МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

ТЮМЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ НАУК

КАФЕДРА ИНФОРМАЦИОННЫХ СИСТЕМ

Отчет по дисциплине:

Программная инженерия

на тему: разработка программного приложения

«Утилизация отходов»

Выполнил:

студент 4 курса

группы ПИ 155-2

Максимова Т.А.

Проверил:

Красиков В.Е.

Оглавление

[**Введение** 3](#_Toc533594191)

[**Диаграмма IDEF1x** 4](#_Toc533594192)

[**База Данных** 6](#_Toc533594193)

[**Разработка Веб-службы** 10](#_Toc533594194)

[**Реализация клиентского приложения** 12](#_Toc533594195)

## **Введение**

Вместе с ростом численности населения планеты и уровнем потребления, растет и количество бытовых отходов в России. В ближайшем будущем оно может достигнуть угрожающих размеров. Встает очевидный вопрос о ликвидации, а в идеале последующей рециклинге. Но здесь сказывается недостаток организаций, которые занимаются вывозом, утилизацией бытовых и промышленных отходов.

Существуют этапы для утилизации бытовых отходов. Перед окончательной ликвидацией ненужных остатков жизнедеятельности людей следует четко знать план действий и представлять последовательность их выполнения.

Первым этап является сбор и вывоз бытовых отходов. Нельзя просто собрать и сжечь весь мусор без сортировки. Следует предварительно рассортировать его по типам, для этого возле большинства жилых массивов расположены контейнеры с названием соответствующего им мусора. Под каждую такую емкость должна иметься соответствующая машина для погрузки и транспортировки содержимого.

Вторым этапом является хранение. Существуют специальные полигоны или свалки для накопления мусора, по городским нормам предусмотрены сроки его содержания, объемы и отведенная под них площадь.

Третий этап – транспортировка отходов с полигонов. Необходимы погрузочные механизмы и машины для доставки мусора в определенные места и на предприятия для ликвидации.

Четвертый этап – утилизация. Самыми распространенными видами являются: мусоросжигание, захоронение и брикетирование. Каждый из них имеет свои отрицательные и положительные стороны.

## **Диаграмма IDEF1x**

Методология моделирования IDEF1x, является расширением стандарта IDEF1 и предназначена для описания данных (информации). В ее основе лежит язык семантического моделирования, основанного на концепции "сущность — связь", позволяющей определять данные и связи между ними. Методология используется для создания информационной модели предметной области с помощью идентификации ее сущностей и связей между ними. Чаще всего такая методология используется для описания данных в целях последующей автоматизации их обработки с помощью систем управления базами данных. Связи в IDEF1X представляют собой ссылки, соединения и ассоциации между сущностями. Связи – это глаголы, которые показывают, как соотносятся сущности между собой.

Диаграмма IDEF1x (см. рис. 1), разработанная для текущего проекта – утилизация отходов, показывается связи и отношения необходимые для разработки базы данных.

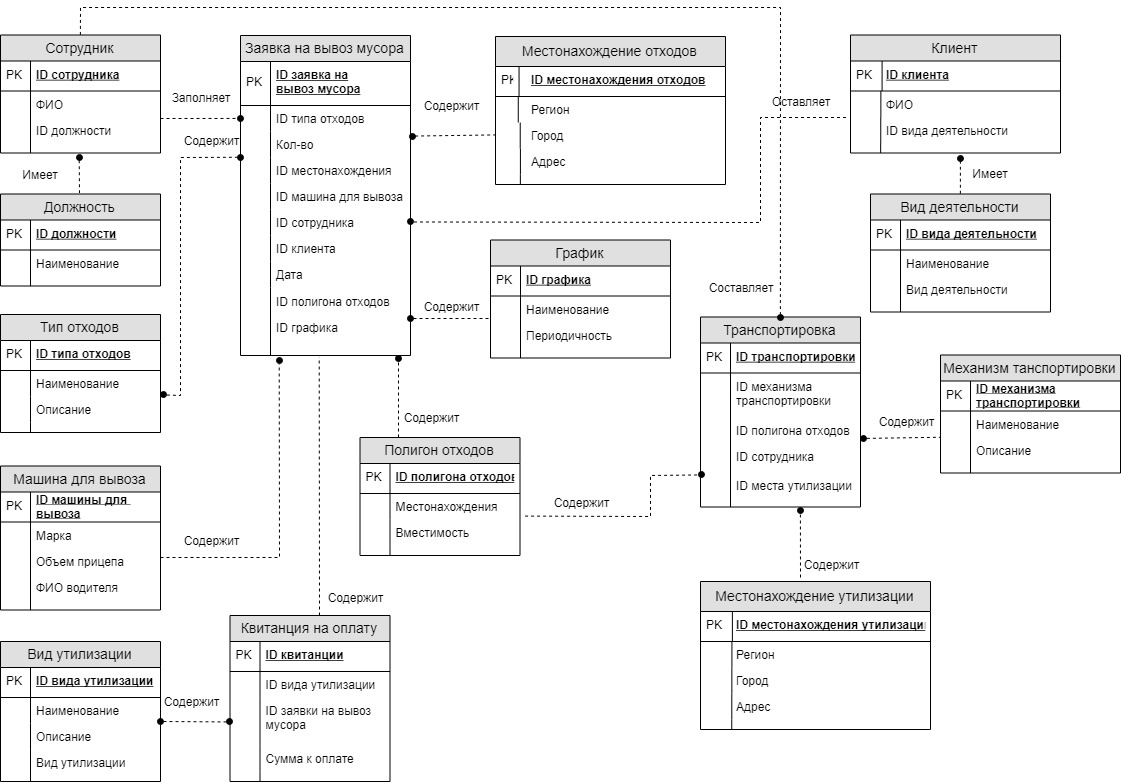


Рисунок 1. Диаграмма IDEF1x

После разработки диаграммы IDEF1x требуется создать базу данных с помощью специального программного обеспечения.

## **База Данных**

СУБД (Система управления базами данных) – это совокупность языковых и программных средств, предназначенных для создания, ведения и совместного использования БД многими пользователями. Система управления базами данных (СУБД) является посредником между базой данных и ее пользователями.

По способу доступа к БД различают:

* Файл-серверные
* Клиент-серверные
* Встраиваемые

В файл-серверных СУБД файлы данных располагаются централизованно на файл-серверные. СУБД располагается на каждом клиентском компьютере (рабочей станции). Доступ СУБД к данным осуществляется через локальную сеть. Синхронизация чтений и обновлений осуществляется посредством файловых блокировок. Примеры: Microsoft Access, Paradox, dBase, FoxPro, Visual FoxPro.

Клиент-серверная СУБД располагается на сервере вместе с БД и осуществляет доступ к БД непосредственно, в монопольном режиме. Все клиентские запросы на обработку данных обрабатываются клиент-серверной СУБД централизованно. Примеры: Oracle, Firebird, Interbase, IBM DB2, Informix, MS SQL Server, Sybase Adaptive Server Enterprise, PostgreSQL, MySQL, Caché, ЛИНТЕР.

Встраиваемая СУБД — СУБД, которая может поставляться как составная часть некоторого программного продукта, не требуя процедуры самостоятельной установки. Встраиваемая СУБД предназначена для локального хранения данных своего приложения и не рассчитана на коллективное использование в сети. Физически встраиваемая СУБД чаще всего реализована в виде подключаемой библиотеки. Доступ к данным со стороны приложения может происходить через SQL либо через специальные программные интерфейсы. Примеры: OpenEdge, SQLite, BerkeleyDB, Firebird Embedded, Microsoft SQL Server Compact, ЛИНТЕР.

В качестве СУБД был выбран SQL Server так как:

SQL Server — это хорошо масштабируемый, полностью реляционный, быстродействующий многопользовательский сервер баз данных масштаба предприятия, способный обрабатывать большие объемы данных для клиент-серверных приложений. Основные характеристики:

* многопользовательская поддержка;
* многоплатформность;
* параллельные архивирование и восстановление БД;
* репликация данных;
* распределенные запросы;
* распределенные транзакции;

База данных (см. рис. 2), созданная в системе управления базами данных MS SQL, включает в себя 15 таблиц, описанных в диаграмме IDEF1x.

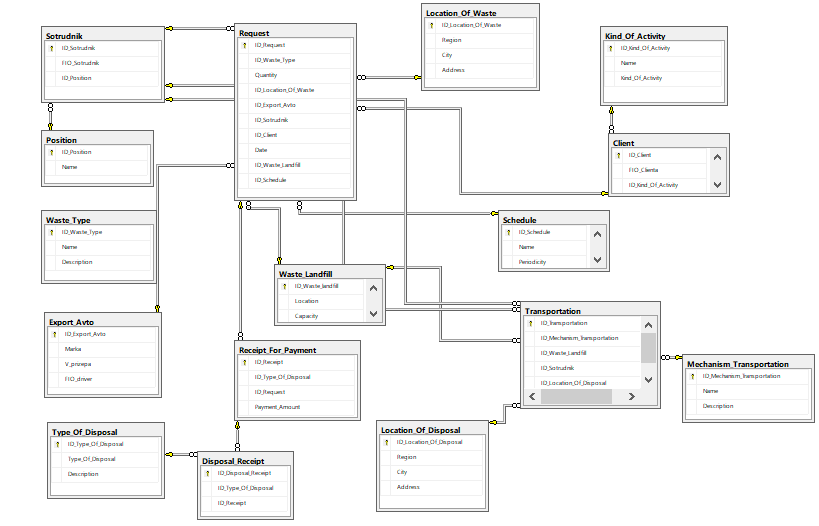


Рисунок 2. Диаграмма базы данных

В ходе работы были созданы и заполнены все таблицы базы данных. Примеры заполненных таблиц (см. рис. 3-8).

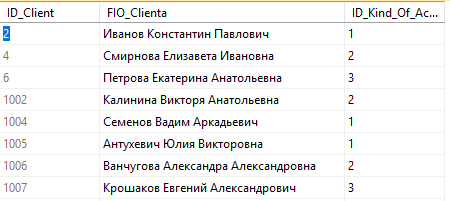


Рисунок 3. Таблица «Клиенты»

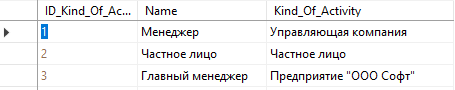


Рисунок 4. Таблица «Вид деятельности»

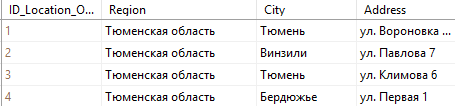


Рисунок 5. Таблица «Местонахождение отходов»

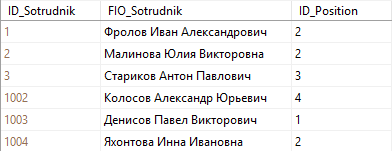


Рисунок 6. Таблица «Сотрудники»

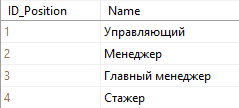


Рисунок 7. Таблица «Должности»

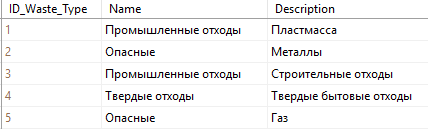


Рисунок 8. Таблица «Виды отходов»

Для наглядного представления были созданы представления. Пример представления (см. рис. 9).



Рисунок 9. Представление «Заявка на вывоз мусора»

## **Разработка Веб-службы**

Для отделения бизнес-логики и логики работы с БД от клиентского приложения используются веб-службы, осуществляющие взаимодействие по REST.

Веб-служба, веб-сервис (англ. web service) — идентифицируемая уникальным веб-адресом (URL-адресом) программная система со стандартизированными интерфейсами, а также HTML-документ сайта, отображаемый браузером пользователя. Веб-службы могут взаимодействовать друг с другом и со сторонними приложениями посредством сообщений, основанных на определённых протоколах и соглашениях.

REST (сокращение от англ. Representational State Transfer — «передача состояния представления») — архитектурный стиль взаимодействия компонентов распределённого приложения в сети. REST определяет ряд архитектурных принципов проектирования Web-сервисов, ориентированных на системные ресурсы, включая способы обработки и передачи состояний ресурсов по HTTP разнообразными клиентскими приложениями, написанными на различных языках программирования. За последние несколько лет REST стала преобладающей моделью проектирования Web-сервисов.

Как происходит управление информацией сервиса – это целиком и полностью основывается на протоколе передачи данных. Наиболее распространенный протокол конечно же HTTP. Так вот, для HTTP действие над данными задается с помощью методов: GET (получить), PUT (добавить, заменить), POST (добавить, изменить, удалить), DELETE (удалить). Таким образом, действия CRUD (Create-Read-Updtae-Delete) могут выполняться как со всеми 4-мя методами, так и только с помощью GET и POST. Для каждой таблицы созданы контроллеры, в которых прописаны данные методы (приложение 1). С помощью строки http://localhost:52256/api/Clients, например, можно получить список всех имеющихся клиентов (см. рис. 10).



Рисунок 10. Список всех клиентов

## **Реализация клиентского приложения**

Для удобного пользования клиента с базой был создан интерфейс приложения. Здесь пользователь может: удалить, редактировать и добавлять необходимые ему данные. Пример таблица «Клиенты» (см. рис. 11).

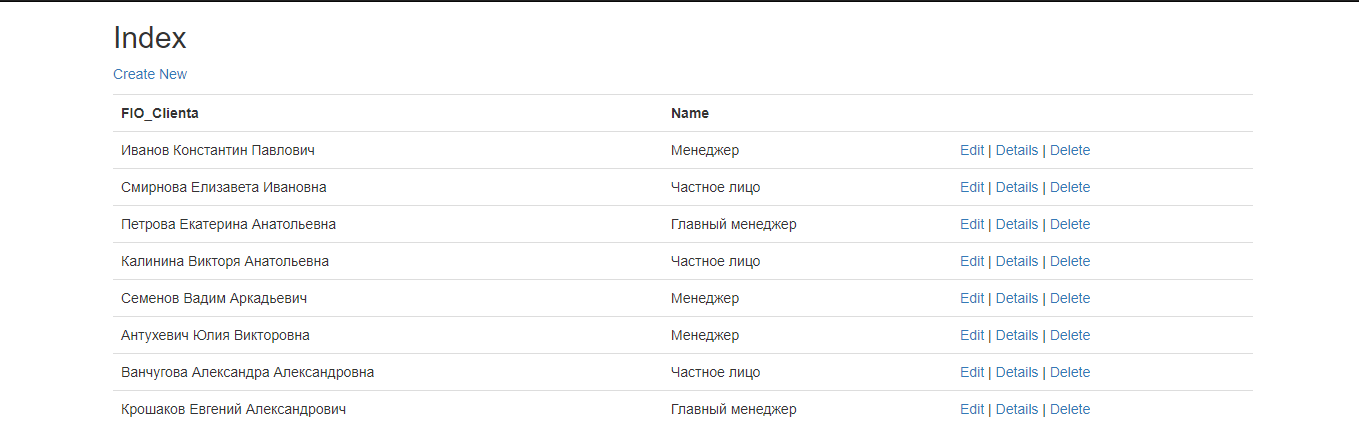


Рисунок 11. Интерфейс таблицы «Клиенты»

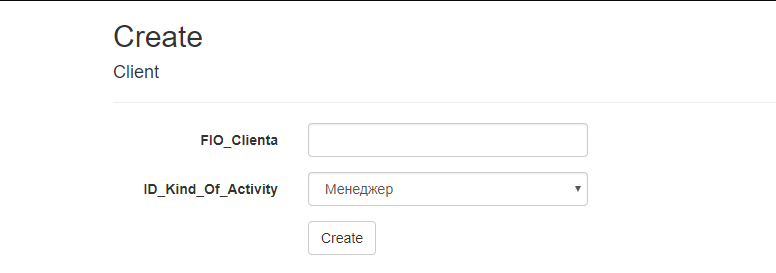


Рисунок 12. Интерфейс создания клиента.

## **Заключение**

В ходе разработки приложения «Утилизация отходов» была создана диаграмма IDEF1x, по данной диаграмме – создана и заполнены база данных в SQL Server. Были подключены веб-службы, осуществляющие взаимодействие по REST и реализовано клиентское приложение, которое позволяет пользователю в удобном формате работать с базой данных.

## **Список использованной литературы**

1. «Утилизация отходов» [Электронный ресурс], URL: https://vtorothodi.ru/utilizaciya/etapy-i-metody-utilizacii-bytovyx-otxodov
2. «Основы методологии IDEF1x» [Электронный ресурс], URL: https://www.cfin.ru/vernikov/idef/idef1x.shtml
3. «Microsoft SQL Server» [Электронный ресурс], URL: https://ru.wikipedia.org/wiki/Microsoft\_SQL\_Server
4. «Веб-сервисы» [Электронный ресурс], URL: https://www.ibm.com/developerworks/ru/library/ws-restfu/index.html
5. «Архитектура REST» [Электронный ресурс], URL: https://habr.com/post/38730/

ПРИЛОЖЕНИЕ 1

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.Entity;

using System.Linq;

using System.Net;

using System.Web;

using System.Web.Mvc;

using WebApplication2.Entities;

namespace WebApplication2.Controllers

{

public class Clients1Controller : Controller

{

private RecyclingEntities db = new RecyclingEntities();

// GET: Clients1

public ActionResult Index()

{

var client = db.Client.Include(c => c.Kind\_Of\_Activity);

return View(client.ToList());

}

// GET: Clients1/Details/5

public ActionResult Details(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Client client = db.Client.Find(id);

if (client == null)

{

return HttpNotFound();

}

return View(client);

}

// GET: Clients1/Create

public ActionResult Create()

{

ViewBag.ID\_Kind\_Of\_Activity = new SelectList(db.Kind\_Of\_Activity, "ID\_Kind\_Of\_Activity", "Name");

return View();

}

// POST: Clients1/Create

// Чтобы защититься от атак чрезмерной передачи данных, включите определенные свойства, для которых следует установить привязку. Дополнительные

// сведения см. в статье https://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Create([Bind(Include = "ID\_Client,FIO\_Clienta,ID\_Kind\_Of\_Activity")] Client client)

{

if (ModelState.IsValid)

{

db.Client.Add(client);

db.SaveChanges();

return RedirectToAction("Index");

}

ViewBag.ID\_Kind\_Of\_Activity = new SelectList(db.Kind\_Of\_Activity, "ID\_Kind\_Of\_Activity", "Name", client.ID\_Kind\_Of\_Activity);

return View(client);

}

// GET: Clients1/Edit/5

public ActionResult Edit(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Client client = db.Client.Find(id);

if (client == null)

{

return HttpNotFound();

}

ViewBag.ID\_Kind\_Of\_Activity = new SelectList(db.Kind\_Of\_Activity, "ID\_Kind\_Of\_Activity", "Name", client.ID\_Kind\_Of\_Activity);

return View(client);

}

// POST: Clients1/Edit/5

// Чтобы защититься от атак чрезмерной передачи данных, включите определенные свойства, для которых следует установить привязку. Дополнительные

// сведения см. в статье https://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Edit([Bind(Include = "ID\_Client,FIO\_Clienta,ID\_Kind\_Of\_Activity")] Client client)

{

if (ModelState.IsValid)

{

db.Entry(client).State = EntityState.Modified;

db.SaveChanges();

return RedirectToAction("Index");

}

ViewBag.ID\_Kind\_Of\_Activity = new SelectList(db.Kind\_Of\_Activity, "ID\_Kind\_Of\_Activity", "Name", client.ID\_Kind\_Of\_Activity);

return View(client);

}

// GET: Clients1/Delete/5

public ActionResult Delete(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Client client = db.Client.Find(id);

if (client == null)

{

return HttpNotFound();

}

return View(client);

}

// POST: Clients1/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public ActionResult DeleteConfirmed(int id)

{

Client client = db.Client.Find(id);

db.Client.Remove(client);

db.SaveChanges();

return RedirectToAction("Index");

}

protected override void Dispose(bool disposing)

{

if (disposing)

{

db.Dispose();

}

base.Dispose(disposing);

}

}

}