МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования

**«Московский технический университет связи и информатики»**

Кафедра «Информационные технологии»

Лабораторная работа №6

**«Технологии доступа к данным»**

по дисциплине

Информационные технологии и программирование

Выполнил: студент гр. БЭИ2202

Кулешов А. С.

Вариант 16

Проверил: Халабия Р.Ф.

Москва, 2024 г

1. Цели и задачи

Изучить и освоить механизмы доступа и передачи данных (XML и JSON).

1. Постановка задачи

Задание:

Написать программу согласно варианту, которая добавляет, удаляет и изменяет записи данных, хранящиеся определенном формате в файле заданного типа. Если файла нет, то необходимо его создавать, если все записи в файле удалены, то файл должен оставаться пустым. Также мы должны выводить информацию из файла на экран в табличном виде. Для редактирования данных необходимо небольшое простое меню. Таблицы организовать внутри программы как классы. И главное таблицы связанны между собой.

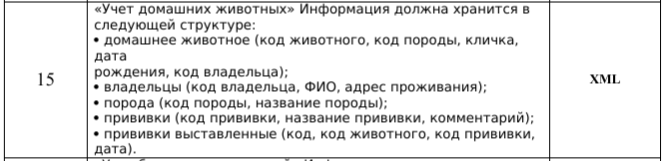


Рисунок 1 – Индивидуальное задание

1. Листинг программы

#include "tinyxml2.cpp"

// Функция для загрузки XML файла

bool LoadXML(const std::string& fileName, XMLDocument& doc) {

XMLError eResult = doc.LoadFile(fileName.c\_str());

cout << eResult << endl;

if (eResult != XML\_SUCCESS) {

std::cerr << "Error loading file: " << fileName << std::endl;

return false;

}

return true;

}

// Функция для сохранения XML файла

bool SaveXML(const std::string& fileName, XMLDocument& doc) {

XMLError eResult = doc.SaveFile(fileName.c\_str());

if (eResult != XML\_SUCCESS) {

std::cerr << "Error saving file: " << fileName << std::endl;

return false;

}

return true;

}

int get\_static\_id(XMLDocument& doc) {

using namespace tinyxml2;

XMLElement\* root = doc.FirstChildElement("Static");

auto attr = root->FindAttribute("id\_counter");

std::cout << "retrieved static\_id " << attr->IntValue() << std::endl;

return attr->IntValue();

}

std::string get\_table\_name(int val) {

vector<std::string> vc = { "Pets", "Owners", "Breeds", "Vaccines", "PetVaccines" };

return vc[val];

}

int update\_static\_id(XMLDocument& doc) {

int static\_id = get\_static\_id(doc) + 1;

XMLElement\* root = doc.FirstChildElement("Static");

root->SetAttribute("id\_counter", static\_id);

cout << root->FindAttribute("id\_counter")->IntValue() << endl;

return static\_id;

}

void AddObject(XMLDocument& doc, DBObject\* obj, const std::string& table) {

XMLElement\* root = doc.FirstChildElement(table.c\_str());

XMLElement\* newObject = doc.NewElement("Object");

obj->SetDefaultAttributes(doc, newObject);

obj->SetAttributes(doc, newObject);

root->InsertEndChild(newObject);

}

void checkFile(const std::string& fileName, XMLDocument& doc) {

// Если файл не существует, создаем новый документ

if (!LoadXML(fileName, doc)) {

XMLElement\* root = doc.NewElement("Pets");

doc.InsertEndChild(root);

root = doc.NewElement("Owners");

doc.InsertEndChild(root);

root = doc.NewElement("Breeds");

doc.InsertEndChild(root);

root = doc.NewElement("Vaccines");

doc.InsertEndChild(root);

root = doc.NewElement("PetVaccines");

doc.InsertEndChild(root);

root = doc.NewElement("Static");

root->SetAttribute("id\_counter", 1);

doc.InsertEndChild(root);

SaveXML(fileName, doc);

}

}

DBObject\* get\_class(int val) {

if (val == 0) return new Pet();

if (val == 1) return new Ownerw();

if (val == 2) return new Breed();

if (val == 3) return new Vaccine();

if (val == 4) return new PetVaccine();

}

class DBObject {

public:

int id;

virtual void SetAttributes(tinyxml2::XMLDocument& doc, tinyxml2::XMLElement\* elem) = 0;

void SetDefaultAttributes(tinyxml2::XMLDocument& doc, tinyxml2::XMLElement\* elem) {

id = update\_static\_id(doc);

elem->SetAttribute("id", id);

}

virtual std::vector<std::string> get\_description() = 0;

//virtual std::vector<std::string> get\_data() = 0;

};

void AddObject(XMLDocument& doc, DBObject\* obj, const std::string& table);

class Pet : public DBObject {

public:

int breedCode;

std::string name;

std::string birthDate;

int ownerCode;

void SetAttributes(tinyxml2::XMLDocument& doc, tinyxml2::XMLElement\* elem) override {

elem->SetAttribute("breedCode", breedCode);

elem->SetAttribute("name", name.c\_str());

elem->SetAttribute("birthDate", birthDate.c\_str());

elem->SetAttribute("ownerCode", ownerCode);

}

Pet() = default;

Pet(const std::string& s1, const std::string& s2, const std::string& s3, const std::string& s4) {

breedCode = atoi(s1.c\_str());

name = s2;

birthDate = s3;

ownerCode = atoi(s4.c\_str());

}

std::vector<std::string> get\_description() override {

return { "breedCode", "name", "birthDate", "ownerCode" };

}

};

class Ownerw : public DBObject {

public:

std::string fullName;

std::string address;

void SetAttributes(tinyxml2::XMLDocument& doc, tinyxml2::XMLElement\* elem) override {

elem->SetAttribute("fullName", fullName.c\_str());

elem->SetAttribute("address", address.c\_str());

}

Ownerw() = default;

Ownerw(const std::string& s1, const std::string& s2) {

fullName = s1;

address = s2;

}

std::vector<std::string> get\_description() override {

return { "fullName", "address" };

}

};

class Breed : public DBObject {

public:

std::string breedName;

void SetAttributes(tinyxml2::XMLDocument& doc, tinyxml2::XMLElement\* elem) override {

elem->SetAttribute("breedName", breedName.c\_str());

}

Breed() = default;

Breed(const std::string& s1) {

breedName = s1;

}

std::vector<std::string> get\_description() override {

return { "breedName" };

}

};

class Vaccine : public DBObject {

public:

std::string vaccineName;

std::string comment;

void SetAttributes(tinyxml2::XMLDocument& doc, tinyxml2::XMLElement\* elem) override {

elem->SetAttribute("vaccineName", vaccineName.c\_str());

elem->SetAttribute("comment", comment.c\_str());

}

Vaccine() = default;

Vaccine(const std::string& s1, const std::string& s2) {

vaccineName = s1;

comment = s2;

}

std::vector<std::string> get\_description() override {

return { "vaccineName", "comment"};

}

};

class PetVaccine : public DBObject {

public:

int petCode;

int vaccineCode;

std::string date;

void SetAttributes(tinyxml2::XMLDocument& doc, tinyxml2::XMLElement\* elem) override {

elem->SetAttribute("petCode", petCode);

elem->SetAttribute("vaccineCode", vaccineCode);

elem->SetAttribute("date", date.c\_str());

}

PetVaccine() = default;

PetVaccine(const std::string& s1, const std::string& s2, const std::string& s3) {

petCode = atoi(s1.c\_str());

vaccineCode = atoi(s2.c\_str());

date = s3;

}

std::vector<std::string> get\_description() override {

return { "petCode", "vaccineCode", "date"};

}

};

1. Схема классов

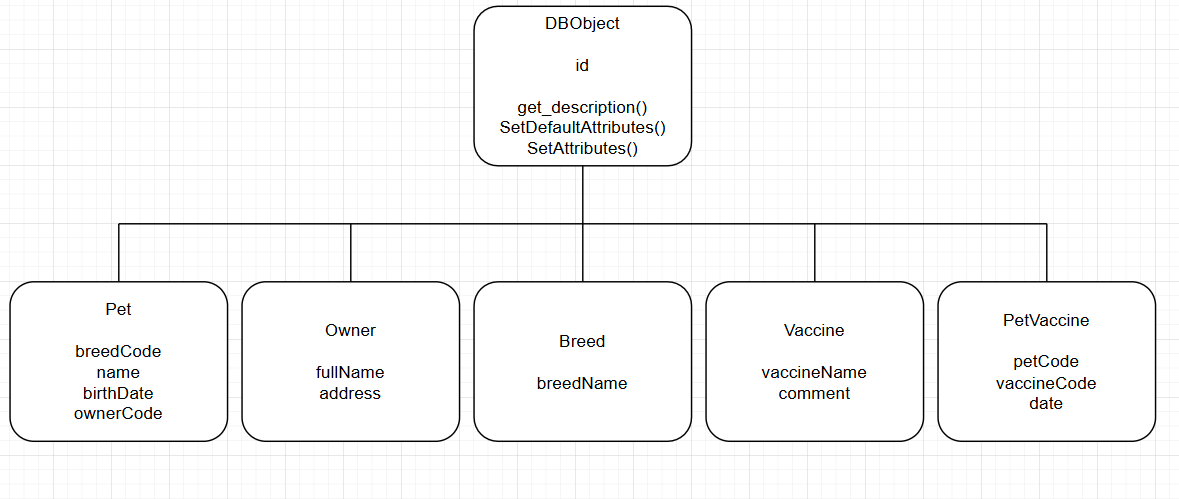


Рисунок 2 – Схема классов

1. Алгоритмы. Псевдокод

Метод **SetDefaultAttributes**

Шаг 1. Инициализируем целочисленную переменную max\_id равной нулю.

Шаг 2. Итерируем по всем элементам таблицы, обновляя значение max\_id максимальным значением поля id

Шаг 3. Обновляем внутреннее поле элемента значением max\_id+1

Шаг 4. Конец

Функция **checkFile**

Шаг 1. Пытаемся открыть файл db.xml, если файл открывается успешно, то заканчиваем выполнение функции

Шаг 2. Создаём файл db.xml

Шаг 3. Создаём 5 корневых элементов в файле db.xml, которые соответствуют пяти таблицам задания

Шаг 4. Конец

1. Контрольный тест. Результаты программы

Результат запуска программы на контрольных данных можно увидеть на рисунке 3.

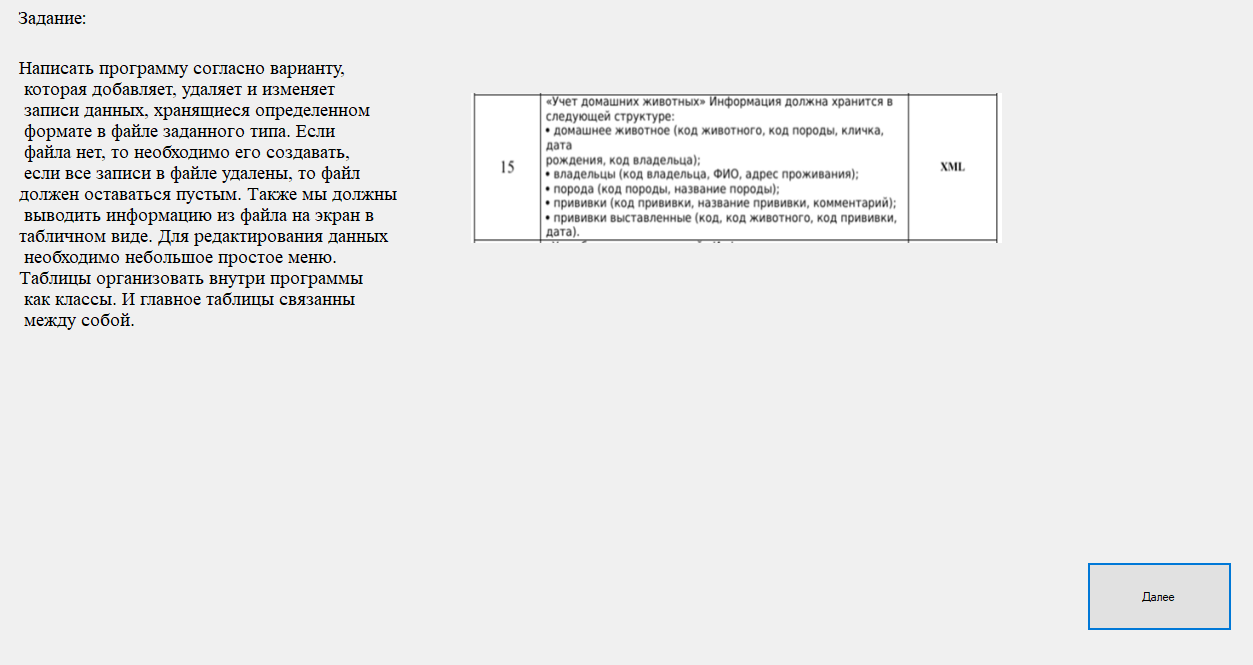


Рисунок 3 – входной экран

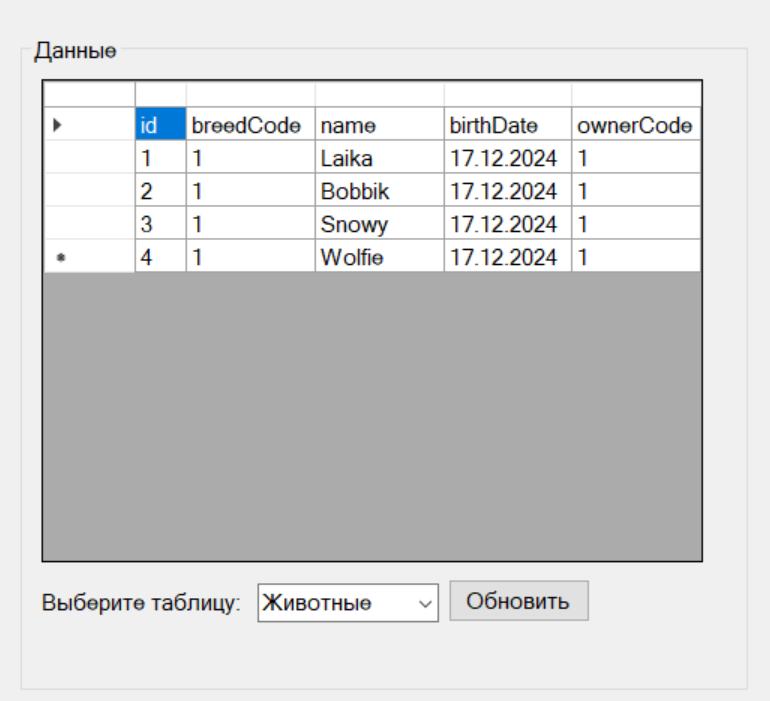


Рисунок 4 – просмотр таблицы

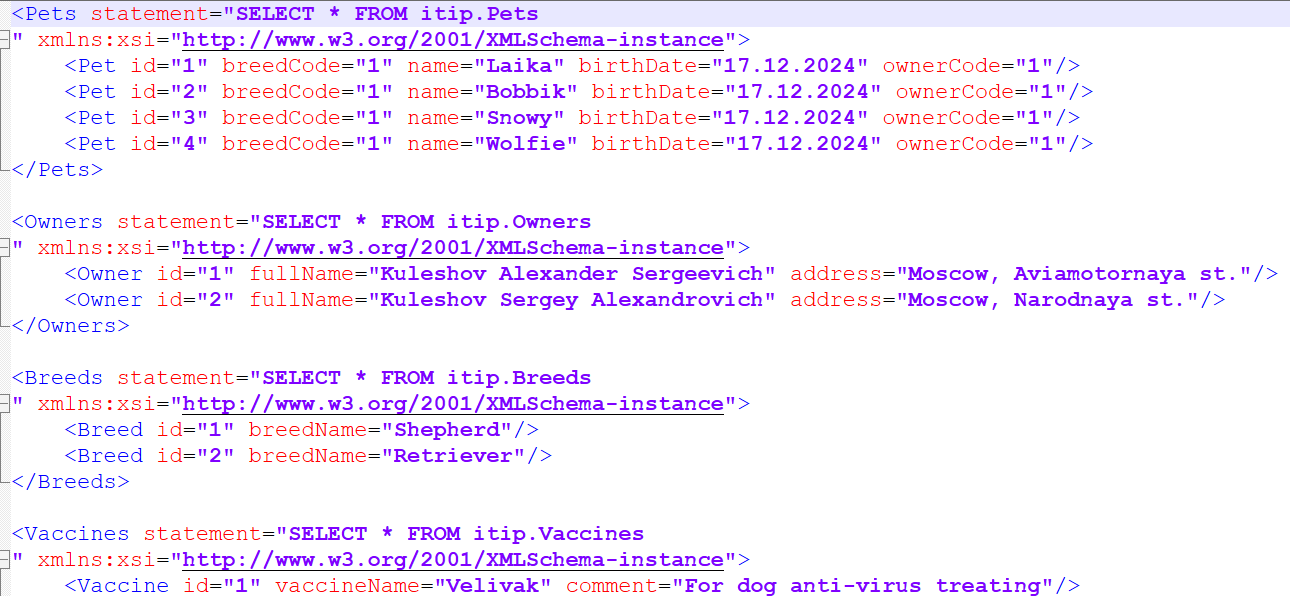


Рисунок 5 – Вид файла

1. Выводы по работе

В процессе работы были изучены механизмы доступа и передачи данных с помощью XML.