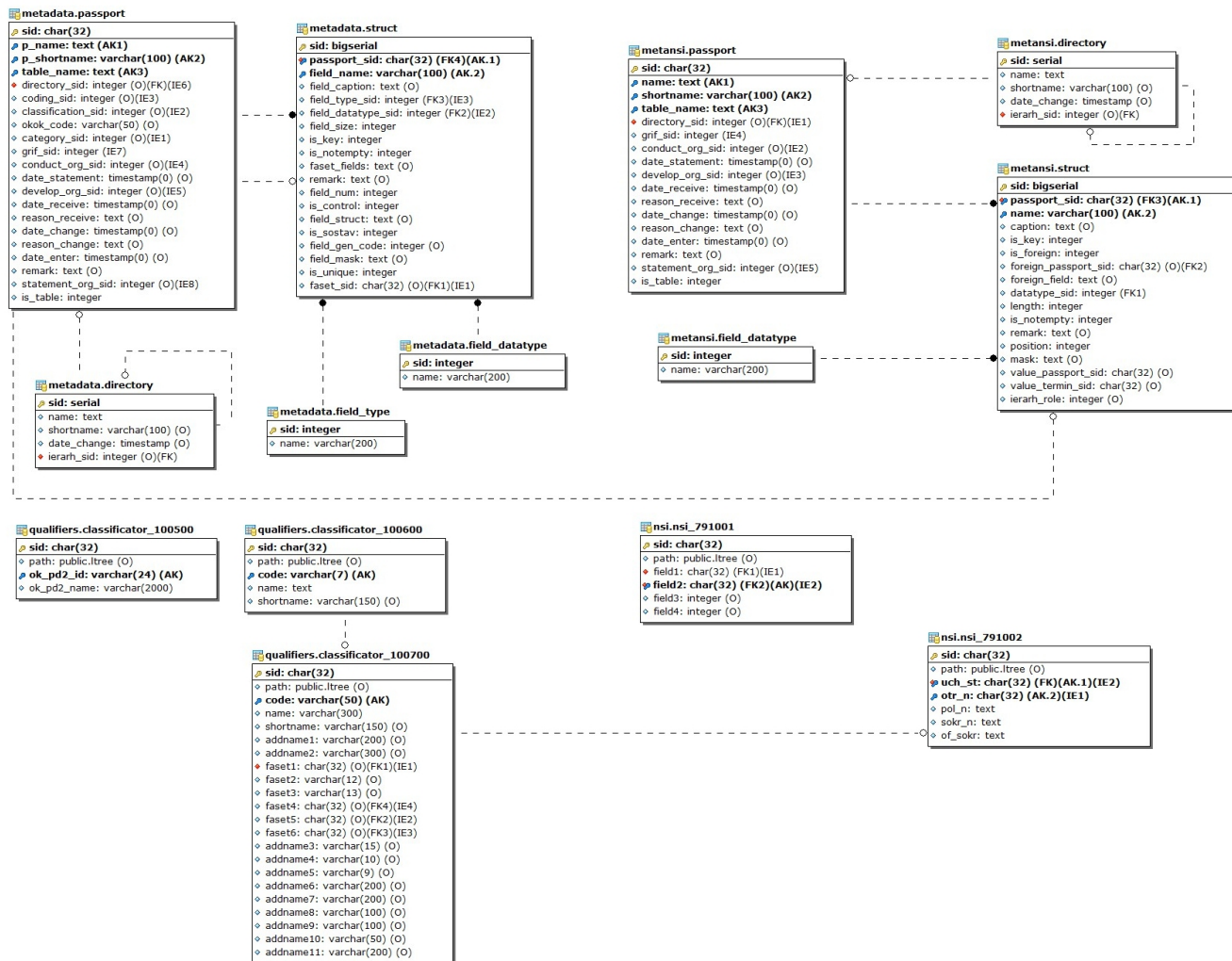


Рисунок 8 – Диаграмма базы данных

Ниже представлены примеры создания таблиц классификаторов и НСИ:

1. Создание паспорта классификатора:

```
CREATE TABLE metadata.passport (
    sid CHAR(32) NOT NULL,
    p_name TEXT NOT NULL,
    p_shortcode VARCHAR(100) NOT NULL,
    table_name TEXT NOT NULL,
    directory_sid INTEGER,
```

```

coding_sid INTEGER,
classification_sid INTEGER,
okok_code VARCHAR(50),
category_sid INTEGER,
grif_sid INTEGER NOT NULL,
conduct_org_sid INTEGER,
date_statement TIMESTAMP(0) WITHOUT TIME ZONE,
develop_org_sid INTEGER,
date_receive TIMESTAMP(0) WITHOUT TIME ZONE,
reason_receive TEXT,
date_change TIMESTAMP(0) WITHOUT TIME ZONE,
reason_change TEXT,
date_enter TIMESTAMP(0) WITHOUT TIME ZONE,
remark TEXT,
statement_org_sid INTEGER,
is_table INTEGER DEFAULT 0 NOT NULL
)

```

2. Создание структуры классификатора:

```

CREATE TABLE metadata.struct (
    sid BIGSERIAL,
    passport_sid CHAR(32) NOT NULL,
    field_name VARCHAR(100) NOT NULL,
    field_caption TEXT,
    field_type_sid INTEGER NOT NULL,
    field_datatype_sid INTEGER NOT NULL,
    field_size INTEGER DEFAULT 0 NOT NULL,
    is_key INTEGER NOT NULL,
    is_notempty INTEGER NOT NULL,
    faset_fields TEXT,
    remark TEXT,
    field_num INTEGER NOT NULL,
    is_control INTEGER NOT NULL,
    field_struct TEXT,
    is_sostav INTEGER NOT NULL,
    field_gen_code INTEGER,
    field_mask TEXT,
    is_unique INTEGER NOT NULL,
    faset_sid CHAR(32)
)

```

3. Создание каталогизации классификатора:

```

CREATE TABLE metadata.directory (
    sid SERIAL,
    name TEXT NOT NULL,
    shortname VARCHAR(100),
    date_change TIMESTAMP WITHOUT TIME ZONE DEFAULT 'now'::text::timestamp(6)
with time zone,
    ierarh_sid INTEGER
)

```

4. Создание термина классификатора:

```
CREATE TABLE qualifiers.classificator_100500 (
  sid CHAR(32) NOT NULL,
  path public.ltree,
  ok_pd2_id VARCHAR(24) NOT NULL,
  ok_pd2_name VARCHAR(2000) NOT NULL
)
```

5. Создание паспорта НСИ:

```
CREATE TABLE metansi.passport (
  sid CHAR(32) NOT NULL,
  name TEXT NOT NULL,
  shortname VARCHAR(100) NOT NULL,
  table_name TEXT NOT NULL,
  directory_sid INTEGER,
  grif_sid INTEGER NOT NULL,
  conduct_org_sid INTEGER,
  date_statement TIMESTAMP(0) WITHOUT TIME ZONE,
  develop_org_sid INTEGER,
  date_receive TIMESTAMP(0) WITHOUT TIME ZONE,
  reason_receive TEXT,
  date_change TIMESTAMP(0) WITHOUT TIME ZONE,
  reason_change TEXT,
  date_enter TIMESTAMP(0) WITHOUT TIME ZONE,
  remark TEXT,
  statement_org_sid INTEGER,
  is_table INTEGER DEFAULT 0 NOT NULL
)
```

6. Создание структуры НСИ:

```
CREATE TABLE metansi.struct (
  sid BIGSERIAL,
  passport_sid CHAR(32) NOT NULL,
  name VARCHAR(100) NOT NULL,
  caption TEXT,
  is_key INTEGER NOT NULL,
  is_foreign INTEGER NOT NULL,
  foreign_passport_sid CHAR(32),
  foreign_field TEXT,
  datatype_sid INTEGER NOT NULL,
  length INTEGER DEFAULT 0 NOT NULL,
  is_notempty INTEGER NOT NULL,
  remark TEXT,
  "position" INTEGER DEFAULT 1 NOT NULL,
  mask TEXT,
  value_passport_sid CHAR(32),
  value_termin_sid CHAR(32),
  ierarh_role INTEGER
)
```

7. Создание каталогизации НСИ:

```
CREATE TABLE metansi.directory (
  sid SERIAL,
  name TEXT NOT NULL,
  shortname VARCHAR(100),
  date_change TIMESTAMP WITHOUT TIME ZONE DEFAULT 'now'::text::timestamp(6)
with time zone,
  ierarh_sid INTEGER
)
```

8. Создание термина НСИ:

```
CREATE TABLE nsi.nsi_791001 (
  sid CHAR(32) NOT NULL,
  path public.ltree,
  field1 CHAR(32) NOT NULL,
  field2 CHAR(32) NOT NULL,
  field3 INTEGER,
  field4 INTEGER
)
```

Ниже представлены хранимые процедуры для работы с таблицами классификаторов и НСИ:

1. Создание паспорта классификатора:

Входные параметры:

- Наименование;
- Сокращенное наименование;
- Наименование таблицы базы данных;
- Директория;
- Система кодирования;
- Классификация;
- Код по ОККО;
- Категория;
- Гриф секретности;
- Организация ответственная за ведение;
- Дата утверждения;
- Организация ответственная за разработку;
- Дата получения;

- Основание для получения;
- Дата изменения;
- Основание для изменения;
- Дата ввода в действие;
- Примечание;
- Организация ответственная за утверждение;

Выходные данные:

- Идентификатор паспорта классификатора - успешно;
- 1 - наименование таблицы классификатора пустое/уже существует;
- 2 - наименование классификатора пустое/уже существует;
- 3 - гриф секретности пустой/не существует.

Код функции представлен в приложении В, п.1.

2. Создание структуры классификатора:

Входные параметры:

- паспорт классификатора;
- наименование поля;
- размер поля;
- порядковый номер поля;
- комментарий;
- тип поля;
- является ли поле ключевым (0-нет;1-да);
- фасетный классификатор;
- список полей фасетного классификатора;
- тип данных поля;
- является ли поле контрольным числом (0-нет;1-да);
- структура поля;
- является ли поле составным (0-нет;1-да);
- алгоритм генерации поля (1-Общероссийский,2-КВТМО);

- маска ввода;
- заголовок поля;
- поле не пустое (0-нет;1-да);
- поле уникальное (0-нет;1-да).

Выходные параметры:

- 1 - ошибка создания записи;
- 2 - паспорт классификатора пустой или не существует;
- 3 - тип поля пустой или не существует;
- 4 - поле с таким наименованием уже существует;
- 5 - фасетного классификатора не существует;
- 0 - идентификатор новой записи.

Код функции представлен в приложении В, п.2.

3. Создание термина классификатора:

Входные параметры:

- идентификатор паспорта классификатора;
- массив значений.

Выходные данные:

- Идентификатор термина классификатора - успешно;
- 1 - таблицы классификатора не существует;
- 2 - пустой массив значений;
- 3 - ссылочного термина не существует;
- 4 - размер поля больше чем в структуре;
- 5 - термин уже существует.

Код функции представлен в приложении В, п.3.

4. Создание каталога классификатора:

Входные параметры:

- наименование;
- сокращенное наименование;

- идентификатор старшей директории.

Выходные данные:

- Идентификатор каталога - успешно;
- 1 - пустое наименование;
- 2 - на заданном уровне каталог уже существует;
- 3 - старшего каталога не существует.

Код функции представлен в приложении Г, п.4.

5. Каскадное удаление паспорта классификатора:

Входные параметры:

- Идентификатор паспорта классификатора.

Выходные данные:

- 0 - успешно;
- 1 - нет доступа к классификатору/классификатора не существует.

Код функции представлен в приложении В, п.5.

6. Создание паспорта НСИ:

Входные параметры:

- Наименование;
- Сокращенное наименование;
- Наименование таблицы базы данных;
- Директория;
- Гриф секретности;
- Организация ответственная за ведение;
- Дата утверждения;
- Организация ответственная за разработку;
- Дата получения;
- Основание для получения;
- Дата изменения;
- Основание для изменения;

- Дата ввода в действие;
- Примечание;
- Организация ответственная за утверждение.

Выходные данные:

- Идентификатор паспорта НСИ - успешно;
- 1 - наименование таблицы НСИ пустое/уже существует;
- 2 - наименование НСИ пустое/уже существует;
- 3 - сокращенное наименование НСИ пустое/уже существует;
- 4 - гриф секретности пустой/не существует.

Код функции представлен в приложении В, п.6.

7. Создание структуры НСИ:

Входные параметры:

- паспорт НСИ;
- наименование;
- заголовок;
- является ли поле ключевым;
- является ли поле ссылочным;
- паспорт классификатора ;
- отображаемое поле классификатора;
- тип данных;
- длина;
- поле не пустое (0-нет;1-да);
- примечание;
- номер по порядку;
- маска ввода;
- роль в построении иерархии;
- порядковый номер;
- максимальный порядковый номер.

Выходные параметры:

- 1 - ошибка создания записи;
- 2 - паспорт НСИ пустой или не существует;
- 3 - тип поля пустой или не существует;
- 4 - поле с таким наименованием уже существует;
- 5 - фасетного классификатора не существует;
- 0 - идентификатор новой записи.

Код функции представлен в приложении В, п.7.

8. Создание новых терминов НСИ:

Входные параметры:

- паспорт НСИ;
- массив значений.

Выходные данные:

- 1 - таблицы НСИ не существует;
- 2 - пустой массив значений;
- 3 - ссылочного термина не существует;
- 4 - размер значения поля превышает допустимое значение;
- идентификатор термина – успешно.

Код функции представлен в приложении Г, п.8.

9. Удаление паспорта НСИ:

Входные параметры:

- идентификатор паспорта НСИ.

Выходные данные:

- 0 - успешно;
- 1 - паспорта НСИ не существует.

Код функции представлен в приложении В, п.9.

3 Разработка программного обеспечения

В соответствии с поставленной задачей была построена модель прецедентов предметной области. Для понимания взаимодействия прецедентов друг с другом наилучшим вариантом будет построение диаграммы прецедентов, которая представлена на рисунке 9.

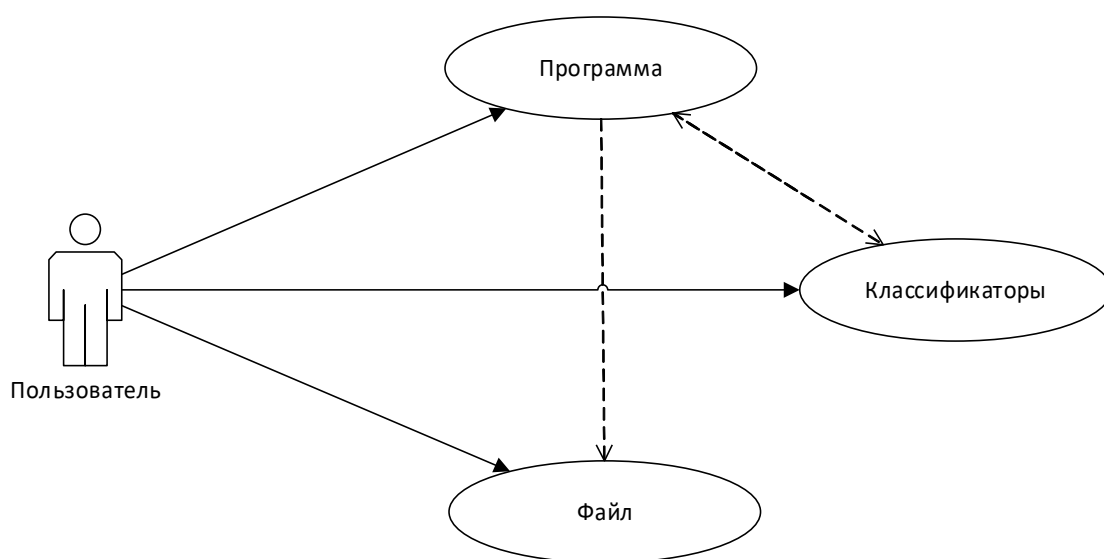


Рисунок 9 - Диаграмма прецедентов

3.1 Описание основных классов

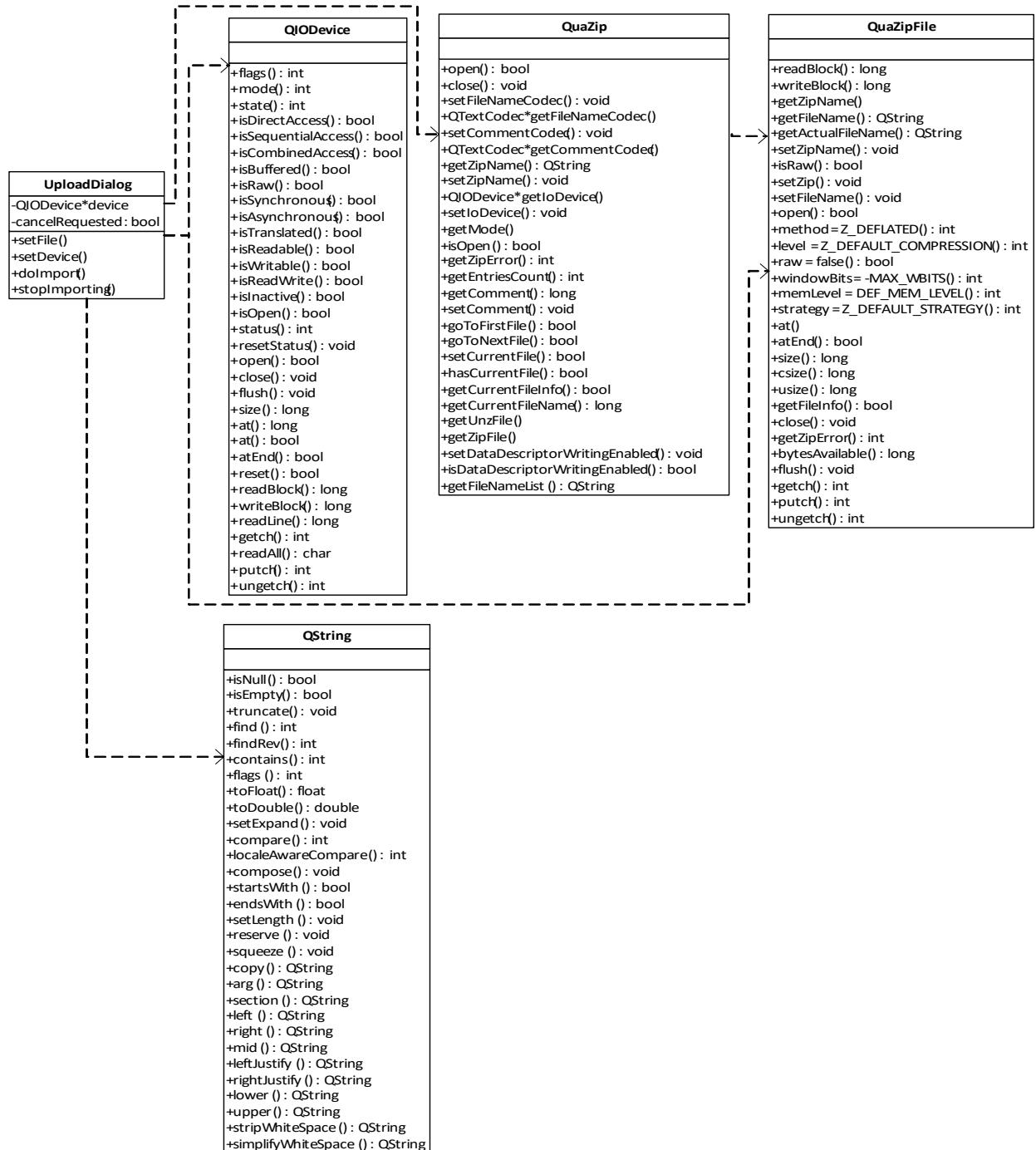


Рисунок 10 – Диаграмма классов (загрузка)

Описание класса:

Загрузка

1) **UploadDialog** – класс диалога загрузки в файл.

Атрибуты:

- `private QIODevice*device` – Устройство из которого производится загрузка (файл сокет и т.д.);
- `private bool cancelRequested` – Индикатор запроса отмены загрузки пользователем.

Методы:

- `void UploadDialog::setFile(const QString &file)` – Установка имени файла для загрузки;
-
- `void UploadDialog::setDevice(QIODevice *device)` – Установка устройства для загрузки (файл, сокет и т.д.);
- `bool UploadDialog::doImport()` – Возвращает успешность загрузки (метод загрузки);
-
- `void UploadDialog::stopImporting()` – Отменяет загрузку (останов процесса).

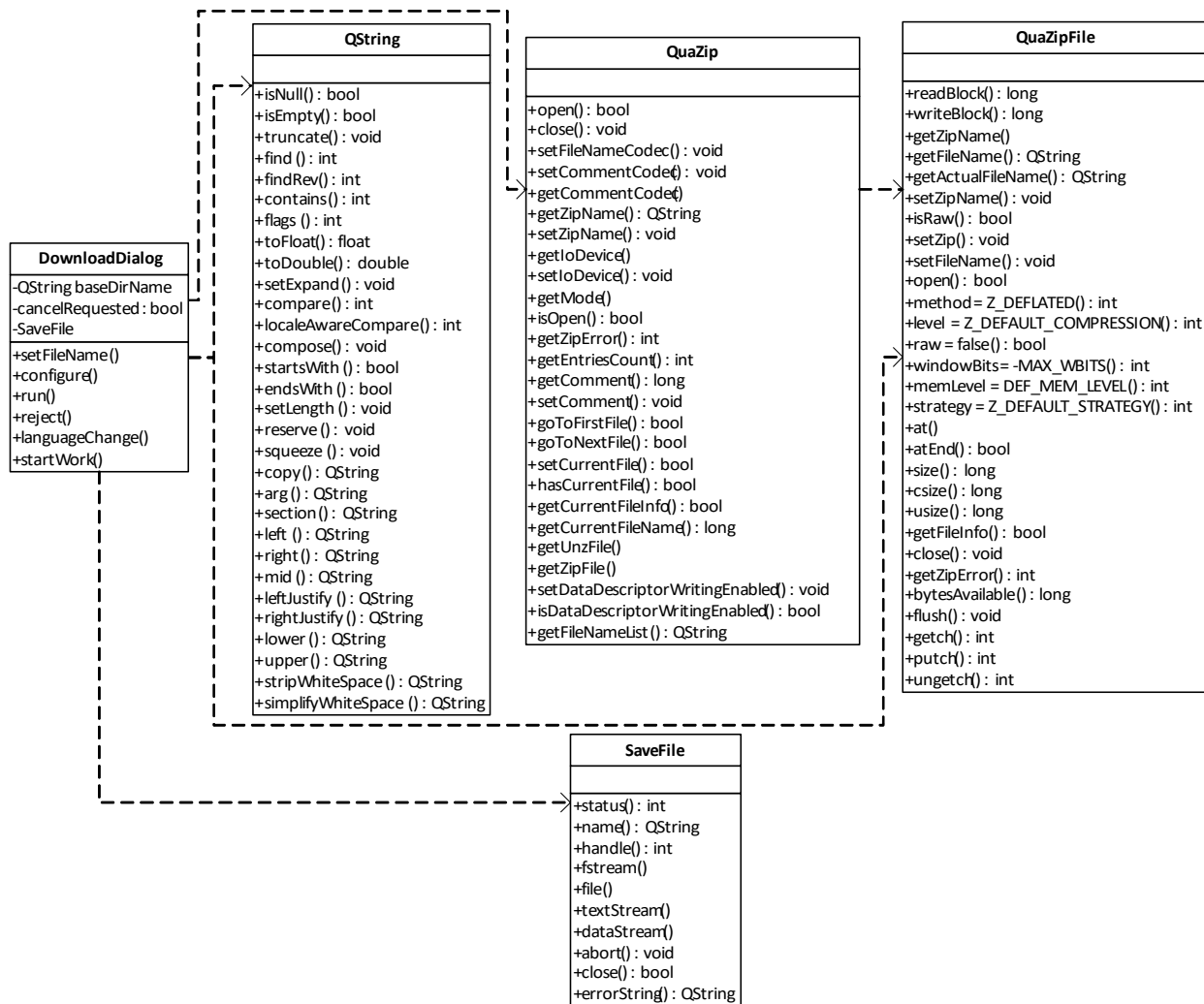


Рисунок 11 – Диаграмма классов (выгрузка)

Описание класса:

Выгрузка

1) **DownloadDialog** – класс диалога выгрузки в файл.

Атрибуты:

- `private bool cancelRequested` – Индикатор запроса отмены загрузки пользователем;
- `private QString baseDirName` – Имя каталога внутри архива;
- `private SaveFile`– Файл, в который происходит выгрузка.

Методы:

- `void DownloadDialog::setFileName(const QString &fileName)` – Установка имени файла для выгрузки;
- `bool DownloadDialog::configure()` – Запуск дополнительного диалога настройки выгрузки;
-
- `int DownloadDialog::run()` – Запуск выгрузки;
- `void DownloadDialog::reject()` – Отмена выгрузки;
- `void DownloadDialog::languageChange()` – Установка языка интерфейса;
- `void DownloadDialog::startWork()` – Старт процесса выгрузки.

Для детального анализа взаимодействий разработанных на этапе анализа предметной области сущностей удобно использовать диаграмму последовательностей, которая отображена на рисунке 12. На данной диаграмме последовательности изображен порядок создания (выгрузки) пакета актуализации.

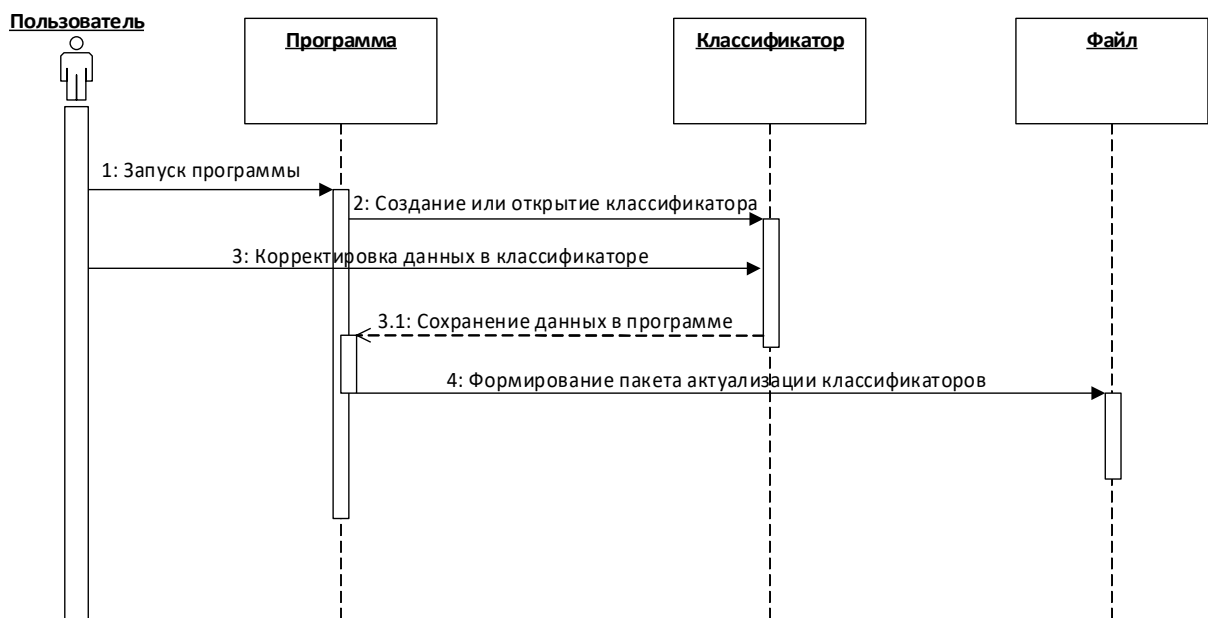


Рисунок 12 - Диаграмма последовательности (выгрузка)

На диаграмме последовательности (рисунок 13) изображен порядок загрузки пакета актуализации.

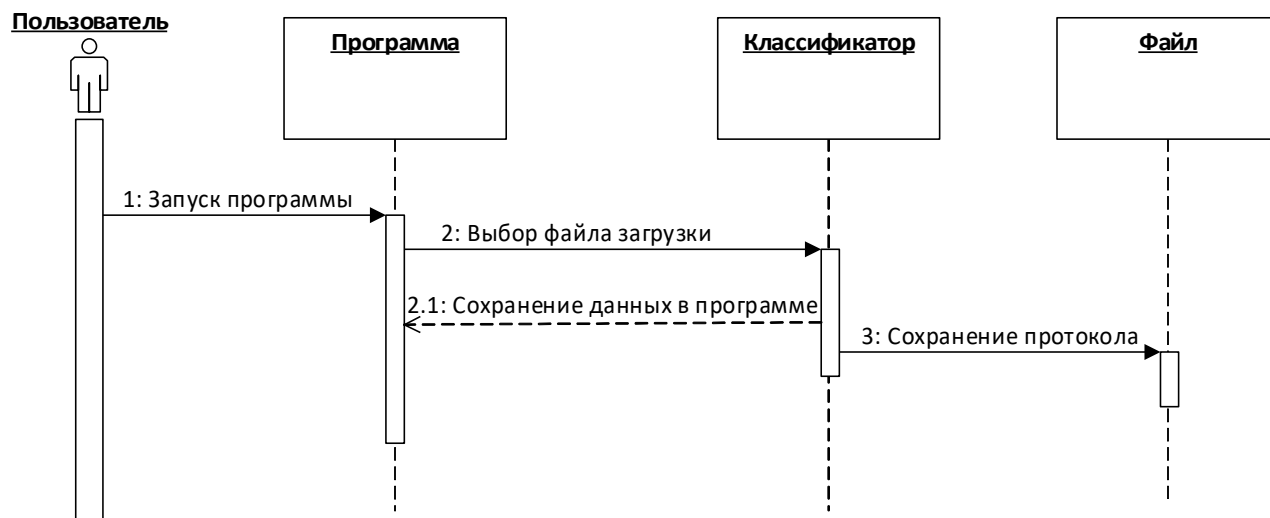


Рисунок 13 - Диаграмма последовательности (загрузка)

3.2 Схема алгоритма

Схема алгоритма выгрузки представлена на рисунке 14.

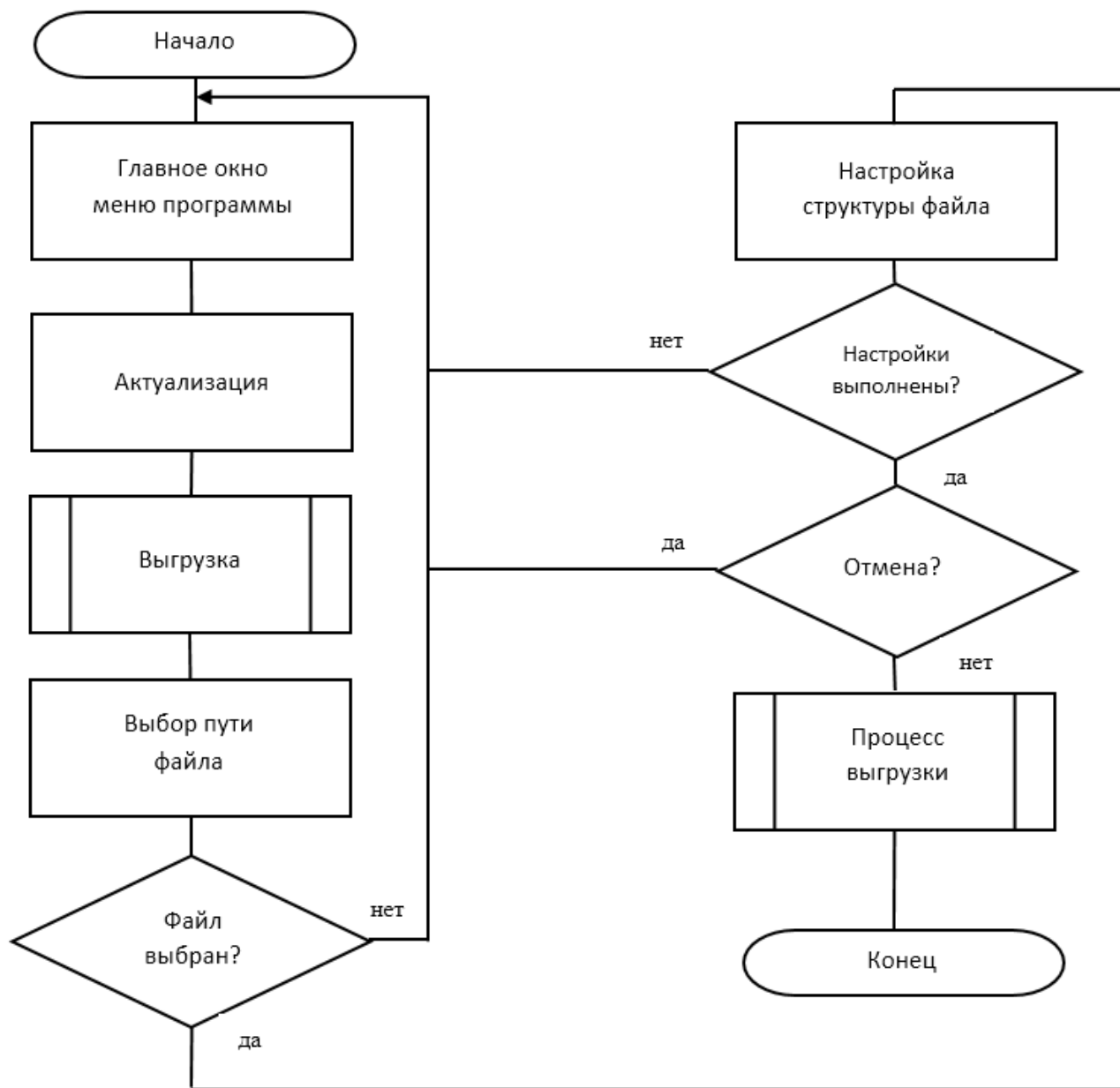


Рисунок 14 –Схема алгоритма выгрузки

Схема алгоритма загрузки представлена на рисунке 15.

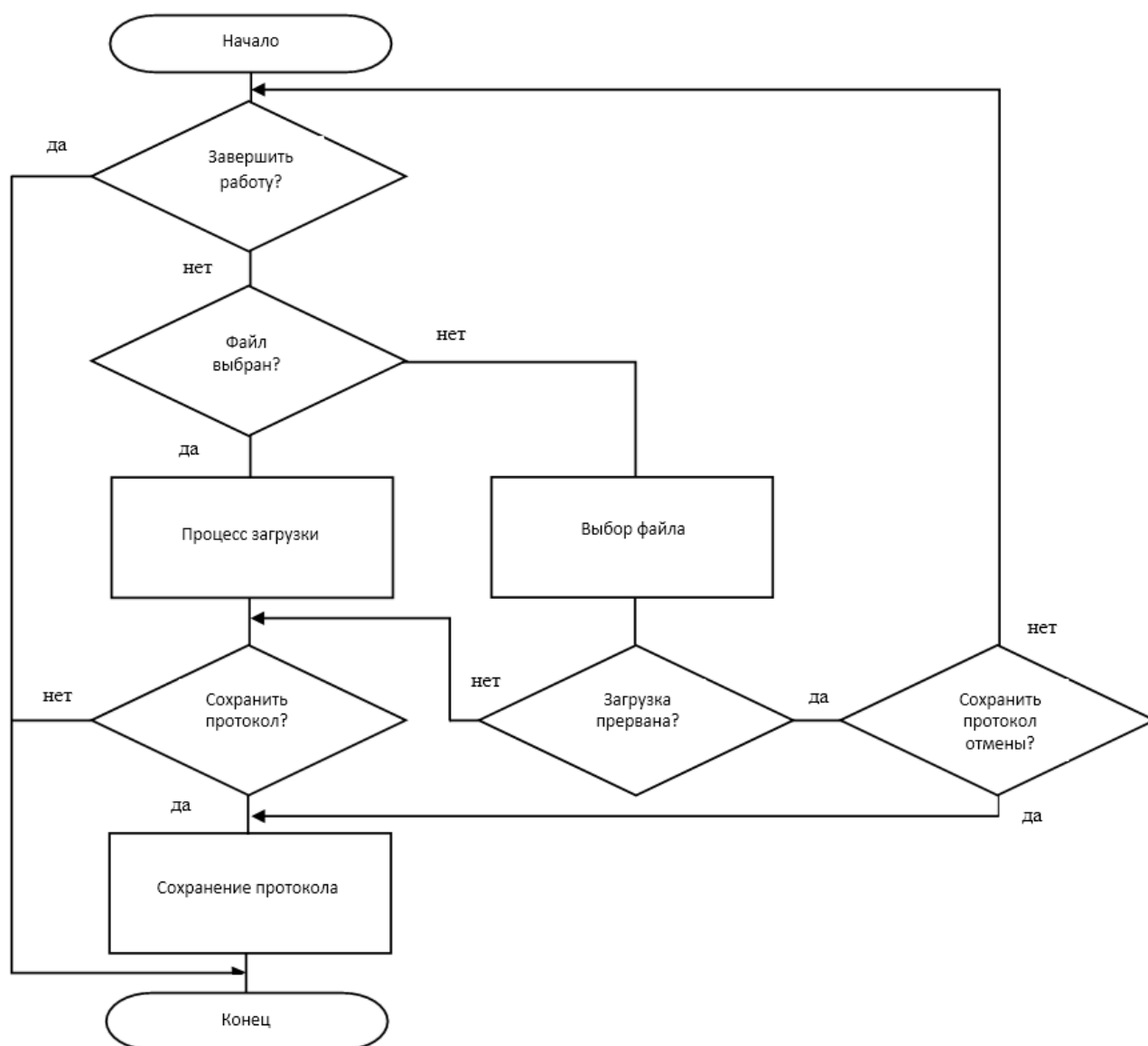


Рисунок 15 –Схема алгоритма загрузки

4 Тестирование программного обеспечения

На рисунке 16 отображено главное окно программы.

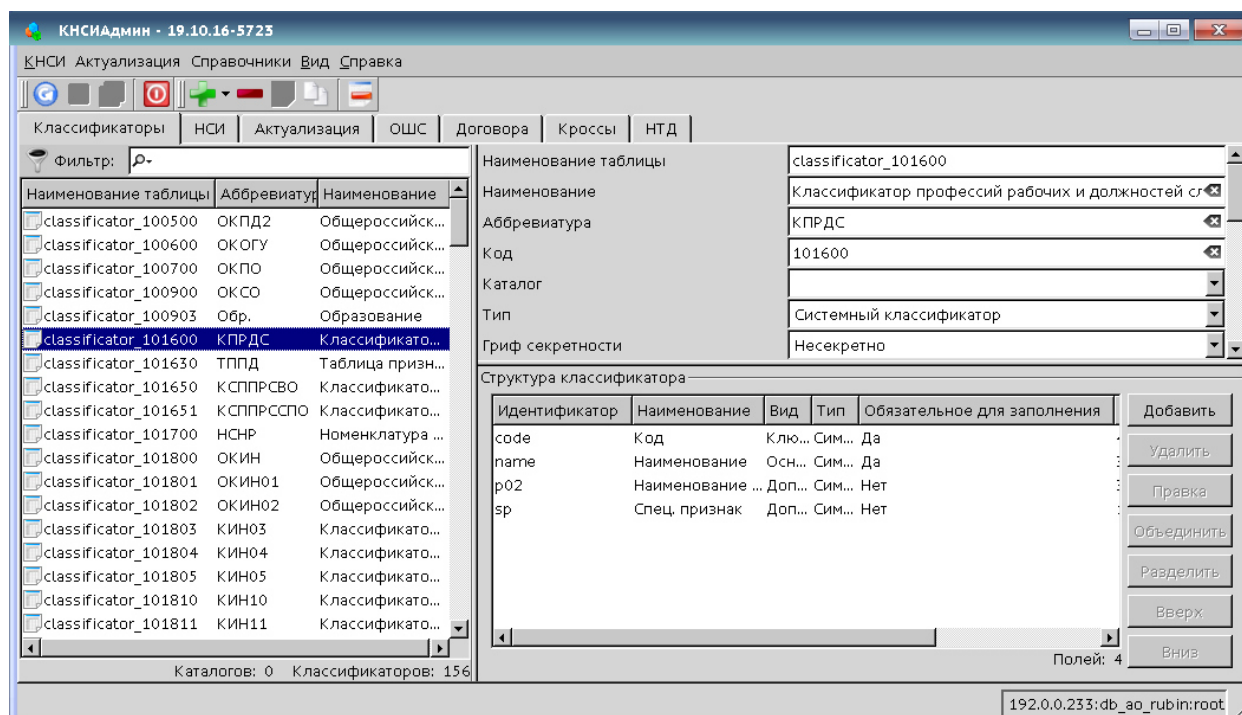


Рисунок 16 – Главное окно программы

Для выгрузки пакета актуализации классификаторов необходимо выбрать последовательно пункты меню программы КНСИ «Актуализация/Выгрузка» (рисунок 17).

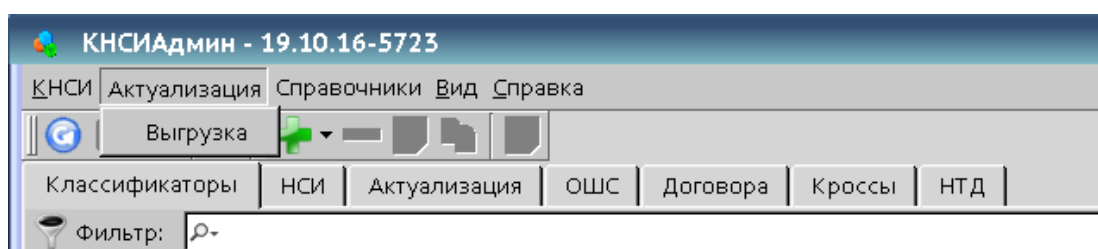


Рисунок 17 – Пункты меню программы КНСИ

В открывшемся окне (рисунок 18) необходимо выбрать путь сохранения пакета актуализации и нажать на кнопку «Сохранить».

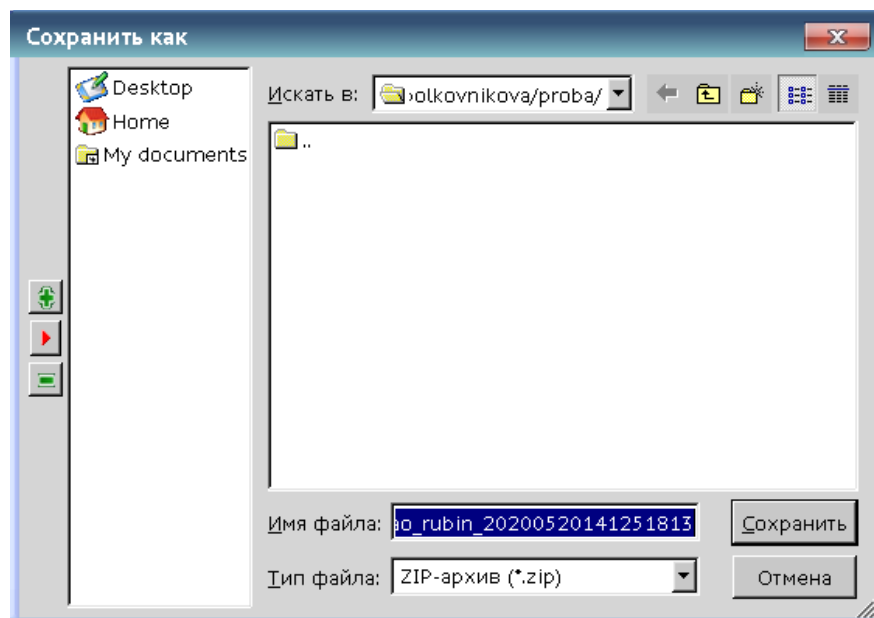


Рисунок 18 – Путь сохранения пакета актуализации

Если пакет актуализации с таким названием уже существует, после нажатия на кнопку «Сохранить» появляется окно, представленное на рисунке 19.

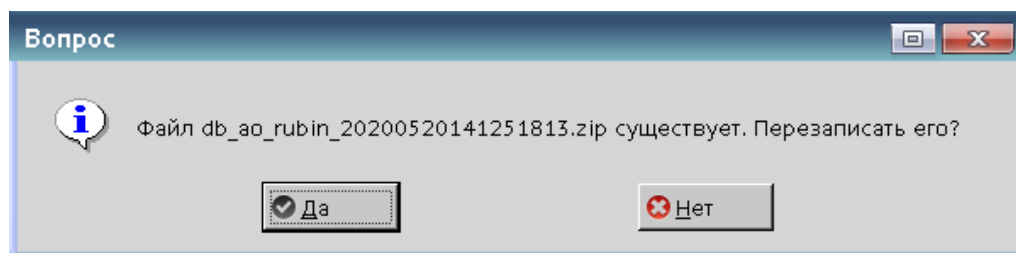


Рисунок 19 – Перезапись файла

При нажатии на кнопку «Нет» снова появляется окно сохранения файла с возможностью изменить название.

Далее в результате появится окно с настройками содержимого пакета (рисунок 20), в котором необходимо выбрать, что необходимо выгрузить и нажать на кнопку «Принять».

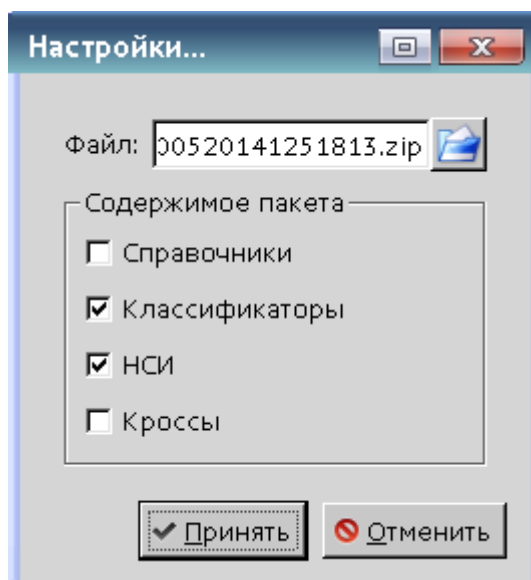


Рисунок 20 – Настройка выгрузки пакета актуализации

Откроется окно выгрузки файла актуализации, представленное на рисунке 21.

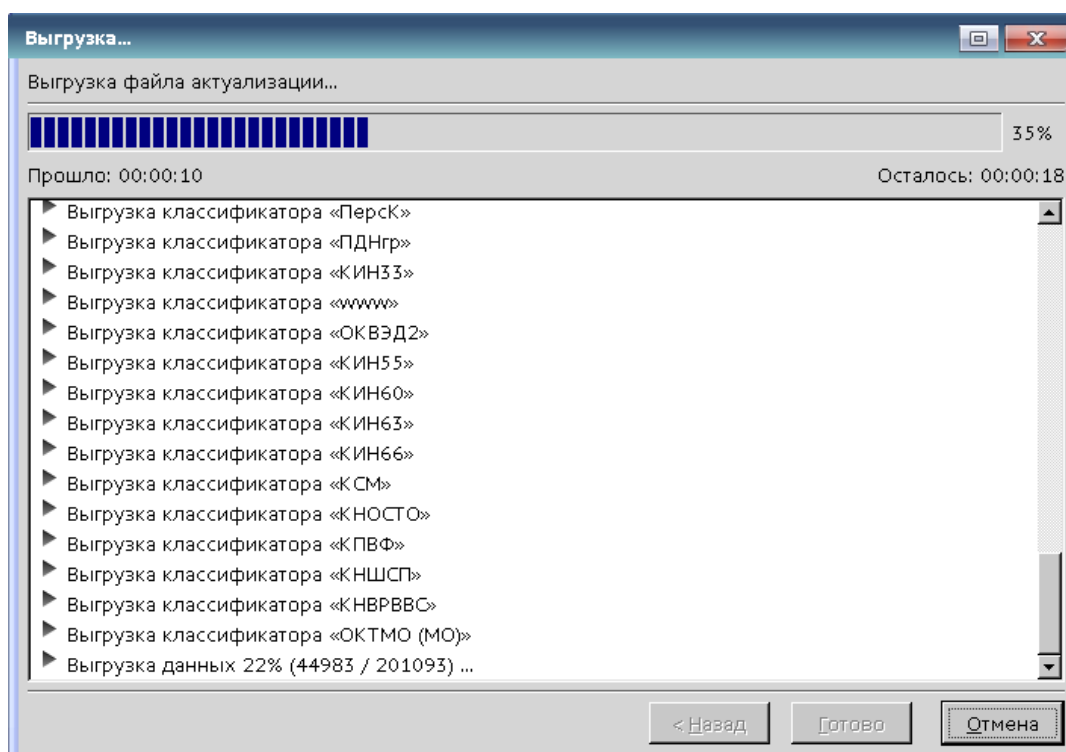


Рисунок 21 – Окно выгрузки файла актуализации

После выгрузки станет доступна кнопка «Готово».

В сохраненном архиве будет выгружено, что было отмечено в настройках содержимого пакета (рисунок 22).

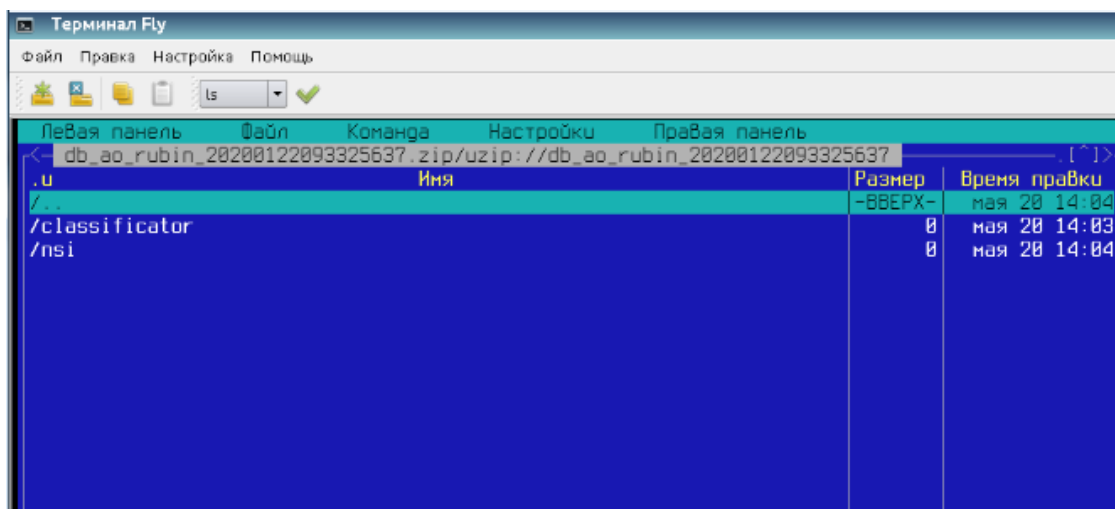


Рисунок 22 – Содержимое архива

При загрузке файла актуализации при нажатии на кнопку «Отмена» (рисунок 23) появится окно, представленное на рисунке 24.

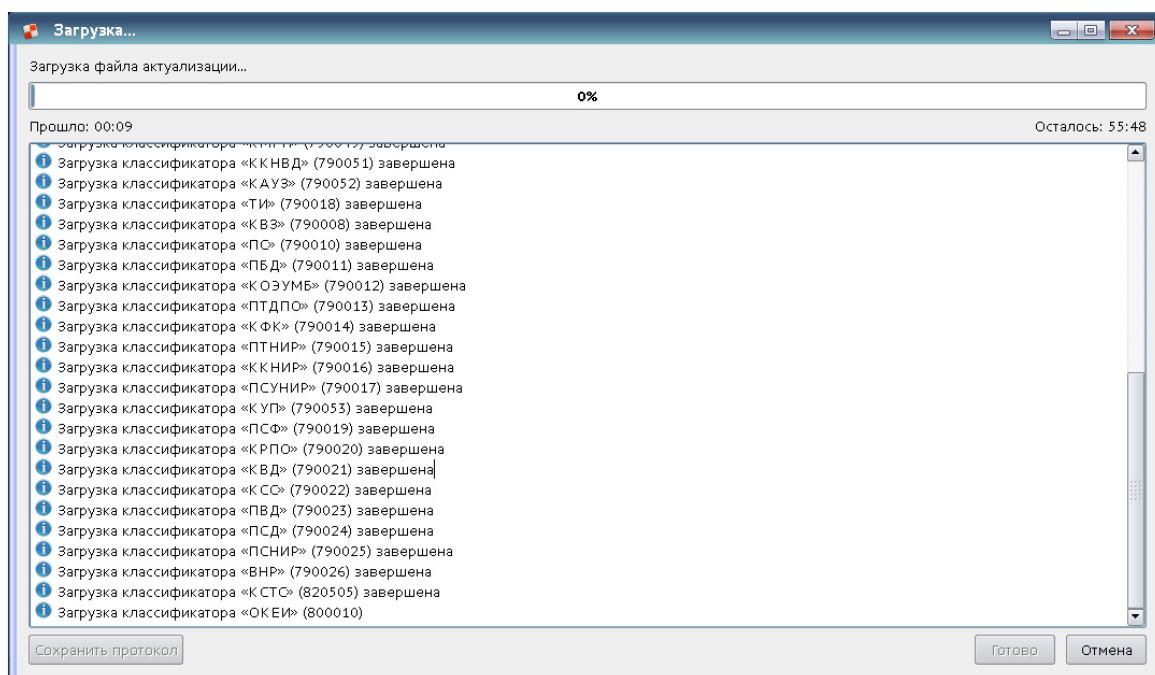


Рисунок 23 – Окно загрузки файла актуализации

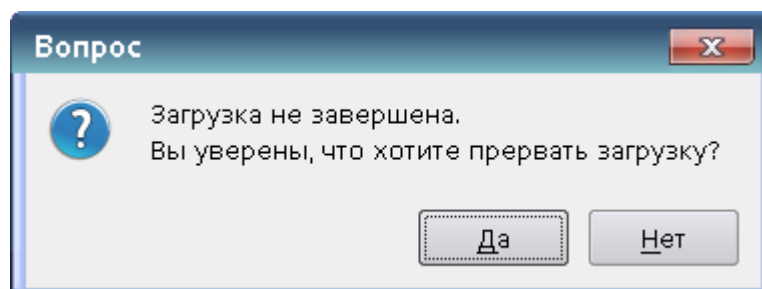


Рисунок 24 – Прерывание загрузки

При нажатии на кнопку «Да» прекращается загрузка, в результате чего появляется строка «Отменено пользователем» (рисунок 25).

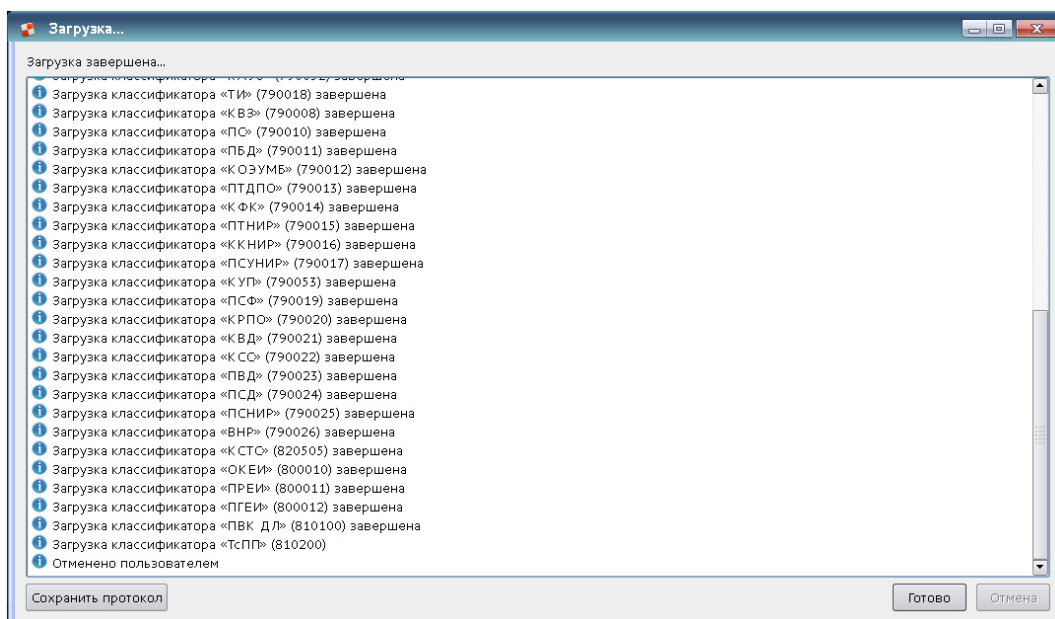


Рисунок 25 – Отмена загрузки пользователем

При нажатии на кнопку «Сохранить протокол» появляется окно, представленное на рисунке 26.

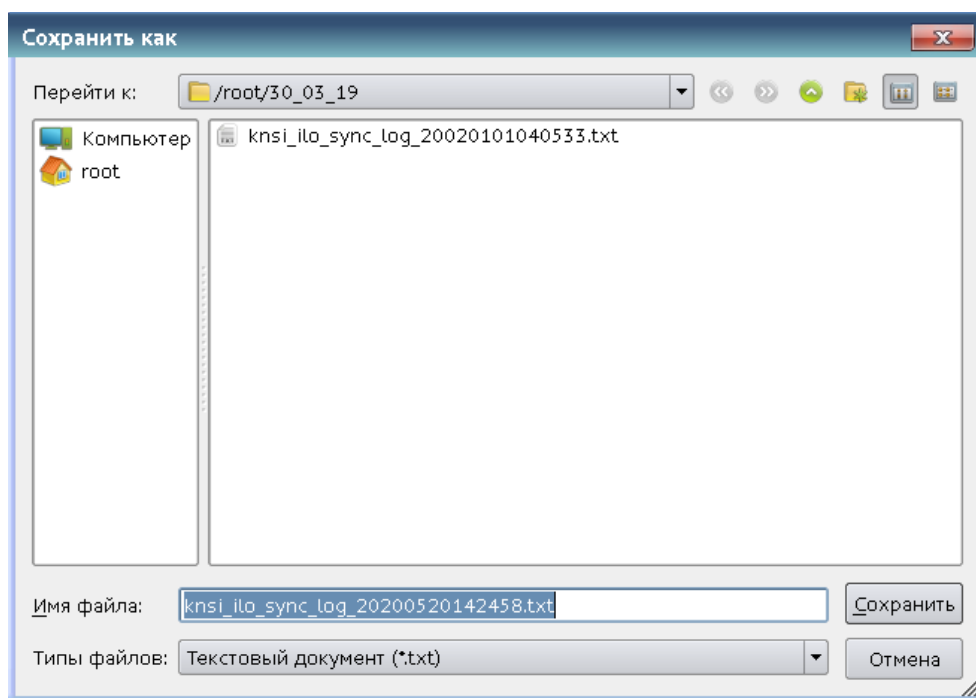


Рисунок 26 – Путь сохранения протокола загрузки

Заключение

В рамках данной выпускной квалификационной работы были разработаны программные модули выгрузки и загрузки пакетов актуализации классификаторов, НСИ и кроссов.

Был проведен анализ предметной области, а также сравнение с существующими на данный момент аналогами – АСУ НСИ 2.0 и программой 1С.

Отличительной особенностью является многоплатформенность и форма выгрузки данных.

В ходе проектирования разработаны диаграммы классов, прецедентов, последовательности и блок-схемы алгоритмов.

Проверено функционирование в средах ОС Microsoft Windows и ОС Astra Linux.

Программные модули загрузки и выгрузки пакетов актуализации были разработаны на языке программирования C++ с использованием библиотеки Qt. Для описания базы данных использован язык PSQL.

Проект выполнен в точном соответствии с поставленным техническим заданием и соблюдением календарно-технического плана.

Список использованных источников

1. Полковникова С.А., Кулакова Е.В., Полковников Е.А. СИСТЕМА ВЕДЕНИЯ КЛАССИФИКАТОРОВ И АКТУАЛИЗАЦИЯ ИНФОРМАЦИИ В НИХ // Студенческий: электрон. научн. журн. 2020. № 18(104). URL: <https://sibac.info/journal/student/104/178605> (Дата обращения 07.04.2020).
2. Виды классификаторов. [Электронный ресурс] // URL: <https://rosstandart.msk.ru/poleznaya-informaciya/chto-takoe-klassifikatory-vidy-klassifikatorov/> (Дата обращения 15.01.2020)
3. Стивен Прата. Язык программирования C++ (C++11). Лекции и упражнения, 6-е издание — М.: Вильямс, 2012. — 1248 с. (Дата обращения 03.02.2020).
4. Программирование с Qt. [Электронный ресурс] // URL: https://www.ibm.com/developerworks/ru/library/l-qt_1/ (Дата обращения 15.03.2020).
5. Документация по PostgreSQL . [Электронный ресурс] // URL: <https://postgrespro.ru/docs/postgresql/9.4/sql> (Дата обращения 20.09.2019)

ПРИЛОЖЕНИЕ А. ФОРМИРОВАНИЕ ПАКЕТОВ АКТУАЛИЗАЦИИ

```

/*****
 *
 * Copyright (C) 2019 Rubin and/or its subsidiary(-ies).
 * Contact: http://www.npp-rubin.ru
 *
 * The program is provided AS IS with NO WARRANTY OF ANY KIND, INCLUDING THE
 * WARRANTY OF DESIGN, MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.
 *
 *****/

#ifndef KNSIADMIN_ACTUALIZATION_UPLOADDIALOG_H
#define KNSIADMIN_ACTUALIZATION_UPLOADDIALOG_H

#include <actualization/actualization_export.h>
#include <core/importwizard.h>

namespace knsiadmin {
namespace actualization {

class ACTUALIZATION_EXPORT UploadDialog : public ImportWizard
{
    Q_OBJECT
public:
    UploadDialog(QWidget *parent = 0, const char *name = 0);
    ~UploadDialog();

    void setFile(const QString &file);
    void setDevice(QIODevice *device);

protected:
    bool doImport();
    void stopImporting();

private:
    class Private;
    Private *const d;

    friend class Private;
};

} // namespace actualization
} // namespace knsiadmin

#endif // KNSIADMIN_ACTUALIZATION_UPLOADDIALOG_H

*****/

#include "uploaddialog.h"

#include <actualization/log.h>

#include <core/i18n/locale.h>
#include <core/progresswidget.h>
#include <core/application.h>
#include <core/helpers.h>

```

```

#include <core/stringbuilder.h>

#include <quazip/quazip.h>
#include <quazip/quazipfile.h>
#include <quazip/quazipdir.h>

#include <pqxx/pqxx>

#include <boost/foreach.hpp>

#include <qtimer.h>
#include <qdir.h>
#include <qpushbutton.h>
#include <qdom.h>

namespace knsiadmin {
namespace actualization {

static bool IsListIntersected(const QStringList &left, const QStringList
&right)
{
    BOOST_FOREACH(QString entry, right) {
        if(left.constEnd() != left.find(entry))
            return true;
    }
    return false;
}

class UploadDialog::Private {
public:
    UploadDialog *q;
    QIODevice *device;
    bool cancelRequested;

    QuaZipDir rootZipDir(QuaZip *zip, bool *ok = 0) const
    {
        static const QStringList markers = QStringList() <<
"classificator/" << "nsi/" << "cross/"
<<
"directories/";
        QuaZipDir dir(zip);
        QStringList entryList = dir.entryList();
        if(IsListIntersected(entryList, markers)) {
            if(ok)
                *ok = true;
            return dir;
        }
        BOOST_FOREACH(QString entry, entryList) {
            if(entry.endsWith("/")) {
                QuaZipDir subdir(zip, dir.filePath(entry));
                if(IsListIntersected(subdir.entryList(), markers)) {
                    if(ok)
                        *ok = true;
                    return subdir;
                }
            }
        }
    }
}
}

```

```

        if(ok)
            *ok = false;
        return dir;
    }

    bool importDirectories(QuaZip *zip, const QString &dirPath);
    bool importClassifiers(QuaZip *zip, const QString &dirPath);
    bool importNsis(QuaZip *zip, const QString &dirPath);
    bool importCrosses(QuaZip *zip, const QString &dirPath);

    bool importDirectory(QIODevice *device, const QString &fileName);
};

UploadDialog::UploadDialog(QWidget *parent, const char *name)
    : ImportWizard(parent, name)
    , d(new Private)
{
    d->q = this;
    d->device = 0;
    d->cancelRequested = false;

    languageChange();
}

UploadDialog::~~UploadDialog()
{
    delete d;
}

void UploadDialog::setFile(const QString &file)
{
    Q_UNUSED(file);
}

void UploadDialog::setDevice(QIODevice *device)
{
    d->device = device;
}

bool UploadDialog::doImport()
{
    d->cancelRequested = false;
    if(!d->device) {
        return false;
    }

    QuaZip zip(d->device);
    if(!zip.open(QuaZip::mdUnzip)) {
        error(i18n("File \"%1\" is not ZIP-archive"));
        return false;
    }

    bool ok;
    QuaZipDir dir = d->rootZipDir(&zip, &ok);
    if(!ok) {
        error(i18n("File \"%1\" has unknown structure"));
    }
}

```

```

        return false;
    }

    ok = ok && !d->cancelRequested;
    ok = ok && dir.exists("directories/") && !d->importDirectories(&zip,
dir.filePath("directories/"));
    ok = ok && !d->cancelRequested;
    ok = ok && dir.exists("classifier/") && !d->importClassifiers(&zip,
dir.filePath("classifier/"));
    ok = ok && !d->cancelRequested;
    ok = ok && dir.exists("nsi/") && !d->importNsis(&zip,
dir.filePath("nsi/"));
    ok = ok && !d->cancelRequested;
    ok = ok && dir.exists("cross/") && !d->importCrosses(&zip,
dir.filePath("cross/"));
    ok = ok && !d->cancelRequested;

    return ok;
}

void UploadDialog::stopImporting()
{
    d->cancelRequested = true;
}

bool UploadDialog::Private::importDirectories(QuaZip *zip, const QString
&dirPath)
{
    QuaZipDir dir(zip, dirPath);

    QStringList fileNameList = dir.entryList(QDir::Files, QDir::Name);
    BOOST_FOREACH(QString fileName, fileNameList) {
        zip->setCurrentFile(dir.filePath(fileName));
        QuaZipFile file(zip);
        if(!file.open(IO_ReadOnly)) {
            q->error(i18n("Error opening file \"%1\"").arg(zip-
>getCurrentFileName()));
            return false;
        }
        if(!importDirectory(&file, zip->getCurrentFileName()))
            return false;
    }
    return true;
}

bool UploadDialog::Private::importClassifiers(QuaZip *zip, const QString
&dirPath)
{
    return true;
}

bool UploadDialog::Private::importNsis(QuaZip *zip, const QString &dirPath)
{
    return true;
}

```

```

bool UploadDialog::Private::importCrosses(QuaZip *zip, const QString
&dirPath)
{
    return true;
}

bool UploadDialog::Private::importDirectory(QIODevice *device, const QString
&fileName)
{
    QDomDocument doc;
    QString errorMessage;
    int errorLine;
    int errorColumn;
    if(!doc.setContent(device, &errorMessage, &errorLine, &errorColumn)) {
        q->error(i18n("Syntax Error file %1 line %2 column %3<br/>%4")
                .arg(fileName)
                .arg(errorLine)
                .arg(errorColumn)
                .arg(errorMessage));
        return false;
    }

    QDomElement rootElement = doc.documentElement();
    const QString tableName = rootElement.nodeName();
    QString keyField = rootElement.attribute("key");
    QString name = rootElement.attribute("name");
    if(name.isEmpty())
        name = rootElement.nodeName();
    q->log(i18n("Importing reference \"%1\"").arg(name));
    for(QDomNode rowNode = rootElement.firstChild(); !rowNode.isNull();
rowNode = rowNode.nextSibling()) {
        if(rowNode.nodeName() != "row")
            continue;
        QMap<QString, QString> values;
        for(QDomNode node = rowNode.firstChild(); !node.isNull(); node =
node.nextSibling()) {
            if(node.isElement() && node.nodeName() == "field")
                values.insert(node.toElement().attribute("name"),
node.firstChild().nodeValue());
        }
        try {
            pqxx::work work(app().connection(),
KNSIADMIN_PQXX_IDENTIFIER);
            bool exists = false;
            if(!keyField.isEmpty()) {
                StringBuilder query;
                query << "select 1 from directories." << tableName << "
where " << keyField
                    << " = " << qQuoted(values[keyField]) << ' ';
                pqxx::result data = work.exec(toDb(query));
                exists = !data.empty();
            }
            StringBuilder query;
            if(exists) {
                query << "update directories." << tableName << " set";
                bool already = false;

```

```

        const QMap<QString, QString>::ConstIterator end =
values.constEnd();
        for(QMap<QString, QString>::ConstIterator it =
values.constBegin(); it != end; ++it) {
            if(it.key() != keyField) {
                if(already)
                    query << ',';
                query << ' ' << it.key() << " = " <<
qQuoted(it.data());
                already = true;
            }
        }
        query << " where " << keyField << " = " <<
qQuoted(values[keyField]) << ';';
    }
    else {
        query << "insert into directories." << tableName <<
"(";
        query.append(QStringList(values.keys()), ", ");
        query << ") values (";

        bool already = false;
        const QMap<QString, QString>::ConstIterator end =
values.constEnd();
        for(QMap<QString, QString>::ConstIterator it =
values.constBegin(); it != end; ++it) {
            if(already)
                query << ", ";
            query << qQuoted(it.data());
            already = true;
        }
        query << ");";
    }
    work.exec(toDb(query));
    work.commit();
}
catch(const pqxx::pqxx_exception &e) {
    q->error(StringBuilder("<pre>") <<
fromDb(e.base().what()).stripWhiteSpace() << "</pre>");
    log_exception() << e;
}
}
return true;
}
} // namespace actualization
} // namespace knsiadmin

```

ПРИЛОЖЕНИЕ Б. ЗАГРУЗКА СФОРМИРОВАННЫХ ПАКЕТОВ
АКТУАЛИЗАЦИИ

```

/*****
 *
 * Copyright (C) 2018 Rubin and/or its subsidiary(-ies).
 * Contact: http://www.npp-rubin.ru
 *
 * The program is provided AS IS with NO WARRANTY OF ANY KIND, INCLUDING THE
 * WARRANTY OF DESIGN, MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.
 *
 *****/

#ifndef KNSIADMIN_ACTUALIZATION_DOWNLOADDIALOG_H
#define KNSIADMIN_ACTUALIZATION_DOWNLOADDIALOG_H

#include <actualization/actualization_export.h>
#include <qwizard.h>

namespace knsiadmin {
namespace actualization {

class ACTUALIZATION_EXPORT DownloadDialog : public QWizard
{
    Q_OBJECT
public:
    DownloadDialog(QWidget *parent = 0, const char *name = 0);
    ~DownloadDialog();

    void setFileName(const QString &fileName);

public slots:
    bool configure();
    int run();
    void reject();

protected slots:
    void languageChange();

private slots:
    void startWork();

    void crossTotalSteps(long);
    void crossProgress(long);

private:
    class Private;
    Private *const d;
};

} // namespace actualization
} // namespace knsiadmin

#endif // KNSIADMIN_ACTUALIZATION_DOWNLOADDIALOG_H

*****/

#include "downloaddialog.h"

#include <actualization/log.h>

```

```

#include <core/application.h>
#include <core/progresswidget.h>
#include <core/i18n/locale.h>
#include <core/dialog/messagebox.h>
#include <core/savefile.h>
#include <core/helpers.h>
#include <core/helpers/fieldvalue.h>
#include <core/xmlstream.h>
#include <core/scopedoverridecursor.h>
#include <core/descendantsproxymodel.h>
#include <core/dialog/dialogbase.h>
#include <core/scoped_ptr.h>
#include <core/pathrequester.h>

#include <classifier/pasport/passport.h>
#include <classifier/pasport/struct.h>
#include <classifier/body/model.h>
#include <classifier/xmlexportvisitor.h>
#include <classifier/body/delayedexportmodel.h>
#include <classifier/xmlexportvisitor.h>
#include <classifier/pasport/treecolumnsconfigpage.h>
#include <classifier/pasport/property.h>

#include <nsi/passport/passport.h>
#include <nsi/passport/struct.h>
#include <nsi/body/model.h>
#include <nsi/xmlexportvisitor.h>
#include <nsi/body/listview.h>

#include <cross/zipexporter.h>

#include <pqxx/pqxx>

#include <quazip/quazip.h>
#include <quazip/quazipfile.h>

#include <qapplication.h>
#include <qfile.h>
#include <qtimer.h>
#include <qptrlist.h>
#include <qdatetime.h>
#include <qpushbutton.h>
#include <qlayout.h>
#include <qvbuttongroup.h>
#include <qcheckbox.h>
#include <qlabel.h>

#include <boost/scoped_ptr.hpp>
#include <boost/foreach.hpp>

namespace cp = knsiadmin::classifier::passport;
namespace cb = knsiadmin::classifier::body;
namespace np = knsiadmin::nsi::passport;
namespace nb = knsiadmin::nsi::body;

namespace knsiadmin {

```

```

namespace actualization {

static QString ZeroPaddedNumberString(uint number, int width)
{
    QString str;
    str.setNum(number);
    return str.rightJustify(width, '0');
}

static uint NumDigits(uint number)
{
    if(number == 0)
        return 1;
    uint num = 0;
    while(number > 0) {
        number /= 10;
        ++num;
    }
    return num;
}

static bool mkzipdir(QuaZip *zip, QString dirName)
{
    if(!dirName.endsWith("/"))
        dirName += '/';
    QuaZipFile dir(zip);
    bool ok = dir.open(IO_WriteOnly, QuaZipNewInfo(dirName, 0755, true));
    if(ok)
        dir.close();
    return ok;
}

static QString codeFromTableName(const QString &tableName)
{
    const QChar *ch = tableName.unicode() + tableName.length();
    const QChar *end = tableName.unicode();
    while(ch != end) {
        --ch;
        if(!ch->isDigit())
            break;
    }
    if(ch == (tableName.unicode() + tableName.length()))
        return QString();
    ++ch;
    uint len = tableName.unicode() + tableName.length() - ch;
    return QConstString(ch, len).string();
}

enum Content {
    DownloadDirectories = 0x01,
    DownloadClassifiers = 0x02,
    DownloadNsi          = 0x04,
    DownloadCross         = 0x08,
    DownloadAll           = DownloadDirectories | DownloadClassifiers |
DownloadNsi | DownloadCross
};

```

```

class DownloadDialog::Private
{
public:
    DownloadDialog *q;
    ProgressWidget *progressPage;
    bool cancelRequested;
    knsiadmin::XmlExportVisitor *visitor;
    long crossTotalSteps;
    long crossProgress;
    int content;

    QString baseDirName;
    boost::scoped_ptr<SaveFile> file;

    class FinishGuard {
    public:
        FinishGuard(Private *dd) : d(dd) {}
        ~FinishGuard()
        {
            if(d->cancelRequested)
                d->progressPage->log(i18n("Cancelled from user"));
            else
                d->progressPage->log(i18n("Finished successfully"));
        }
    private:
        Private *const d;
    };

    class ClassifierData {
    public:
        ClassifierData()
        {
        }
        ClassifierData(const cp::Passport &thePassport,
pqxx::transaction_base &work)
            : passport(thePassport)
        {
            if(!passport.sid.isEmpty())
                setStruct(cp::Struct::read(passport.sid, work));
        }

        cp::Passport passport;
        cp::Struct struc;

        QStringList dependencyTables;
        QList<ClassifierData> dependencyDatas;

        inline QString tableName() const { return passport.table_name; }

        void setStruct(const cp::Struct &data);

        void resolveDependencies(const QList<ClassifierData>
&dataList);

        template<typename Visitor>
        void accept(Visitor &visitor)

```

```

        {
            passport.accept(visitor);
            struc.accept(visitor);
        }
    };

class CrossExporter : public cross::ZipExporter {
public:
    CrossExporter(QuaZip *zip, DownloadDialog::Private *dd,
                  QObject *parent = 0, const char *name = 0);

protected:
    bool exportStarted() { return true; }
    bool exportData(const pqxx::result &data);
    bool exportData(const pqxx::result &data, QIODevice *device);
    bool exportFinished();

private:
    QFile m_file;
    QuaZip *m_zip;
    DownloadDialog::Private *const d;
};

CrossExporter *crossExporter;
QPtrList<ClassifierData> unknownCodeClassifiers;

QPtrList<ClassifierData> classifiersLoadQueue();
bool classifiersLoadQueue(ClassifierData *data,
QPtrList<ClassifierData> &queue,
                        QPtrList<ClassifierData>
&circularityCheckQueue) const;
QPtrList<ClassifierData> readClassifiers();
void downloadDirectory(const QString &table, const QStringList &fields,
const QString &keyField,
                        const QString &name, QuaZip *zip);

void downloadDirectories(QuaZip *zip);
void downloadClassifiers(QuaZip *zip, const QPtrList<ClassifierData>
&queue);
void downloadNsi(QuaZip *zip, const QValueList<nsi::passport::Passport>
&passList);
void downloadCross(QuaZip *zip);
};

DownloadDialog::DownloadDialog(QWidget *parent, const char *name)
: QWizard(parent, name)
, d(new Private)
{
    d->q = this;
    d->progressPage = new ProgressWidget;
    d->cancelRequested = false;
    d->visitor = 0;
    d->crossTotalSteps = 0;
    d->crossProgress = 0;
    d->crossExporter = 0;
    d->content = DownloadAll;
}

```

```

        addPage(d->progressPage, QString());

        setHelpEnabled(d->progressPage, false);
        setFinishEnabled(d->progressPage, false);

        languageChange();
    }

DownloadDialog::~DownloadDialog()
{
    if(d->file)
        d->file->abort();
    delete d;
}

void DownloadDialog::setFileName(const QString &fileName)
{
    if(d->file)
        d->file->abort();
    d->file.reset(new SaveFile(fileName));

    QFileInfo fi(fileName);
    d->baseDirName = fi.baseName(true) + '/';
}

bool DownloadDialog::configure()
{
    using namespace dialog;
    DialogBase *dlg = new DialogBase(DialogBase::Plain,
i18n("Settings..."),
                                DialogBase::Ok|DialogBase::Cancel,
DialogBase::Ok);
    QLabel *pathLabel = new QLabel(dlg->plainPage());
    PathRequester *pathRequester = new PathRequester(d->file->name(), dlg->plainPage());
    QVButtonGroup *buttonGroup = new QVButtonGroup(dlg->plainPage());
    QCheckBox *directoriesCheckBox = new QCheckBox(buttonGroup);
    QCheckBox *classifiersCheckBox = new QCheckBox(buttonGroup);
    QCheckBox *nsiCheckBox = new QCheckBox(buttonGroup);
    QCheckBox *crossCheckBox = new QCheckBox(buttonGroup);

    pathLabel->setText(i18n("File:"));
    pathRequester->setMode(PathRequester::File);
    pathRequester->setFilter(i18n("ZIP-archive (*.zip)"));
    buttonGroup->setTitle(i18n("Package Content"));
    directoriesCheckBox->setText(i18n("Directories"));
    directoriesCheckBox->setChecked(true);
    classifiersCheckBox->setText(i18n("Classifiers"));
    classifiersCheckBox->setChecked(true);
    nsiCheckBox->setText(i18n("NSI"));
    nsiCheckBox->setChecked(true);
    crossCheckBox->setText(i18n("Cross"));
    crossCheckBox->setChecked(true);

    QVBoxLayout *layout = new QVBoxLayout(dlg->plainPage(), 11, 6);
    QHBoxLayout *pathLayout = new QHBoxLayout;
    pathLayout->addWidget(pathLabel);

```

```

pathLayout->addWidget(pathRequester);

layout->addLayout(pathLayout);
layout->addWidget(buttonGroup);
layout->addItem(new QSpacerItem(0, 0, QSizePolicy::Minimum,
QSizePolicy::MinimumExpanding));

    QSize initialSize = dlg-
>configDialogSize("Actualization/ConfigDialog");
    dlg->resize(initialSize.expandedTo(QApplication::globalStrut()));

    bool ok = (dlg->exec() == QDialog::Accepted);
    if(ok) {
        if(pathRequester->path() != d->file->name())
            setFileName(pathRequester->path());
        int content = 0;
        if(directoriesCheckBox->isChecked())
            content |= DownloadDirectories;
        if(classifiersCheckBox->isChecked())
            content |= DownloadClassifiers;
        if(nsiCheckBox->isChecked())
            content |= DownloadNsi;
        if(crossCheckBox->isChecked())
            content |= DownloadCross;
        d->content = content ? content : DownloadAll;
    }

    dlg->saveDialogSize("Actualization/ConfigDialog");

    delete dlg;
    return ok;
}

int DownloadDialog::run()
{
    QTimer::singleShot(0, this, SLOT(startWork()));
    return exec();
}

void DownloadDialog::reject()
{
    if(d->progressPage->progress() < d->progressPage->totalSteps()) {
        const int answer =
            dialog::MessageBox::questionYesNo(this,
                                                i18n("Downloading is not
finished. "
                                                "Are you sure you want
quit?"));
        if(answer == dialog::MessageBox::Yes) {
            d->cancelRequested = true;
            cancelButton()->setEnabled(false);
            if(d->visitor)
                d->visitor->cancel();
            if(d->crossExporter)
                d->crossExporter->cancel();
        }
        return;
    }
}

```

```

    }
    QWizard::reject();
}

void DownloadDialog::languageChange()
{
    setCaption(i18n("Downloading..."));
    setTitle(d->progressPage, i18n("Actualization File Downloading..."));
}

void DownloadDialog::startWork()
{
    if(d->file->status()) {
        d->progressPage->error(i18n("Error openning file: %1").arg(d->file->name()));
        setFinishEnabled(d->progressPage, true);
        return;
    }

    ScopedOverrideCursor SOC(Qt::waitCursor);

    QuaZip zip(d->file->file());
    if(!zip.open(QuaZip::mdCreate)) {
        log_warning() << "failed open zip";
        return;
    }

    mkzipdir(&zip, d->baseDirName);

    QPtrList<Private::ClassifierData> clfQueue;
    if(d->content & DownloadClassifiers)
        clfQueue = d->classifiersLoadQueue();
    clfQueue.setAutoDelete(true);

    if(!d->unknownCodeClassifiers.isEmpty()) {
        QValueList<QStringList> table;
        QStringList head;
        QPtrList<cp::Property> properties;
        properties.setAutoDelete(true);
        BOOST_FOREACH(int fieldType, cp::TreeColumnsConfigPage::columns())
        {
            cp::Property::Type propertyType =
static_cast<cp::Property::Type>(fieldType);
            cp::Property *prop = cp::Property::create(propertyType);
            head.append(prop->name());
            properties.append(prop);
        }
        table.append(head);
        QPtrListIterator<Private::ClassifierData> itData(d->unknownCodeClassifiers);
        for(; itData.current(); ++itData) {
            QStringList line;
            for(QPtrListIterator<cp::Property> itProp(properties);
itProp.current(); ++itProp) {
                QString value = itProp.current()->valueFrom(itData.current()->passport);
                line.append(value);
            }
        }
    }
}

```

```

        }
        table.append(line);
    }
    properties.clear();

    SOC.restore();
    int answer =
        dialog::MessageBox::warningContinueCancelList(
            this,
            i18n("%1 classifier does not have a code",
                "%1 classifiers does not have a code",
                d->unknownCodeClassifiers.count()),
            table);
    if(answer == dialog::MessageBox::Cancel) {
        zip.close();
        d->file->abort();
        d->file.reset();
        d->progressPage->setTotalSteps(1);
        d->progressPage->setProgress(1);
        d->progressPage->error(i18n("Cancelled from user"));
        setFinishEnabled(d->progressPage, true);
        return;
    }
    SOC.reset(Qt::waitCursor);
}

Private::FinishGuard finishGuard(d);

QValueList<np::Passport> nsiList;
if(d->content & DownloadNsi) {
    try {
        pqxx::read_transaction work(app().connection(),
KNSIADMIN_PQXX_IDENTIFIER);
        nsiList = np::Passport::readAll(work);
    }
    catch(const pqxx::pqxx_exception &e) {
        log_exception() << e;
    }
}

long totalSteps = clfQueue.count() + nsiList.count() + 1;
d->progressPage->setTotalSteps(totalSteps);
d->progressPage->setProgress(0);

if(d->content & DownloadDirectories) {
    d->downloadDirectories(&zip);
    if(d->cancelRequested) {
        zip.close();
        d->file->abort();
        d->file.reset();
        d->progressPage->setProgress(totalSteps);
        setFinishEnabled(d->progressPage, true);
        return;
    }
}

if(d->content & DownloadClassifiers) {

```

```

        d->downloadClassifiers(&zip, clfQueue);
        if(d->cancelRequested) {
            zip.close();
            d->file->abort();
            d->file.reset();
            d->progressPage->setProgress(totalSteps);
            setFinishEnabled(d->progressPage, true);
            return;
        }
    }

    if(d->content & DownloadNsi) {
        d->downloadNsi(&zip, nsiList);
        if(d->cancelRequested) {
            zip.close();
            d->file->abort();
            d->file.reset();
            d->progressPage->setProgress(totalSteps);
            setFinishEnabled(d->progressPage, true);
            return;
        }
    }

    if(d->content & DownloadCross) {
        d->downloadCross(&zip);
        if(d->cancelRequested) {
            zip.close();
            d->file->abort();
            d->file.reset();
            d->progressPage->setProgress(totalSteps);
            setFinishEnabled(d->progressPage, true);
            return;
        }
    }

    zip.close();

    d->file->close();
    d->file.reset();

    d->progressPage->setProgress(totalSteps + d->crossTotalSteps);

    setFinishEnabled(d->progressPage, true);
}

void DownloadDialog::crossTotalSteps(long steps)
{
    d->crossTotalSteps = steps;
    d->progressPage->setTotalSteps(d->progressPage->totalSteps() + steps);
}

void DownloadDialog::crossProgress(long value)
{
    long progress = value - d->crossProgress;
    // double progress = (value - d->crossProgress) * kCrossTotalSteps /
    double(d->crossTotalSteps);
    // d->progressPage->advanceProgress(qRound(progress));
}

```

```

        d->progressPage->advanceProgress(progress);
        d->crossProgress = value;
    }

QPtrList<DownloadDialog::Private::ClassifierData>
DownloadDialog::Private::classifiersLoadQueue()
{
    QPtrList<ClassifierData> queue;
    QPtrList<ClassifierData> list = readClassifiers();
    for(ClassifierData * data = list.first(); data; data = list.next()) {
        QPtrList<ClassifierData> circularityCheckQueue;
        classifiersLoadQueue(data, queue, circularityCheckQueue);
        if(cancelRequested) {
            queue.setAutoDelete(true);
            return QPtrList<ClassifierData>();
        }
    }
    return queue;
}

bool DownloadDialog::Private::classifiersLoadQueue(
    DownloadDialog::Private::ClassifierData *data,
    QPtrList<DownloadDialog::Private::ClassifierData> &queue,
    QPtrList<DownloadDialog::Private::ClassifierData>
&circularityCheckQueue) const
{
    if(queue.containsRef(data)) {
        return true;
    }
    if(circularityCheckQueue.containsRef(data)) {
        log_warning() << "Circular dependency detected:";
        circularityCheckQueue.findRef(data);
        for(; circularityCheckQueue.current();
circularityCheckQueue.next()) {
            ClassifierData *dep = circularityCheckQueue.current();
            log_warning() << dep->tableName() << " depends on";
        }
        log_warning() << data->tableName();
        return false;
    }
    circularityCheckQueue.append(data);
    for(QPtrListIterator<ClassifierData> it(data->dependencyDatas);
it.current(); ++it) {
        ClassifierData *dependency = it.current();
        if(!classifiersLoadQueue(dependency, queue,
circularityCheckQueue)) {
            return false;
        }
    }
    queue.append(data);
    return true;
}

QPtrList<DownloadDialog::Private::ClassifierData>
DownloadDialog::Private::readClassifiers()
{

```

```

QPtrList<ClassifierData> list;
unknownCodeClassifiers.clear();

try {
    pqxx::read_transaction work(app().connection(),
KNSIADMIN_PQXX_IDENTIFIER);
    BOOST_FOREACH(const cp::Passport &pass,
cp::Passport::readAll(work)) {
        ClassifierData *data = new ClassifierData(pass, work);
        list.append(data);
        if(pass.code.isEmpty() &&
codeFromTableName(pass.table_name).isEmpty())
            unknownCodeClassifiers.append(data);
    }
} catch(const pqxx::pqxx_exception &e) {
    log_exception() << e;
}
for(ClassifierData *data = list.first(); data; data = list.next()) {
    data->resolveDependencies(list);
}
return list;
}

void DownloadDialog::Private::downloadDirectory(const QString &table, const
QStringList &fields,
                                                const QString &keyField,
const QString &name, QuaZip *zip)
{
    progressPage->log(i18n("Downloading reference \"%1\"").arg(name));

    QuaZipFile zipFile(zip);
    zipFile.open(IO_WriteOnly, QuaZipNewInfo(baseDirName + "directories/" +
table + ".xml", 0644));

    XmlStreamWriter xml(&zipFile);
    xml.setAutoFormatting(true);
    xml.setAutoFormattingIndent(-1);

    xml.writeStartDocument();
    xml.writeStartElement(table);
    if(!keyField.isEmpty())
        xml.writeAttribute("key", keyField);
    xml.writeAttribute("name", name);

    bool showLog = false;
    QTime timer;
    timer.start();

    pqxx::result data;

    try {
        pqxx::read_transaction work(app().connection(),
KNSIADMIN_PQXX_IDENTIFIER);
        QString query = "select " + fields.join(",") + " from
directories." + table;
        data = work.exec(toDb(query));
    }
}

```

```

catch(const pqxx::pqxx_exception &e) {
    log_exception() << e;
}

const pqxx::result::size_type rows = data.size();
for(pqxx::result::size_type row = 0; row < rows; ++row) {
    xml.writeStartElement("row");
    BOOST_FOREACH(pqxx::result::field col, data[row]) {
        QString value = fromDb(col).stripWhiteSpace();

        if(value.isEmpty())
            xml.writeEmptyElement("field");
        else
            xml.writeStartElement("field");

        xml.writeAttribute("name", fromDb(col.name()));

        if(!value.isEmpty()) {
            xml.writeCharacters(value);
            xml.writeEndElement();
        }

        if(timer.elapsed() >= 150) {
            const QString message = i18n("Downloading data %1%
(%2 / %3) ...");

            .arg(row*100/rows).arg(row).arg(rows);
            if(showLog) {
                progressPage->replaceLastMessage(message);
            }
            else {
                showLog = true;
                progressPage->log(message);
            }
            timer.restart();
        }
    }
    xml.writeEndElement();
}

if(showLog)
    progressPage->removeLastMessage();

xml.writeEndElement();
xml.writeEndDocument();

zipFile.close();
}

void DownloadDialog::Private::downloadDirectories(QuaZip *zip)
{
    const QString defaultKeyField = "sid";
    const QStringList defaultFields = QStringList() << defaultKeyField <<
"shortname" << "name";

    mkzipdir(zip, baseDirName + "directories");
}

```

```

downloadDirectory("category", defaultFields, defaultKeyField,
    il8n("List of types of classifiers"), zip);

downloadDirectory("classification", defaultFields, defaultKeyField,
    il8n("List of systems of classification"), zip);

downloadDirectory("coding", defaultFields, defaultKeyField, il8n("List
of systems of coding"),
    zip);

downloadDirectory("grif", defaultFields, defaultKeyField,
    il8n("List of security classifications"), zip);

downloadDirectory("organisation", QStringList() << "inn" << "name" <<
"mailing_address"
                                << "contact_person" <<
"contact_phone",
    "inn", il8n("List of the organizations"), zip);
}

void DownloadDialog::Private::downloadClassifiers(QuaZip *zip, const
QPtrList<ClassifierData> &queue)
{
    progressPage->log(il8n("Reading classifiers"));

    mkzipdir(zip, baseDirName + "classifier");

    if(!queue.isEmpty()) {
        int progress = 0;
        uint index = 0;
        QTime time;
        time.start();
        const uint numFieldWidth = NumDigits(queue.count()+1);

        for(QPtrListIterator<ClassifierData> it(queue); it.current(); +
+it) {
            ClassifierData *data = it.current();
            QString name = data->passport.shortname;
            if(name.isEmpty())
                name = data->passport.name;
            if(name.isEmpty())
                name = data->passport.table_name;
            progressPage->log(il8n("Downloading
classifier \"%1\"").arg(name));
            QString zipFileName = data->passport.code;
            if(zipFileName.isEmpty())
                zipFileName = data->passport.table_name;
            zipFileName = QString(baseDirName + "classifier/"
%1.%2.xml")
                                .arg(ZeroPaddedNumberString(++index,
numFieldWidth),
                                zipFileName);
            QuaZipFile zipFile(zip);
            zipFile.open(IO_WriteOnly, QuaZipNewInfo(zipFileName,
0644));

            classifier::XmlExportVisitor visitor;

```

```

        visitor.setAutoFormatting(true);
        visitor.setAutoFormattingIndent(-1);
        visitor.setDevice(&zipFile);
        visitor.setProgressWidget(progressPage);
        visitor.open();

        this->visitor = &visitor;

        data->accept(visitor);
        if(cancelRequested)
            return;

        if(data->passport.is_table) {
            cb::DelayedExportModel model;
            model.setStruct(data->struc);
            model.setSid(data->passport.sid);
            model.accept(visitor);
        }

        ++progress;
        if(time.elapsed() >= 150) {
            progressPage->advanceProgress(progress);
            progress = 0;
            time.restart();
        }
        if(cancelRequested)
            return;

        this->visitor = 0;

        visitor.close();
        visitor.close();

        zipFile.close();
    }

    progressPage->advanceProgress(progress);
}

void DownloadDialog::Private::downloadNsi(QuaZip *zip, const
QValueList<np::Passport> &passList)
{
    mkzipdir(zip, baseDirName + "nsi");
    int progress = 0;

    QTime time;
    time.start();

    BOOST_FOREACH(np::Passport passport, passList) {
        QString name = passport.shortname;
        if(name.isEmpty())
            name = passport.name;
        if(name.isEmpty())
            name = passport.table_name;
        progressPage->log(i18n("Downloading NSI \"%1\"").arg(name));
        QuaZipFile zipFile(zip);
    }
}

```

```

        QuaZipNewInfo zipFileInfo(QString(baseDirName + "nsi/
%1.xml").arg(passport.table_name), 0644);
        zipFile.open(IO_WriteOnly, zipFileInfo);

        nsi::XmlExportVisitor visitor;
        visitor.setAutoFormatting(true);
        visitor.setAutoFormattingIndent(-1);
        visitor.setDevice(&zipFile);
        visitor.setProgressWidget(progressPage);
        visitor.setModelRole(nb::Model::RefCodeRole);
        visitor.open();

        this->visitor = &visitor;

        passport.accept(visitor);

        np::Struct struc;
        try {
            pqxx::read_transaction work(app().connection(),
KNSIADMIN_PQXX_IDENTIFIER);
            struc = np::Struct::read(passport.sid, work);
        }
        catch(const pqxx::pqxx_exception &e) {
            log_exception() << e;
        }

        struc.accept(visitor);
        if(cancelRequested)
            return;

        if(passport.is_table) {
            boost::scoped_ptr<nb::ListView> view(new nb::ListView);
            view->setStruct(struc);

            try {
                pqxx::read_transaction work(app().connection(),
KNSIADMIN_PQXX_IDENTIFIER);
                pqxx::result data = work.exec(toDb(QString("select *
from metansi.get_terms('%1');")
.arg(passport.sid)));
                pqxx::icursorstream stream(work, data[0][0]);
                view->setTerminStream(stream, 0);
            }
            catch(const pqxx::pqxx_exception &e) {
                log_exception() << e;
            }

            boost::scoped_ptr<AbstractItemModel> model(view-
>createModel());
            boost::scoped_ptr<DescendantsProxyModel> proxy(new
DescendantsProxyModel);
            proxy->setSourceModel(model.get());

            proxy->accept(visitor);
        }

```

```

        this->visitor = 0;

        visitor.close();
        visitor.close();

        zipFile.close();

        ++progress;
        if(time.elapsed() >= 150) {
            progressPage->advanceProgress(progress);
            progress = 0;
            time.restart();
        }
        if(cancelRequested)
            return;
    }

    progressPage->advanceProgress(progress);
}

void DownloadDialog::Private::downloadCross(QuaZip *zip)
{
    mkzipdir(zip, baseDirName + "cross");
    crossTotalSteps = 0;
    crossProgress = 0;
    boost::scoped_ptr<CrossExporter> exporter(new CrossExporter(zip,
this));
    crossExporter = boost::get_pointer(exporter);
    QObject::connect(crossExporter, SIGNAL(totalSteps(long)), q,
SLOT(crossTotalSteps(long)));
    QObject::connect(crossExporter, SIGNAL(progress(long)), q,
SLOT(crossProgress(long)));
    exporter->start();
    crossExporter = 0;
}

void DownloadDialog::Private::ClassifierData::setStruct(const
classificator::passport::Struct &data)
{
    struc = data;

    BOOST_FOREACH(const cp::Field &field, data) {
        if(field.dataType == cp::Field::Reference) {
            QString dependency = field.fasetClf;
            if(dependencyTables.findIndex(dependency) == -1)
                dependencyTables.append(dependency);
        }
    }
}

void DownloadDialog::Private::ClassifierData::resolveDependencies(
    const QPtrList<DownloadDialog::Private::ClassifierData> &dataList)
{
    QPtrList<ClassifierData> resolvedDependencies;
    BOOST_FOREACH(const QString &table, dependencyTables) {

```

```

        if(table == passport.sid) {
            continue;
        }
        ClassifierData *found = 0;
        for(QPtrListIterator<ClassifierData> it(dataList); it.current(); +
+it) {
            ClassifierData * data = it.current();
            if(data->passport.sid == table) {
                found = data;
                break;
            }
        }
        if(!found) {
            log_warning() << "file for classifier with table sid \"" <<
table
                                << "\" not found";
            continue;
        }
        resolvedDependencies.append(found);
    }
    dependencyDatas = resolvedDependencies;
}

```

```

DownloadDialog::Private::CrossExporter::CrossExporter(QuaZip *zip,
DownloadDialog::Private *dd,
                                                    QObject *parent, const
char *name)
: cross::ZipExporter(parent, name)
, m_zip(zip)
, d(dd)
{
    setDevice(&m_file);
}

```

```

bool DownloadDialog::Private::CrossExporter::exportData(const pqxx::result
&data)
{
    QuaZipFile file(m_zip);
    file.open(IO_WriteOnly, QuaZipNewInfo(d->baseDirName + "cross/" +
zipFileName(), 0644));

    bool result = exportData(data, &file);

    file.close();

    return result;
}

```

```

bool DownloadDialog::Private::CrossExporter::exportData(const pqxx::result
&data, QIODevice *device)
{
    d->progressPage->log(i18n("Downloading cross \"%1\"
-> \"%2\"").arg(leftPassport().shortname,
rightPassport().shortname));
}

```

```

        return cross::ZipExporter::exportData(data, device);
    }

bool DownloadDialog::Private::CrossExporter::exportFinished()
{
    d->progressPage->advanceProgress(d->crossTotalSteps - d->crossProgress);
    return true;
}

} // namespace actualization
} // namespace knsiadmin

```

ПРИЛОЖЕНИЕ В. СОЗДАНИЕ ФУНКЦИЙ БАЗЫ ДАННЫХ

1. Создание нового паспорта:

```
CREATE OR REPLACE FUNCTION metadata.new_passport (  
    text,  
    varchar,  
    text,  
    integer,  
    integer,  
    integer,  
    varchar,  
    integer,  
    integer,  
    integer,  
    timestamp,  
    integer,  
    timestamp,  
    text,  
    timestamp,  
    text,  
    timestamp,  
    text,  
    integer  
)  
RETURNS char AS  
$body$  
declare  
  
    i_name                alias for $1; -- Наименование  
    i_shortcode           alias for $2; -- Сокращенное наименование  
    i_table_name          alias for $3; -- Наименование таблицы базы данных  
    i_directory_sid       alias for $4; -- Директория  
    i_coding_sid          alias for $5; -- Система кодирования  
    i_classification_sid  alias for $6; -- Классификация  
    i_okok_code           alias for $7; -- Код по ОКОК  
    i_category_sid        alias for $8; -- Категория  
    i_grif_sid            alias for $9; -- Гриф секретности  
    i_conduct_org_sid     alias for $10; -- Организация ответственная за  
ведение  
    i_date_statement      alias for $11; -- Дата утверждения
```

```

        i_develop_org_sid      alias for $12; -- Организация ответственная за
разработку
        i_date_receive         alias for $13; -- Дата получения
        i_reason_receive       alias for $14; -- Основание для получения
        i_date_change          alias for $15; -- Дата изменения
        i_reason_change        alias for $16; -- Основание для изменения
        i_date_enter           alias for $17; -- Дата ввода в действие
        i_remark               alias for $18; -- Примечание
        i_statement_org_sid    alias for $19; -- Организация ответственная за
утверждение

        result char(32);

        clfnum text;

begin

    -- проверка на наименование таблицы
        if (exists(select * from metadata.get_passports() where table_name =
i_table_name) and (i_table_name is null)) then
            return -1;
        end if;

    -- проверка на наименование
        if (exists(select * from metadata.get_passports() where p_name =
i_name) and (i_name is null)) then
            return -2;
        end if;

    -- проверка на сокращенное наименование
        if (exists(select * from metadata.get_passports() where p_shortcode =
i_shortcode) and (i_shortcode is null)) then
            return -3;
        end if;

    -- проверка на гриф секретности
        if (not exists(select * from directories.grif where sid = i_grif_sid)
and (i_grif_sid is null)) then

```

```

        return -4;
    end if;

    if exists(select 1 from directories.knsi_view where kview=1)
    then
        if (i_okok_code isnull) then
            clfnum:=split_part(i_table_name,'cl_',2);
            if (clfnum isnull) then
                raise EXCEPTION 'Поле "Код" не заполнено, либо "Наименование
таблицы" должно быть вида "cl_111111"!';
            end if;
        else
            clfnum:=i_okok_code;
        end if;

        perform
public.classif_new_pasport(clfnum::integer,i_name,i_shortcode,0::int2,0::int2
,i_remark,current_date::text,0::int2,i_grif_sid,current_date,0);
    else
        insert into
metadata.passport(p_name,p_shortcode,table_name,directory_sid,coding_sid,clas
sification_sid,okok_code,category_sid,grif_sid,
conduct_org_sid,date_statement,develop_org_
sid,date_receive,reason_receive,date_change,reason_change,date_enter,
remark,statement_org_sid)
values(i_name,i_shortcode,i_table_name,i_directory_sid,i_coding_sid,i_c
lassification_sid,i_okok_code,i_category_sid,i_grif_sid,
i_conduct_org_sid,i_date_statement,i_develo
p_org_sid,i_date_receive,i_reason_receive,i_date_change,i_reason_change,i_dat
e_enter,
i_remark,i_statement_org_sid);
    end if;

    select * into result from metadata.hash(i_table_name||'.'||
i_name||'.'||i_shortcode);

return result;
end;
$body$

```

```

LANGUAGE 'plpgsql'
VOLATILE
CALLED ON NULL INPUT
SECURITY INVOKER
COST 100.

```

2. Создание структуры классификатора:

```

CREATE OR REPLACE FUNCTION metadata.new_struct (
    char,
    varchar,
    integer,
    integer,
    text,
    integer,
    integer,
    char,
    text,
    integer,
    integer,
    text,
    integer,
    integer,
    text,
    text,
    integer,
    integer
)
RETURNS integer AS
$body$
declare
    i_passport_sid    alias for $1; -- паспорт классификатора
    i_field_name      alias for $2; -- наименование поля
    i_field_size      alias for $3; -- размер поля
    i_field_num       alias for $4; -- порядковый номер поля
    i_remark          alias for $5; -- комментарий
    i_field_type_sid  alias for $6; -- тип поля
    i_is_key          alias for $7; -- является ли поле ключевым
    i_faset_sid       alias for $8; -- фасетный классификатор
    i_faset_fields    alias for $9; -- список полей фасетного классификатора
    i_field_datatype  alias for $10; -- тип данных поля
    i_is_control      alias for $11; -- является ли поле контрольным числом
    i_field_struct    alias for $12; -- структура поля
    i_is_sostav       alias for $13; -- является ли поле составным
    i_field_gen_code  alias for $14; -- алгоритм генерации поля
    i_field_mask      alias for $15; -- маска ввода
    i_field_caption   alias for $16; -- заголовок поля
    i_is_notempty     alias for $17; -- поле не пустое
    i_is_unique       alias for $18; -- поле уникальное

    l_field_num       INTEGER;      -- порядковый номер поля
    m_field_num       INTEGER;      -- максимальный порядковый номер поля

    l_field_sid       INTEGER;
    l_table_name      TEXT;

```

```

l_field_type INTEGER;
l_field_dbname TEXT;
l_remark text;

result      INTEGER;

row1 record;

begin

    select table_name into l_table_name from metadata.get_passports() where
sid = i_passport_sid;

    -- если паспорт не определен или такого паспорта не существует
    if (not Found)
    then
        return -2;
    end if;

    -- если тип поля не определен или такого типа поля не существует
    if (i_field_type_sid ISNULL) or not exists(select * from
metadata.field_type where sid = i_field_type_sid)
    then
        return -3;
    end if;

    -- поле с таким именем уже существует
    if exists(select 1 from metadata.get_struct(i_passport_sid) where
field_name = i_field_name)
    then
        raise notice 'Поле с таким именем уже существует';
        return -4;
    end if;

    -- фасетного классификатора не существует
    if (not exists(select * from metadata.get_passports() where sid =
i_faset_sid) and (i_faset_sid is not null))
    then
        return -5;
    end if;

    -- находим максимальный номер поля
    begin
        select max(field_num) into m_field_num from
metadata.get_struct(i_passport_sid);
    end;

    -- если номер поля не определен или больше чем последний номер то текущий
номер на единицу больше
    if ((i_field_num > m_field_num) or (i_field_num ISNULL))
    then
        l_field_num := coalesce(m_field_num,0) + 1;
    else
        l_field_num := i_field_num;
    end if;

```

```

-- перенумерация полей если номер поля меньше или равен существующим
for row1 in
    select sid, field_num + 1 as field_num
    from metadata.get_struct(i_passport_sid)
    where field_num >= l_field_num
    order by field_num desc
loop
    if exists(select 1 from directories.knsi_view where kview=1)
    then
        update public.classificator_struct set field_num = row1.field_num
        where sid = row1.sid;
    else
        update metadata.struct set field_num = row1.field_num where sid =
row1.sid;
    end if;
end loop;

if exists(select 1 from directories.knsi_view where kview=1) then
    l_remark:=i_remark;

    if i_field_datatype=6 then
        l_field_type:=3;
        l_field_dbname:='faset';

        select okok_code from metadata.get_passport(i_faset_sid) into
l_remark;

    elsif i_is_key=1 then
        l_field_type:=0;
        l_field_dbname:='code';
    elsif i_field_type_sid=0 and i_field_datatype<>6
    then
        if i_is_notempty=1
        then
            if not exists(select 1 from public.classif_get_struct((select
okok_code from metadata.get_passport(i_passport_sid))::integer)
            where upper(field_dbname)='NAME')
            then
                l_field_type:=1;
                l_field_dbname:='name';
            elsif not exists(select 1 from public.classif_get_struct((select
okok_code from metadata.get_passport(i_passport_sid))::integer)
            where upper(field_dbname)='SHORTNAME')
            then
                l_field_type:=2;
                l_field_dbname:='shortname';
            else
                l_field_type:=4;
                l_field_dbname:='addname';
            end if;
        else
            if not exists(select 1 from public.classif_get_struct((select
okok_code from metadata.get_passport(i_passport_sid))::integer)
            where upper(field_dbname)='SHORTNAME')
            then
                l_field_type:=2;
                l_field_dbname:='shortname';

```

```

        else
            l_field_type:=4;
            l_field_dbname:='addname';
        end if;
    end if;
    -----
elseif i_field_type_sid=1 and i_field_datatype<>6
then
    if i_is_notempty=1
    then
        if not exists(select 1 from public.classif_get_struct((select
okok_code from metadata.get_passport(i_passport_sid))::integer)
            where upper(field_dbname)='NAME')
        then
            l_field_type:=1;
            l_field_dbname:='name';
        elsif not exists(select 1 from public.classif_get_struct((select
okok_code from metadata.get_passport(i_passport_sid))::integer)
            where upper(field_dbname)='SHORTNAME')
        then
            l_field_type:=2;
            l_field_dbname:='shortname';
        else
            l_field_type:=4;
            l_field_dbname:='addname';
        end if;
    else
        if not exists(select 1 from public.classif_get_struct((select
okok_code from metadata.get_passport(i_passport_sid))::integer)
            where upper(field_dbname)='SHORTNAME')
        then
            l_field_type:=2;
            l_field_dbname:='shortname';
        else
            l_field_type:=4;
            l_field_dbname:='addname';
        end if;
    end if;
    -----
else
    l_field_type:=4;
    l_field_dbname:='addname';
end if;

    if l_field_dbname in ('addname','faset') then
        select upper(l_field_dbname) || (coalesce((select
max(field_array_num)
from public.classif_get_struct((select okok_code from
metadata.get_passport(i_passport_sid))::integer)
            where upper(field_dbname) like
upper(l_field_dbname||'%')),0)+1)::text
            into l_field_dbname;
    end if;

    perform public.classif_new_struct((select okok_code from
metadata.get_passport(i_passport_sid))::integer,

```

```

i_field_caption,i_field_size,l_field_num,l_remark,l_field_type,upper(l_field_
dbname));
    else
        insert into
metadata.struct(passport_sid,field_name,field_size,field_num,remark,field_typ
e_sid,is_key,faset_sid,faset_fields,

field_datatype_sid,is_control,field_struct,is_sostav,field_gen_code,field_mas
k,field_caption,

                                is_notempty,is_unique)

values(i_passport_sid,i_field_name,i_field_size,l_field_num,i_remark,i_field_
type_sid,i_is_key,i_faset_sid,i_faset_fields,

i_field_datatype,i_is_control,i_field_struct,i_is_sostav,i_field_gen_code,i_f
ield_mask,i_field_caption,

                                i_is_notempty,i_is_unique);

    end if;

    -- сид новой записи
    select sid into l_field_sid from metadata.get_struct(i_passport_sid)
where field_name = i_field_name;

    result := 0;

    if exists(select 1 from directories.knsi_view where kview=1) then
        perform metadata.create_view_nkilo((select okok_code from
metadata.get_passport(i_passport_sid)));
    else
        -- если таблица классификатора существует
        if (metadata.qualifier_table_exists(l_table_name))
        then
            -- добавляем поле в таблицу

            select * into result from metadata.add_field(l_field_sid);
        end if;
    end if;

    -- если поле в таблицу добавлено успешно или не добавлялось вообще
    if (result = 0)
    then
        -- возвращаем сид новой записи
        return l_field_sid;
    else
        -- ошибка добавления поля в таблицу
        raise notice 'ошибка добавления поля в таблицу';
        return result - 100;
    end if;

end;
$body$
LANGUAGE 'plpgsql'
VOLATILE
CALLED ON NULL INPUT
SECURITY INVOKER
COST 100.

```

3. Создание термина классификатора:

```
CREATE OR REPLACE FUNCTION metadata.new_termin (
    char,
    text []
)
RETURNS char AS
$body$
declare
    i_passport_sid  alias for $1;  -- паспорт классификатора
    i_values         alias for $2;  -- значения

    _TABLE_NAME text;

    QUERY text;          -- запрос
    QUERY_FIELDS text;    -- список полей
    QUERY_VALUES text;    -- список значений

    SCHEMA_NAME text;

    WHERE_SQL text;

    row1 record;
    row2 record;

    result char(32);
    len integer;

begin

    select table_name into _TABLE_NAME from metadata.get_passports() where sid
= i_passport_sid;

    if (not metadata.qualifier_table_exists(_TABLE_NAME))
    then
        return '-1';
    end if;

    -- пустой массив значений
    if (i_values isnull)
    then
        return '-2';
    end if;

    -- если для пользователя определена роль допуска к части содержимого клас-
сификатора
    -- берем его схему
    select g.s_name
    into SCHEMA_NAME
    from metadata.users u,
        metadata.groups g
    where u.u_login = CURRENT_USER
        and u.group_sid = g.sid
        and metadata.isuserinrole(_TABLE_NAME, 'part'::text);

    -- иначе схему классификатора
```

```

if (not Found)
then
    SCHEMA_NAME = 'qualifiers';
end if;

WHERE_SQL := 'where 1=1 ';

for row1 in
    select s.field_name,
           (i_values)[s.field_num] as field_value,
           -- если поле символьное, дата или время, то добавляем кавычки в
условие
           case when s.field_datatype_sid in (1,4,5,6,7) then TRUE else FALSE
end as flag,
           s.field_num,
           s.is_key,
           s.field_datatype_sid,
           s.faset_sid,
           p.table_name as faset_table_name,
           s.field_size
    from metadata.get_struct(i_passport_sid) s
         left join metadata.get_passports() p on p.sid = s.faset_sid
    order by s.field_num
loop

    -- проверка на ссылочный термин
    if ((row1.faset_sid is not null) and (trim(row1.field_value) <> ''))
    then
        for row2 in
            EXECUTE 'select sid from "' || SCHEMA_NAME || '"."' ||
quote_ident(row1.faset_table_name) || '" where sid = ' ||
quote_literal(row1.field_value) || ';'
        loop
        end loop;

        if (not Found)
        then
            return '-3';
        end if;
    end if;

    -- проверка на размерность поля (только символьное)
    if ((row1.field_datatype_sid = 1) and (row1.field_size > 0) and
(length(row1.field_value) > row1.field_size))
    then
        len := octet_length(row1.field_value);
        raise exception 'Поле N% '%'' имеет размер % символов, а значение
'%'' - % символов', row1.field_num, row1.field_name, row1.field_size,
row1.field_value, len;
        return '-4';
    end if;

    if (row1.field_num = 1)
    then
        QUERY_FIELDS := ' "' || row1.field_name || '"';

        if (row1.flag)

```

```

then
    if ((row1.field_datatype_sid = 6) and (row1.field_value = ''))
    then
        QUERY_VALUES := 'NULL';
    elsif ((row1.field_datatype_sid = 6) and (row1.field_value <>
''))
    then
        QUERY_VALUES := quote_literal(row1.field_value);
    else
        QUERY_VALUES := quote_literal(row1.field_value);
    end if;
else
    QUERY_VALUES :=
coalesce(quote_literal(nullif(row1.field_value, '')), 'NULL');
end if;

else
    QUERY_FIELDS := QUERY_FIELDS || ' , "' || row1.field_name || '"';

    if (row1.flag)
    then
        if ((row1.field_datatype_sid = 6) and (row1.field_value = ''))
        then
            QUERY_VALUES := QUERY_VALUES || ', NULL';
        elsif (((row1.field_datatype_sid = 6) and (row1.field_value <>
''))))
        then
            QUERY_VALUES := QUERY_VALUES || ', ' ||
quote_literal(row1.field_value);
        else
            QUERY_VALUES := QUERY_VALUES || ', ' ||
quote_literal(row1.field_value);
        end if;
    else
        QUERY_VALUES := QUERY_VALUES || ', ' ||
coalesce(quote_literal(nullif(row1.field_value, '')), 'NULL');
    end if;
end if;

if (row1.is_key = 1)
then
    WHERE_SQL := WHERE_SQL || 'and "' || row1.field_name || '" = ';

    if (row1.flag)
    then
        WHERE_SQL := WHERE_SQL || quote_literal(row1.field_value);
    else
        WHERE_SQL := WHERE_SQL ||
coalesce(quote_literal(nullif(row1.field_value, '')), 'NULL');
    end if;
end if;

end loop;

QUERY := 'INSERT INTO "' || SCHEMA_NAME || '"."' ||
quote_ident(_TABLE_NAME) || '"(' || QUERY_FIELDS || ') values(' || QUERY_VALUES
|| ');';

```

```

-- проверка на существование термина
for row1 in
    EXECUTE 'select sid from "' || SCHEMA_NAME || '"."' ||
quote_ident(_TABLE_NAME) || '"' || WHERE_SQL || ';'
loop
    result := row1.sid;
end loop;

if (Found)
then
    return '-5';
else
    EXECUTE QUERY;
end if;

-- сид нового термина
for row1 in
    EXECUTE 'select sid from "' || SCHEMA_NAME || '"."' ||
quote_ident(_TABLE_NAME) || '"' || WHERE_SQL || ';'
loop
    result := row1.sid;
end loop;

return result;
end;
$body$
LANGUAGE 'plpgsql'
VOLATILE
CALLED ON NULL INPUT
SECURITY INVOKER
COST 100.

```

4. Создание каталога классификатора:

```

CREATE OR REPLACE FUNCTION metadata.new_dir (
    varchar,
    varchar,
    integer
)
RETURNS integer AS
$body$
declare
    i_name          alias for $1; -- наименование
    i_shortcode     alias for $2; -- сокращенное наименование
    i_ierarh_sid    alias for $3; -- ссылка на родителя

begin

    -- пустое наименование
    if (nullif(i_name, '') isnull)
    then
        return -1;
    end if;

    -- если на заданном уровне есть каталог
    if exists(select 1

```

```

        from metadata.directory
        where "name" = i_name
            and coalesce(ierarh_sid,-1) = coalesce(i_ierarh_sid,-1))
    then
        return -2;
    end if;

    -- старшего каталога не существует
    if (i_ierarh_sid notnull and not exists(select 1 from metadata.directory
where sid = i_ierarh_sid))
    then
        return -3;
    end if;

    insert into metadata.directory ("name",shortname,ierarh_sid)
values(i_name,i_shortname,i_ierarh_sid);

return currval('metadata.directory_sid_seq');
end;
$body$
LANGUAGE 'plpgsql'
VOLATILE
CALLED ON NULL INPUT
SECURITY INVOKER
COST 100;

```

5. Каскадное удаление паспорта классификатора:

```

CREATE OR REPLACE FUNCTION metadata.del_passport (
    char
)
RETURNS integer AS
$body$
declare
    i_sid alias for $1; -- сид записи

    body_removed integer;

begin
    if exists(select 1 from directories.knsi_view where kview=1)
    then
        update public.classificator_name set delete=true
        where clf_spnum_sid = (select okok_code from
metadata.get_passport(i_sid));

        -- удалить таблицу
        select metadata.del_body(i_sid) into body_removed;
        if(body_removed != 0)
        then
            return body_removed;
        end if;
    else
        -- удалить таблицу
        select metadata.del_body(i_sid) into body_removed;
        if(body_removed != 0)
        then
            return body_removed;

```

```

end if;

-- удалить структуру
delete from metadata.struct where passport_sid = i_sid;

-- удалить паспорт
delete from metadata.passport where sid = i_sid;
end if;

return 0;
end;
$body$
LANGUAGE 'plpgsql'
VOLATILE
CALLED ON NULL INPUT
SECURITY INVOKER
COST 100.

```

6. Создание паспорта НСИ:

```

CREATE OR REPLACE FUNCTION metansi.new_passport (
    text,
    varchar,
    text,
    integer,
    integer,
    integer,
    timestamp,
    integer,
    timestamp,
    text,
    timestamp,
    text,
    timestamp,
    text,
    integer
)
RETURNS char AS
$body$
declare

    i_name            alias for $1; -- Наименование
    i_shortcode       alias for $2; -- Сокращенное наименование
    i_table_name      alias for $3; -- Наименование таблицы базы данных
    i_directory_sid   alias for $4; -- Директория
    i_grif_sid        alias for $5; -- Гриф секретности
    i_conduct_org_sid alias for $6; -- Организация ответственная за ве-
дение
    i_date_statement  alias for $7; -- Дата утверждения
    i_develop_org_sid alias for $8; -- Организация ответственная за раз-
работку
    i_date_receive     alias for $9; -- Дата получения
    i_reason_receive   alias for $10; -- Основание для получения
    i_date_change      alias for $11; -- Дата изменения
    i_reason_change     alias for $12; -- Основание для изменения
    i_date_enter       alias for $13; -- Дата ввода в действие
    i_remark           alias for $14; -- Примечание

```

```

        i_statement_org_sid  alias for $15; -- Организация ответственная за
утверждение

        result char(32);
        clfnum text;

begin

    -- проверка на наименование таблицы
    if (exists(select * from metansi.get_passports() where table_name =
i_table_name) and (i_table_name is null)) then
        return -1;
    end if;

    -- проверка на наименование
    if (exists(select * from metansi.get_passports() where "name" = i_name)
and (i_name is null)) then
        return -2;
    end if;

    -- проверка на сокращенное наименование
    if (exists(select * from metansi.get_passports() where shortname =
i_shortname) and (i_shortname is null)) then
        return -3;
    end if;

    -- проверка на гриф секретности
    if (not exists(select * from directories.grif where sid = i_grif_sid)
and (i_grif_sid is null)) then
        return -4;
    end if;

    if exists(select 1 from directories.knsi_view where kview=1)
then
        clfnum:=split_part(i_table_name,'nsi_',2);
        if (clfnum isnull) then
            raise EXCEPTION 'Поле "Наименование таблицы" должно быть вида
"nsi_111111" !';
        end if;
        perform
public.ngi_new_pasport(clfnum::integer,i_name,i_shortname,i_grif_sid,i_remark
);
    else
        insert into
metansi.passport("name",shortname,table_name,directory_sid,grif_sid,conduct_o
rg_sid,date_statement,develop_org_sid,date_receive,reason_receive,date_change
,reason_change,date_enter,remark,statement_org_sid)

values(i_name,i_shortname,i_table_name,i_directory_sid,i_grif_sid,i_conduct_o
rg_sid,i_date_statement,i_develop_org_sid,i_date_receive,i_reason_receive,i_d
ate_change,i_reason_change,i_date_enter,i_remark,i_statement_org_sid);
    end if;

    select * into result from metadata.hash(i_table_name||'.'||
i_name||'.'||i_shortname);

return result;

```

```

end;
$body$
LANGUAGE 'plpgsql'
VOLATILE
CALLED ON NULL INPUT
SECURITY INVOKER
COST 100.

```

7. Создание структуры НСИ:

```

CREATE OR REPLACE FUNCTION metansi.new_struct (
    char,
    varchar,
    text,
    integer,
    integer,
    char,
    text,
    integer,
    integer,
    integer,
    text,
    integer,
    text,
    integer
)
RETURNS integer AS
$body$
declare
    i_passport_sid          alias for $1;  -- паспорт НСИ
    i_name                  alias for $2;  -- наименование
    i_caption               alias for $3;  -- заголовок
    i_is_key                alias for $4;  -- является ли поле ключевым
    i_is_foreign            alias for $5;  -- является ли поле ссылочным
    i_foreign_passport_sid  alias for $6;  -- паспорт классификатора
    i_foreign_field         alias for $7;  -- отображаемое поле классификато-
pa
    i_datatype_sid          alias for $8;  -- тип данных
    i_length                alias for $9;  -- длина
    i_is_notempty           alias for $10; -- не пустое
    i_remark                alias for $11; -- примечание
    i_position              alias for $12; -- номер по порядку
    i_mask                  alias for $13; -- маска ввода
    i_ierarh_role           alias for $14; -- роль в построении иерархии

    l_position      INTEGER;  -- порядковый номер
    m_position      INTEGER;  -- максимальный порядковый номер

    l_sid  INTEGER;
    l_table_name TEXT;

    result      INTEGER;

    row1 record;

begin

```

```

select table_name into l_table_name from metansi.get_passports() where
sid = i_passport_sid;

-- если паспорт не определен или такого паспорта не существует
if (not Found)
then
    return -2;
end if;

-- поле с таким именем уже существует
if exists(select * from metansi.get_struct(i_passport_sid) where "name" =
i_name)
then
    return -3;
end if;

-- ссылочного классификатора не существует
if (not exists(select 1 from metadata.get_passports() where sid =
i_foreign_passport_sid) and (i_foreign_passport_sid is not null))
then
    return -4;
end if;

-- находим максимальный номер поля
select max("position") into m_position from
metansi.get_struct(i_passport_sid);

-- если номер поля не определен или больше чем последний номер то текущий
номер на единицу больше
if ((i_position > m_position) or (i_position ISNULL))
then
    l_position := coalesce(m_position,0) + 1;
else
    l_position := i_position;
end if;

-- перенумерация полей если номер поля меньше или равен существующим
for row1 in
    select sid, "position" + 1 as "position"
    from metansi.get_struct(i_passport_sid)
    where "position" >= l_position
    order by "position" desc
loop
    if exists(select 1 from directories.knsi_view where kview=1)
    then
        update public.ngi_struct set field_num = row1."position" where sid =
row1.sid;
    else
        update metansi.struct set "position" = row1."position" where sid =
row1.sid;
    end if;
end loop;

if exists(select 1 from directories.knsi_view where kview=1)
then

```

```

        insert into public.ngi_struct
(ngi_spnum_sid,field_num,fieldtype,valuetype,clf_spnum_sid,field_dbname,remark,field_length,is_noempty)
        values( split_part(l_table_name,'nsi_',2)::integer,l_position,
                (case when i_is_key=1 then 0 else 1 end),
                (case when i_datatype_sid=1 then 0 else i_datatype_sid end),
                (select okok_code from metadata.get_passports() where sid =
i_foreign_passport_sid)::integer,
                i_caption,i_remark,i_length,i_is_notempty);
    else
        insert into
metansi.struct(passport_sid,"name",caption,is_key,is_foreign,foreign_passport
_sid,foreign_field,datatype_sid,length,is_notempty,remark,position,mask,ierar
h_role)

values(i_passport_sid,i_name,i_caption,i_is_key,i_is_foreign,i_foreign_passpo
rt_sid,i_foreign_field,i_datatype_sid,i_length,i_is_notempty,i_remark,l_posit
ion,i_mask,i_ierarh_role);
    end if;

    -- сид новой записи
    select sid into l_sid from metansi.get_struct(i_passport_sid) where
"name" = i_name;

    result := 0;

    if exists(select 1 from directories.knsi_view where kview=1) then
        perform metansi.create_view_nkilo(split_part(l_table_name,'nsi_',2));
    else
        -- если таблица классификатора существует
        if (metansi.table_exists(l_table_name))
        then
            -- добавляем поле в таблицу
            select * into result from metansi.add_field(l_sid);
        end if;
    end if;

    -- если поле в таблицу добавлено успешно или не добавлялось вообще
    if (result = 0)
    then
        -- возвращаем сид новой записи
        return l_sid;
    else
        -- ошибка добавления поля в таблицу
        return result - 100;
    end if;

end;
$body$
LANGUAGE 'plpgsql'
VOLATILE
CALLED ON NULL INPUT
SECURITY INVOKER
COST 100.

```

8. Создание новых терминов НСИ:

```

CREATE OR REPLACE FUNCTION metansi.new_termin (
    char,
    text []
)
RETURNS char AS
$body$
declare
    i_passport_sid  alias for $1;  -- паспорт НСИ
    i_values         alias for $2;  -- значения

    _TABLE_NAME text;

    QUERY text;          -- запрос
    QUERY_FIELDS text;   -- список полей
    QUERY_VALUES text;   -- список значений

    SCHEMA_NAME text;

    WHERE_SQL text;

    row1 record;
    row2 record;

    result char(32);

begin

    select table_name into _TABLE_NAME from metansi.get_passports() where sid =
i_passport_sid;

    if (not metansi.table_exists(_TABLE_NAME))
    then
        return '-1';
    end if;

    -- пустой массив значений
    if (i_values isnull)
    then
        return '-2';
    end if;

    SCHEMA_NAME = 'nsi';

    WHERE_SQL := 'where 1=1 ';

    for row1 in
        select s."name" as field_name,
            (i_values)[s."position"] as field_value,
            -- если поле символьное, дата или время, то добавляем кавычки в
условие
            case when s.datatype_sid in (1,4,5) then TRUE else FALSE end as
flag,
            s."position" as field_num,
            s.is_key,
            s.is_foreign,
            p.table_name as foreign_table_name,
            s."length",

```

```

        s.datatype_sid
    from metansi.get_struct(i_passport_sid) s
        left join metansi.get_passports() p on p.sid =
s.foreign_passport_sid
    order by s."position"
loop

    -- проверка на ссылочный термин
    if ((row1.foreign_table_name is not null) and (trim(row1.field_value)
<> ''))
    then
        for row2 in
            EXECUTE 'select sid from "nsi"."' ||
quote_ident(row1.foreign_table_name) || '" where sid = ' ||
quote_literal(row1.field_value) || ';'
        loop
            end loop;

            if (not Found)
            then
                return '-3';
            end if;
        end if;

    -- проверка на размерность поля (только символьное)
    if ((row1.is_foreign = 0) and (row1.datatype_sid = 1) and
(row1."length" > 0) and (length(row1.field_value) > row1."length"))
    then
        return '-4';
    end if;

    if (row1.field_num = 1)
    then
        QUERY_FIELDS := ' "' || row1.field_name || '"';

        if (row1.flag)
        then
            if ((row1.is_foreign = 1) and (row1.field_value = ''))
            then
                QUERY_VALUES := 'NULL';
            elsif ((row1.is_foreign = 1) and (row1.field_value <> ''))
            then
                QUERY_VALUES := quote_literal(row1.field_value);
            else
                QUERY_VALUES := quote_literal(row1.field_value);
            end if;
        else
            QUERY_VALUES :=
coalesce(quote_literal(nullif(row1.field_value, '')), 'NULL');
        end if;

    else
        QUERY_FIELDS := QUERY_FIELDS || ' , "' || row1.field_name || '"';

        if (row1.flag)
        then
            if ((row1.is_foreign = 1) and (row1.field_value = ''))

```

```

        then
            QUERY_VALUES := QUERY_VALUES || ',NULL';
        elsif ((row1.is_foreign = 1) and (row1.field_value <> ''))
        then
            QUERY_VALUES := QUERY_VALUES || ', ' ||
quote_literal(row1.field_value);
        else
            QUERY_VALUES := QUERY_VALUES || ', ' ||
quote_literal(row1.field_value);
        end if;
    else
        QUERY_VALUES := QUERY_VALUES || ', ' ||
coalesce(quote_literal(nullif(row1.field_value, '')), 'NULL');
    end if;
end if;

if (row1.is_key = 1)
then
    WHERE_SQL := WHERE_SQL || 'and "' || row1.field_name || '" = ';

    if (row1.flag)
    then
        WHERE_SQL := WHERE_SQL || quote_literal(row1.field_value);
    else
        WHERE_SQL := WHERE_SQL ||
coalesce(quote_literal(nullif(row1.field_value, '')), 'NULL');
    end if;
end if;
end loop;

QUERY := 'INSERT INTO "' || SCHEMA_NAME || '"."' ||
quote_ident(_TABLE_NAME) || '"(' || QUERY_FIELDS || ') values(' || QUERY_VALUES
|| ');';

EXECUTE QUERY;

-- сид нового термина
for row1 in
    EXECUTE 'select sid from "' || SCHEMA_NAME || '"."' ||
quote_ident(_TABLE_NAME) || '" ' || WHERE_SQL || '; '
loop
    result := row1.sid;
end loop;

return result;
end;
$body$
LANGUAGE 'plpgsql'
VOLATILE
CALLED ON NULL INPUT
SECURITY INVOKER
COST 100.

```

9. Удаление паспорта НСИ:

```

CREATE OR REPLACE FUNCTION metansi.del_passport (
    char
)

```

```

RETURNS integer AS
$body$
declare
    i_sid alias for $1; -- сид записи

    _TABLE_NAME text;

begin

    -- нет HCI
    if (not exists(select * from metansi.get_passports() where sid = i_sid))
    then
        return -1;
    end if;

    select table_name into _TABLE_NAME from metansi.get_passports() where sid =
i_sid;

    if exists(select 1 from directories.knsi_view where kview=1)
    then
        perform
public.ngi_del_pasport(split_part(_TABLE_NAME,'nsi_',2)::integer);

        -- удалить таблицу
        if (metansi.table_exists(_TABLE_NAME))
        then
            EXECUTE 'DROP VIEW "nsi"."' || _TABLE_NAME || '" CASCADE;';
        end if;

        -- удалить журнал
        if (metansi.table_exists(_TABLE_NAME || '_journal'))
        then
            EXECUTE 'DROP VIEW "nsi"."' || _TABLE_NAME || '_journal";';
        end if;
    else
        -- удалить структуру
        delete from metansi.struct where passport_sid=i_sid;

        -- удалить таблицу
        if (metansi.table_exists(_TABLE_NAME))
        then
            EXECUTE 'DROP TABLE "nsi"."' || _TABLE_NAME || '" CASCADE;';
            EXECUTE 'DROP FUNCTION "nsi".generate_' || _TABLE_NAME || '_pk();';
            EXECUTE 'DROP FUNCTION "nsi".update_' || _TABLE_NAME || '_pk();';
        end if;

        -- удалить журнал
        if (metansi.table_exists(_TABLE_NAME || '_journal'))
        then
            EXECUTE 'DROP TABLE "nsi"."' || _TABLE_NAME || '_journal";';
            EXECUTE 'DROP FUNCTION "nsi"."' || _TABLE_NAME || '_journal_insert();';
            EXECUTE 'DROP FUNCTION "nsi"."' || _TABLE_NAME || '_journal_update();';
            EXECUTE 'DROP FUNCTION "nsi"."' || _TABLE_NAME || '_journal_delete();';
        end if;

        -- удалить паспорт
        delete from metansi.passport where sid = i_sid;

```

```
        end if;

        return 0;
end;
$body$
LANGUAGE 'plpgsql'
VOLATILE
CALLED ON NULL INPUT
SECURITY INVOKER
COST 100.
```

ПРИЛОЖЕНИЕ Г. ПРЕЗЕНТАЦИЯ

Ведение и актуализация классификаторов по пирамидальному принципу в сложных информационных системах

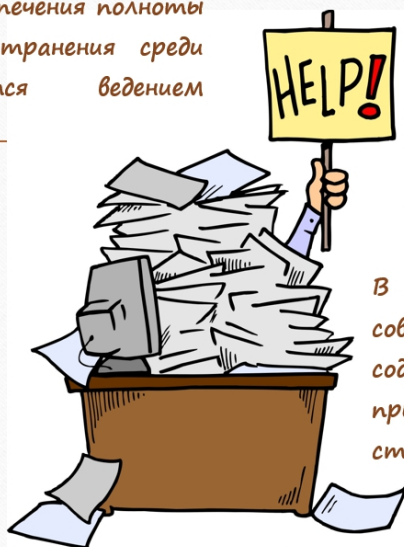
магистрант 2 курса
Полковникова Светлана
Андреевна

Классификатор – это сгруппированный, по различным признакам, перечень наименованных объектов, где каждому из них присвоен уникальный код, в соответствии с их общими признаками или различиями.



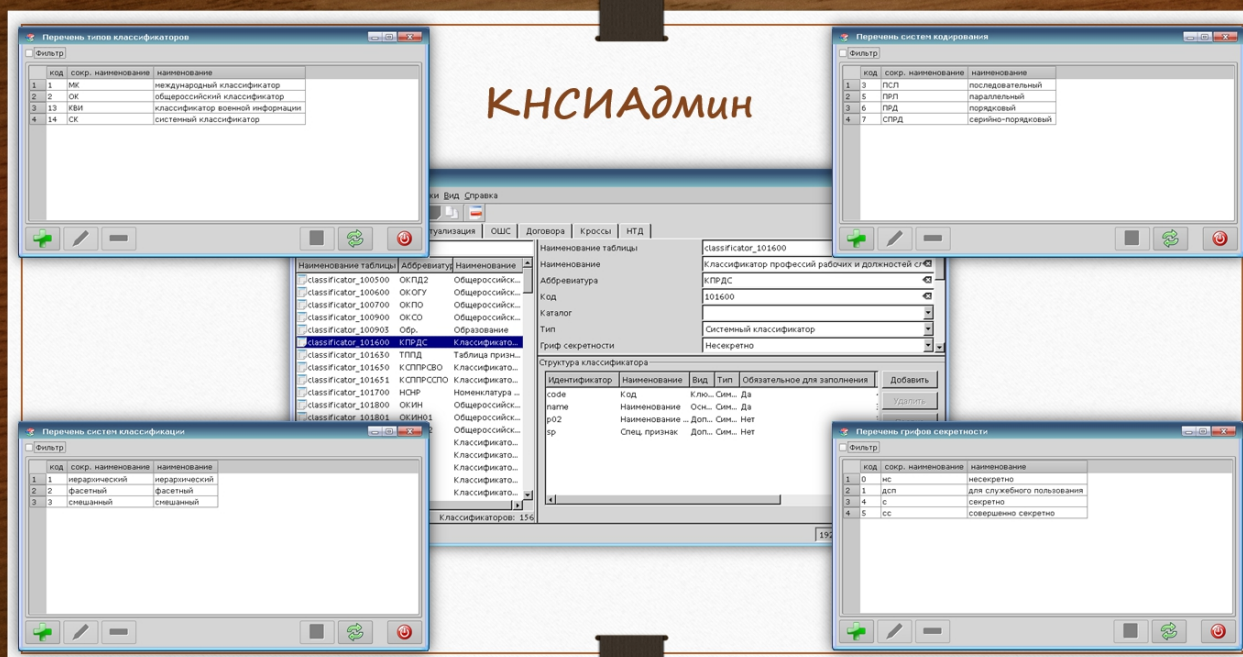
3 Слайд

Процесс поддержания классификаторов в актуальном состоянии, обеспечения полноты информации и ее распространения среди пользователей называется ведением классификаторов.



В процессе ведения классификаторов совершенствуются их состав и содержание, но не нарушаются принципы их построения и целостная структура.

4 Слайд



Программы для ведения классификаторов и НСИ



Программа 1С

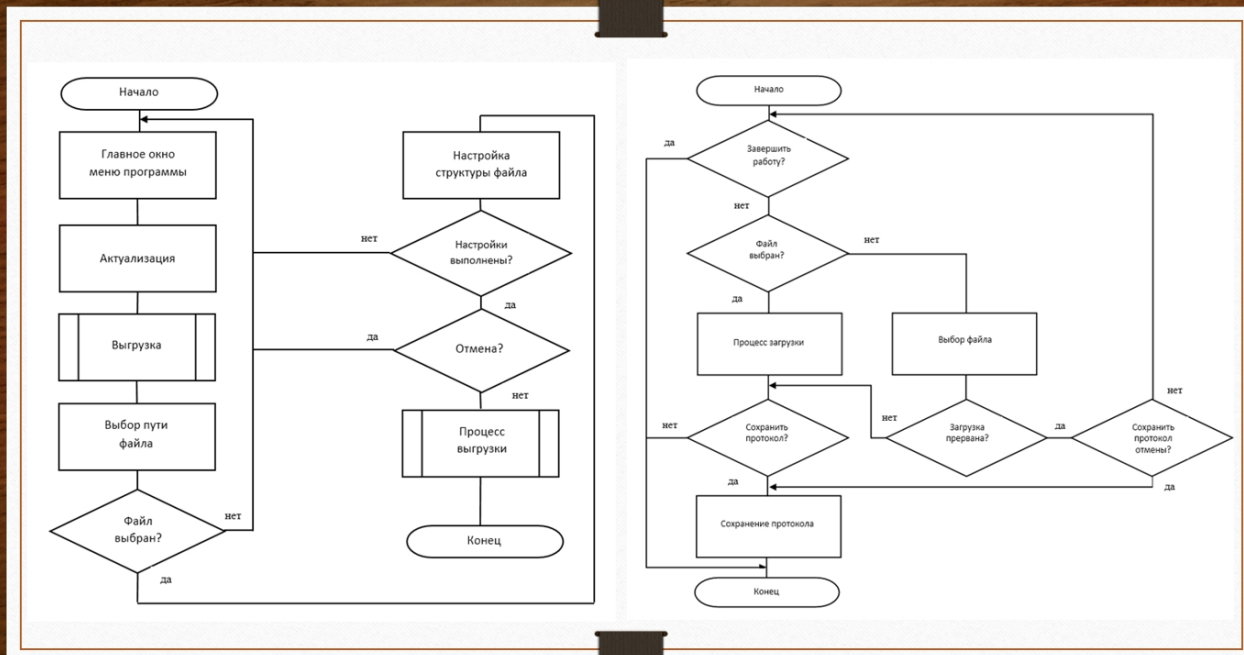


АСУ НСИ 2.0

Сравнение разрабатываемого программного обеспечения с аналогами

	АСУ НСИ 2.0	1С	НСИ
1. Поддерживаемые ОС	Windows	Windows Linux	Windows Linux
2. Стоимость	Стоимость аренды мощностей у поставщика облачных вычислений + стоимость развертывания системы + стоимость администрирования и предоставления доступа к системе	Лицензия оплачивается каждый год + стоимость обслуживания	Лицензия оплачивается 1 раз + стоимость обслуживания
3. Ведение классификаторов, НСИ, кроссов	Только ведение НСИ	Ведение классификаторов и НСИ (но требует разработки конфигурации)	Ведение классификаторов, НСИ, кроссов
4. Форматы выгрузки	doc, docx	xl	zip, xml, xmlx, json, jsonx
5. Форма выгрузки	Выгружает только справочники	Выгружает отдельные классификаторы	Возможна выгрузка как отдельных классификаторов, так и всех сразу + одновременная выгрузка НСИ и кроссов

Блок-схема алгоритма выгрузки



Блок-схема алгоритма загрузки

Описание базы данных:

Ниже представлен пример создания таблицы структуры классификатора:

```

CREATE TABLE metadata.struct (
  sid BIGSERIAL,
  passport_sid CHAR(32) NOT NULL,
  field_name VARCHAR(100) NOT NULL,
  field_caption TEXT,
  field_type_sid INTEGER NOT NULL,
  field_datatype_sid INTEGER NOT NULL,
  field_size INTEGER DEFAULT 0 NOT NULL,
  is_key INTEGER NOT NULL,
  is_notempty INTEGER NOT NULL,
  faset_fields TEXT,
  remark TEXT,
  field_num INTEGER NOT NULL,
  is_control INTEGER NOT NULL,
  field_struct TEXT,
  is_sostav INTEGER NOT NULL,
  field_gen_code INTEGER,
  field_mask TEXT,
  is_unique INTEGER NOT NULL,
  faset_sid CHAR(32)
)
  
```

Ниже представлен пример создания таблицы терминов НСИ:

```

CREATE TABLE nsi.nsi_791001 (
  sid CHAR(32) NOT NULL,
  path public.ltree,
  field1 CHAR(32) NOT NULL,
  field2 CHAR(32) NOT NULL,
  field3 INTEGER,
  field4 INTEGER
)
  
```

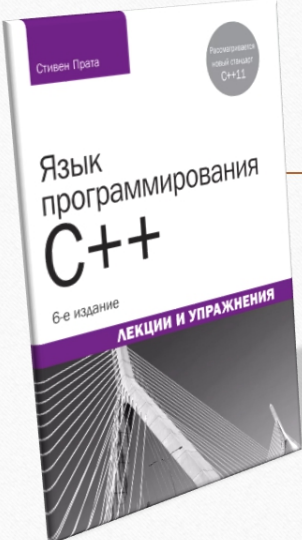
Диаграммы последовательности



Технические характеристики :

- операционная система Linux (Astra Linux «Орёл»), Microsoft Windows;
- объем занимаемый на диске в ОС Microsoft Windows – 15 МБ,
- объем занимаемый на диске в ОС Linux – 46 МБ,
- объем программного модуля в ОС Microsoft Windows – 226,5 КБ,
- объем программного модуля в ОС Linux – 1,7 МБ,
- ПЭВМ с процессором совместимым с Intel x86, частота процессора не менее 2 ГГц;
- оперативная память не менее 1Гб;
- накопитель на жестком диске не менее 256 Гб;

Заключение



Подсистема актуализации:

- 2 программных модуля
- Язык программирования *Си++*
- Библиотека *Qt*



Спасибо за внимание!

магистрант 2 курса
Полковникова Светлана
Андреевна