БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Кафедра интеллектуальных информационных технологий

**Отчёт**

по лабораторной работе №3

“Приложение для работы с онтологиями”

по дисциплине

“Проектирование баз знаний”

Выполнил студент группы 021702

Локтев К.А.

Проверила

Василевская А.П.

Минск 2023

**Содержание**

1. **Обоснование выбора языка и средств реализации**
   1. Анализ языков для работы с онтологиями
   2. Анализ хранилищ онтологии
   3. Анализ языков для разработки приложения
   4. Анализ средств разработки приложений
   5. Вывод
2. **Онтология**
   1. Описание предметной области
   2. Классы и свойства объектов
   3. Примеры экземпляров
3. **Описание разработанного приложения**
   1. Структура приложения
   2. Хранилище онтологии
   3. Веб-приложение
4. **Вывод**
5. **Источники**

# **Анализ**

## **Анализ языков для работы с онтологиями**

От выбора языка для работы с онтологиями зависит, какой язык программирования мы будем использовать для реализации всей программы. К ним относятся [1]:

* CycL;
* DOGMA;
* Gellish;
* OWL.

**Cycl** является специфичным для Cyc-проектов, а также обладает относительно сложным синтаксисом: например, всегда в начале названий сущостей, отношений, классов и прочего нужно писать $#. Многие языки программирования (например, Java) его не поддерживают[2].

**DOGMA** также используется в основном для проектов Vrije Universitiet Blussel. Из-за этого довольно тяжело найти хорошую документацию и поддержку сообщества. Как и с предыдущим вариантом, этот язык почти не поддерживается популярными языками разработки приложений[2].

**Gellish** обладает своим словарем понятий, однако файлы можно конвертировать в формат, который подходит для OWL. Тем не менее, этот язык малоиспользуем и слабо совместим с языками программирования[2].

В свою очередь, формат **RDF/XML** явялется одним из самых используемых языков в программах и веб-приложениях для работы с онтологиями, таких как, например, WebProtege и Jena. Также этот язык может быть обработан большинством языков программирования, например, Java (Groovy), Python (Owlready2[2]).

Исходя из вышенаписанного, было принято решение использовать формат **RDF/XML.**

## **Анализ хранилищ онтологий**

Существует множество хранилищ, использующих в качестве языка отнологий RDF/XML. Среди них есть как те, которые разворачиваются локально, так и те, которые сущетсвуют в облаке.

Рассмотрим сначала хранилища, которые находятся в облаке. Их очевидное преимущество - отсутствие необходимости нагружать свой компьютер, а также большая база с онотлогиями, но для работы требуется подключение к интернету.

К таким хранилищам относятся[3]:

* Virtuoso;
* MarkLogic;
* Stardog.

**Virtuoso** обладает наиболее быстрой скоростью загрузки RDF-файлов, однако является одним из самых медленных для работы с записями, но при использования языка Sparql. Хранилище можно было бы использовать для программы, но полноценную работу можно получить только по подписке.

**MarkLogic** обладет и относительно медленной скоростью загрузки RDF-файлов, а также тратит много времени на манипуляцию с данными.

**Stardog** обладает сравнительно хорошей скоростью загрузки и манипулирования, но имеет существенный недостаток - полную версию можно получить только по подписке или нужно оформлять пробный период.

Так все вышеописанные варианты были либо слабопроизводительными, либо платными, рассмотрим хранилища, которые можно развернуть локально, на своём компьютере. Использование таких хранилищ не потребует подключения к интернету.

К таким хранилищам относятся[3]:

* Sesame
* Blazegraph
* N3

Хранилище **N3** хорошо тем, что оно позволяет парсить онтологии, описанные в большом количестве форматов (RDF/XML, Turtle, OWL, …), сохранять онтологии в виде текстовых файлов в этих же форматах, а также предлагает быстрый, асинхронный и потоко-ориентированный инструментарий для работы с онтологиями.

**Blazegraph** работает только на Linux, поэтому для его работы понадобилось бы устанавливать виртуальную машину или операционную систему.

**Sesame** нуждается в загрузке Docker, а также не имеет официального Docker Image, что усложняет его настройку и работу с ним.

Таким образом, в качестве хранилища для разработки приложения было решено выбрать хранилище N3.

## **Анализ языков для разработки приложения**

При выборе языка программирования следует ориентироваться как на удобство сосздания веб-приложения для работы с онотологиями (а именно наличие хороших средств и фреймворков), так и на то, есть ли хорошая и удобная библиотека для обработки онотлогий. Рассмотрим следующие языки[4]:

* JavaScript;
* Java;
* Python.

JavaScript является одним из наиболее популярных языков для создания веб-приложений. В совокупности с локальным сервером, написанным с использованием Node.js, имеем объединённый инструментарий, обладающий всем необходимым функционалом для комфортной работы с онтологиями

Что касается языка Java: он неплох для создания веб-приложений, однако библиотека для обработки онтологий обладает скудной документацией.

В языке Python основной и самой популярной библиотекой для обработки онтологий является Owlready2. Она просто в использовании, а также обладает относительно большим сообществом. Также Python сам по себе является довольно простым языком.

Учитывая вышеописанные достоинства и недостатки каждого языка, для разработки приложений был выбран язык JavaScript.

## **Анализ средств разработки приложений**

После выбора языка нужно подобрать фреймворки для создания сайта. Одними из самых популярных модулей для работы с локальными серверами для Node.js являются[5]:

* ExpressJS;
* Fastify;
* EmberJS.

**ExpressJS** - это минималистичный фреймворк для создания веб-приложений на Node.js. Он предоставляет базовый набор функций для обработки запросов и маршрутизации, и позволяет легко добавлять дополнительные функции через плагины.

**Fastify** - это быстрый и эффективный фреймворк для создания веб-приложений на Node.js. Он предоставляет множество функций, включая сжатие данных, валидацию запросов и автоматическую генерацию документации API.

**EmberJS** - это фреймворк для создания одностраничных приложений (SPA). Он использует шаблонизацию и двустороннюю привязку данных для упрощения создания и поддержки сложных приложений. EmberJS также предоставляет множество дополнительных библиотек и инструментов для ускорения разработки.

ExpressJS имеет ряд преимуществ по сравнению с Fastify и EmberJS. Во-первых, он очень прост в использовании и понимании, что позволяет быстро начать создание приложения. Во-вторых, он обладает большим сообществом и экосистемой, что делает его более гибким и расширяемым. В-третьих, он предоставляет ряд дополнительных функций, таких как обработка cookie и сессий, встроенный механизм шаблонизации и многое другое.

Учитывая написанное выше, для разработки приложения был выбран фреймворк ExpressJS.

## **Вывод**

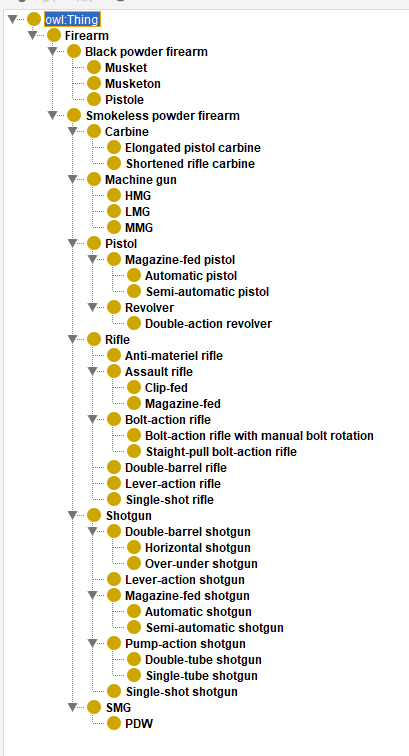
После проведения анализа будем использовать:

* Для работы с онтологиями — язык RDF/XML;
* Хранилище — N3 Store
* Для создания приложения — язык JavaScript и фреймворк ExpressJS.

**Онтология**

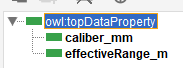
В рамках лабораторной работы была разработана онтология огнестрельного оружия. Назначение онтологии - структуризация и обобщение информации о некоторых экземплярах огнестрельного оружия.

Онтология построена на иерархии классов огнестрельного оружия. Всего было выделено 44 класса.



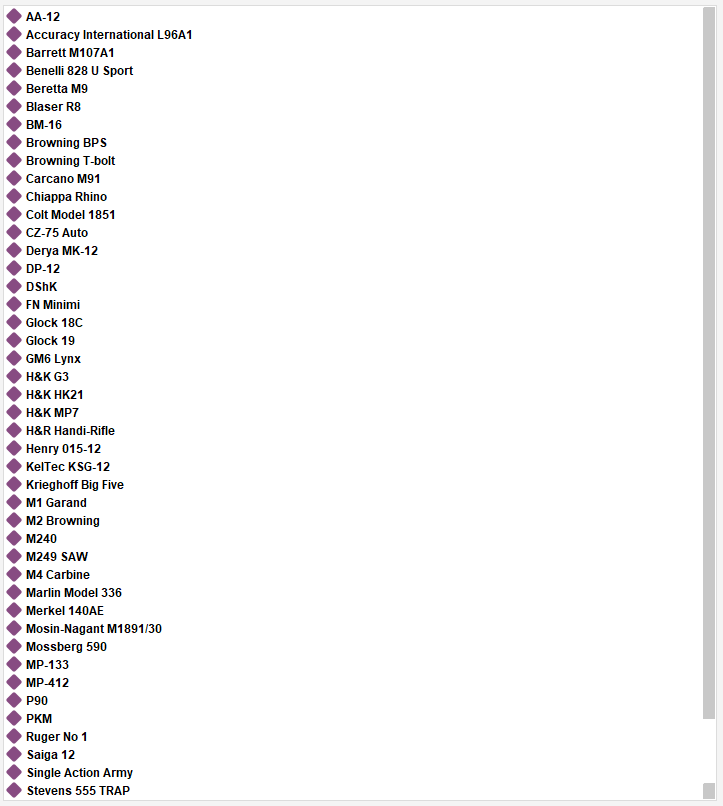
*Рис.1. Классы*

В онтологии выделено два отношения типа Data property: caliber\_mm (калибр) и эффективная effectiveRange\_m (эффективная дистанция).



*Рис.2. Отношения*

Также в онтологии выделено в общей сложности 46 экземпляров (как правило, два экземпляра на каждый оконечный класс). Все экземпляры создавались в том числе с помощью отношений caliber\_mm и effectiveRange\_m.



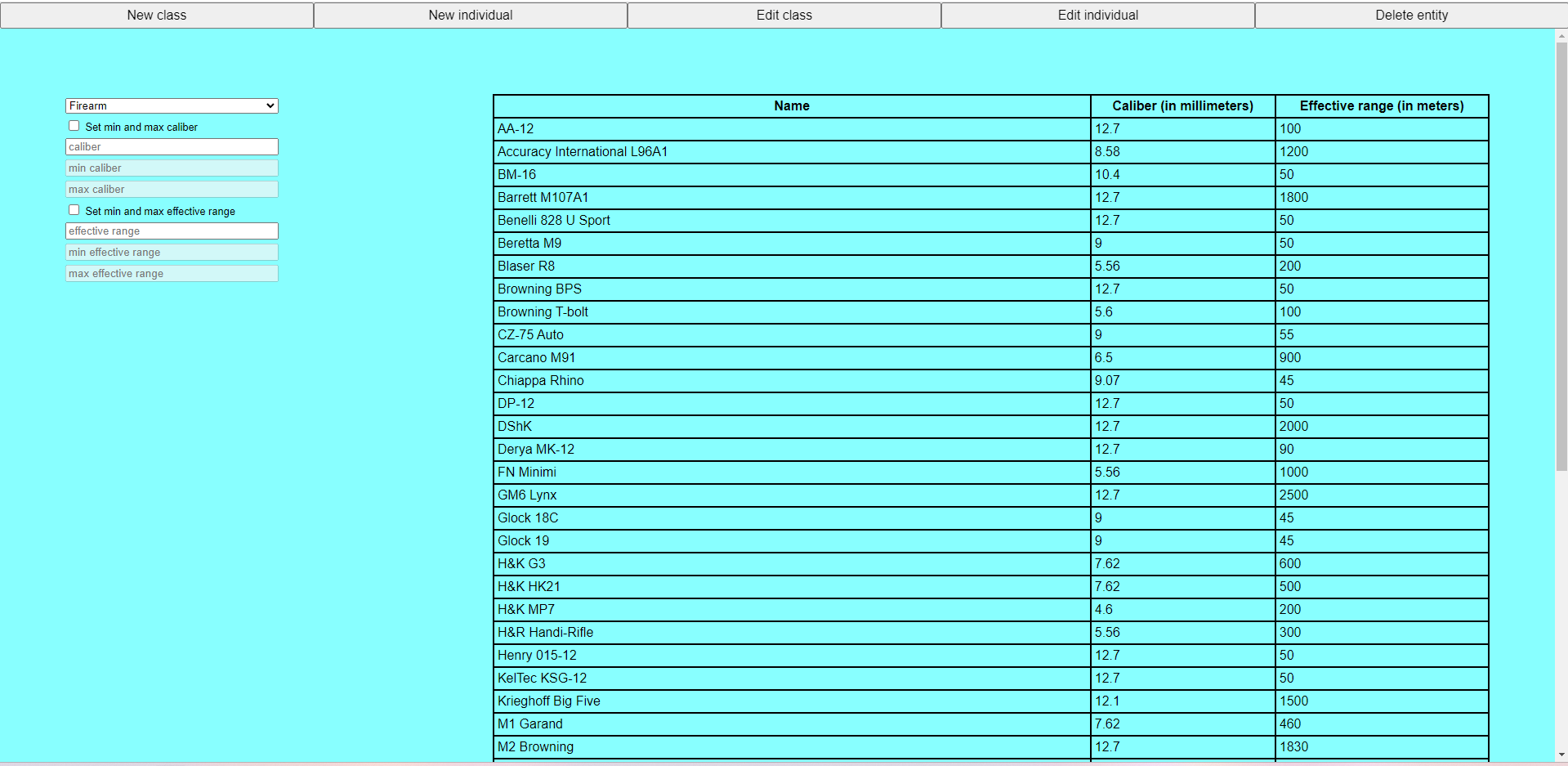
*Рис.3. Экземпляры*

**Описание разработанного приложения**

**Структура приложения**

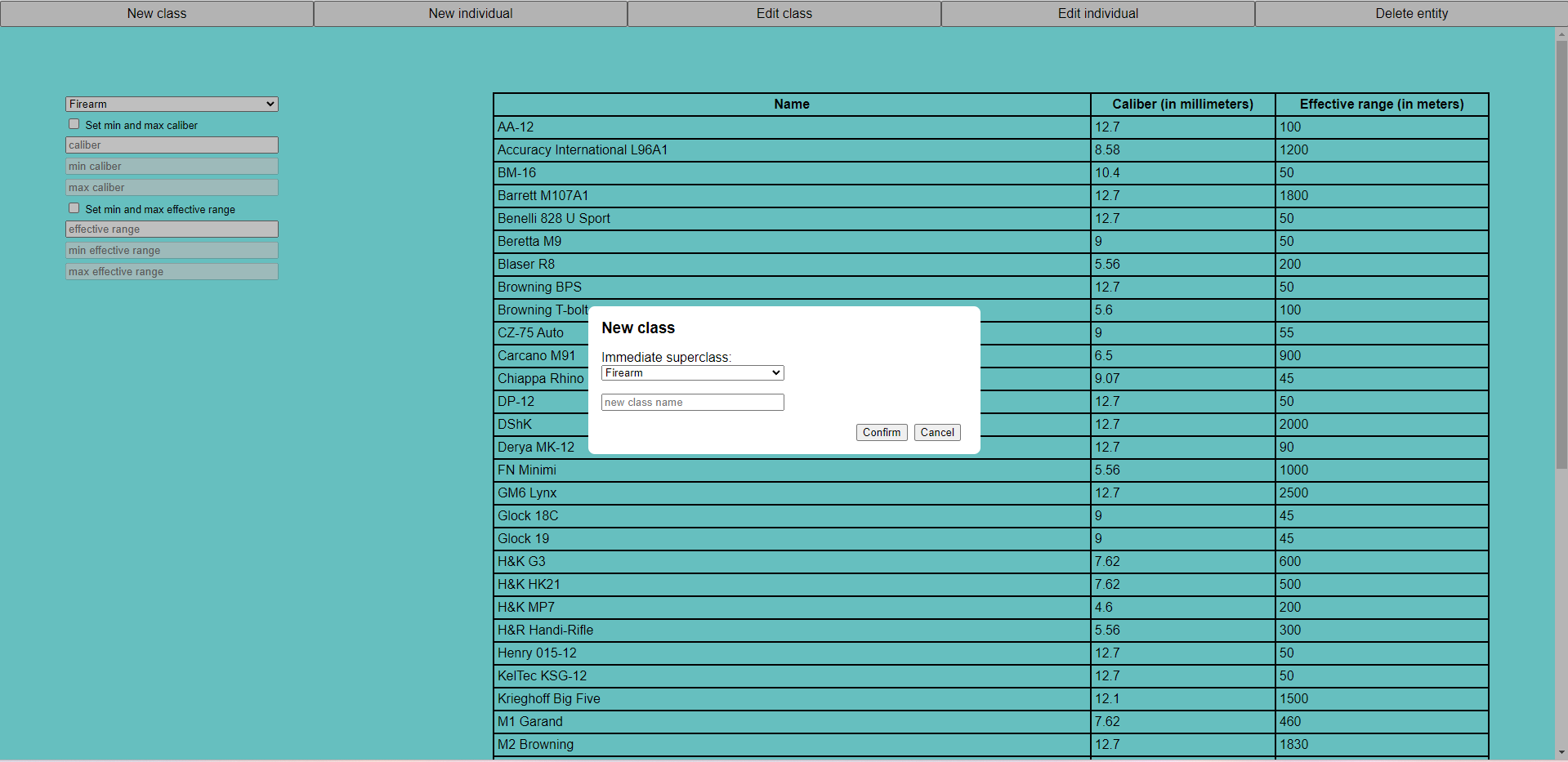
Приложение реализовано с использованием методологии SPA (Single page application). Структура представляет собой классическую связку frontend-backend. Сервер, созданный при помощи фреймворка ExpressJS, получает HTTP-запросы из frontend-составляющей приложения, и в ответ на эти запросы производит соответствующие манипуляции с онтологией, погруженной в хранилище N3.

Интерфейс приложения состоит из меню навигации в верхней части экрана и основной части, позволяющей просматривать онтологию с выставлением желаемых фильтров: по классу, по калибру, по эффективной дистанции, по минимальному/максимальному калибру, по минимальной/максимальной эффективной дистанции.



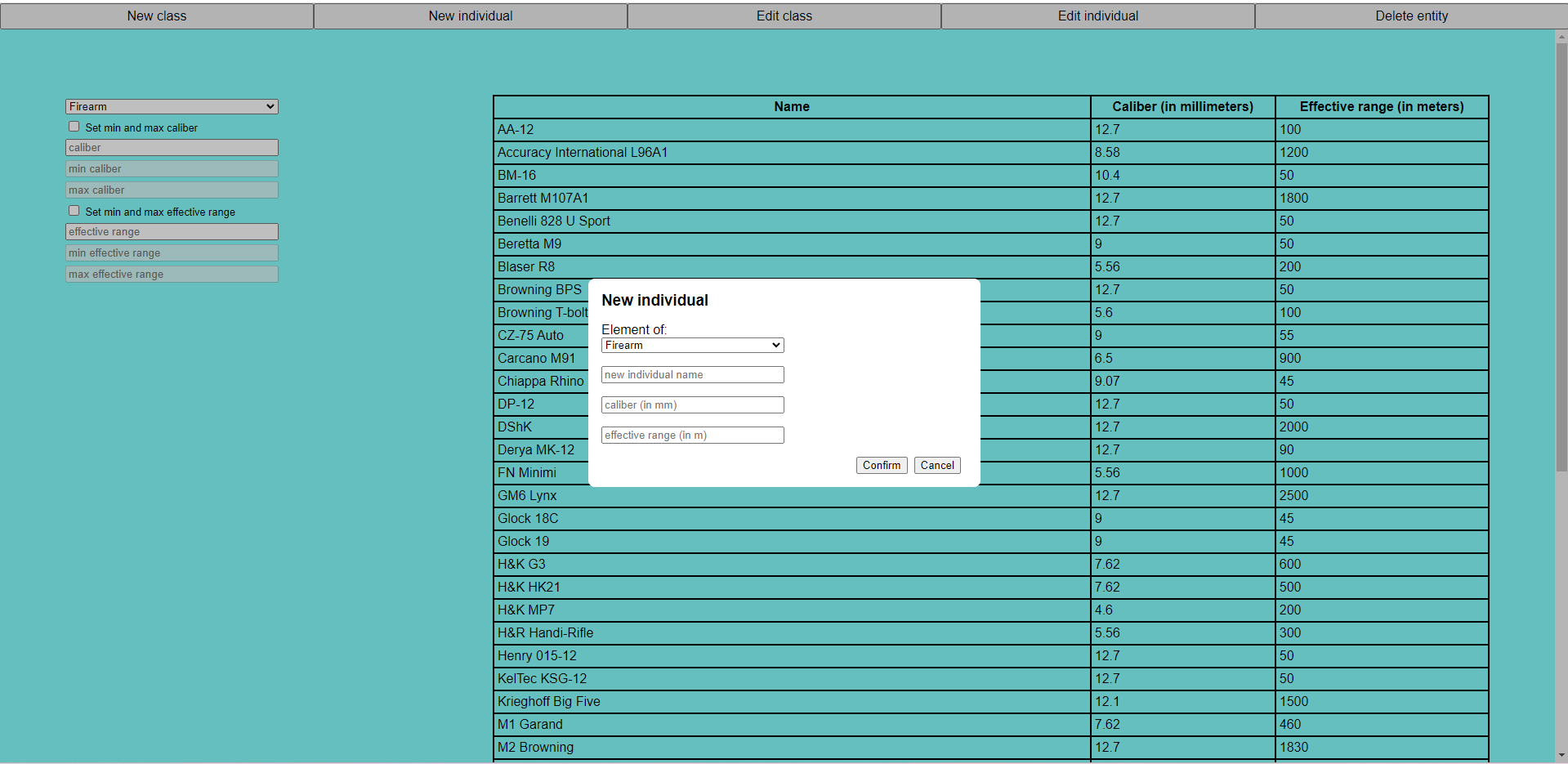
*Рис.4. Интерфейс приложения*

По нажатию на кнопку “New class” открывается модальное окно создания нового класса в онтологии. Нажатие на кнопку “Confirm” после введения разрешённого имени класса создаст новый класс в онтологии. Также здесь можно выбрать, подклассом какого существующего класса будет являться создаваемый класс.



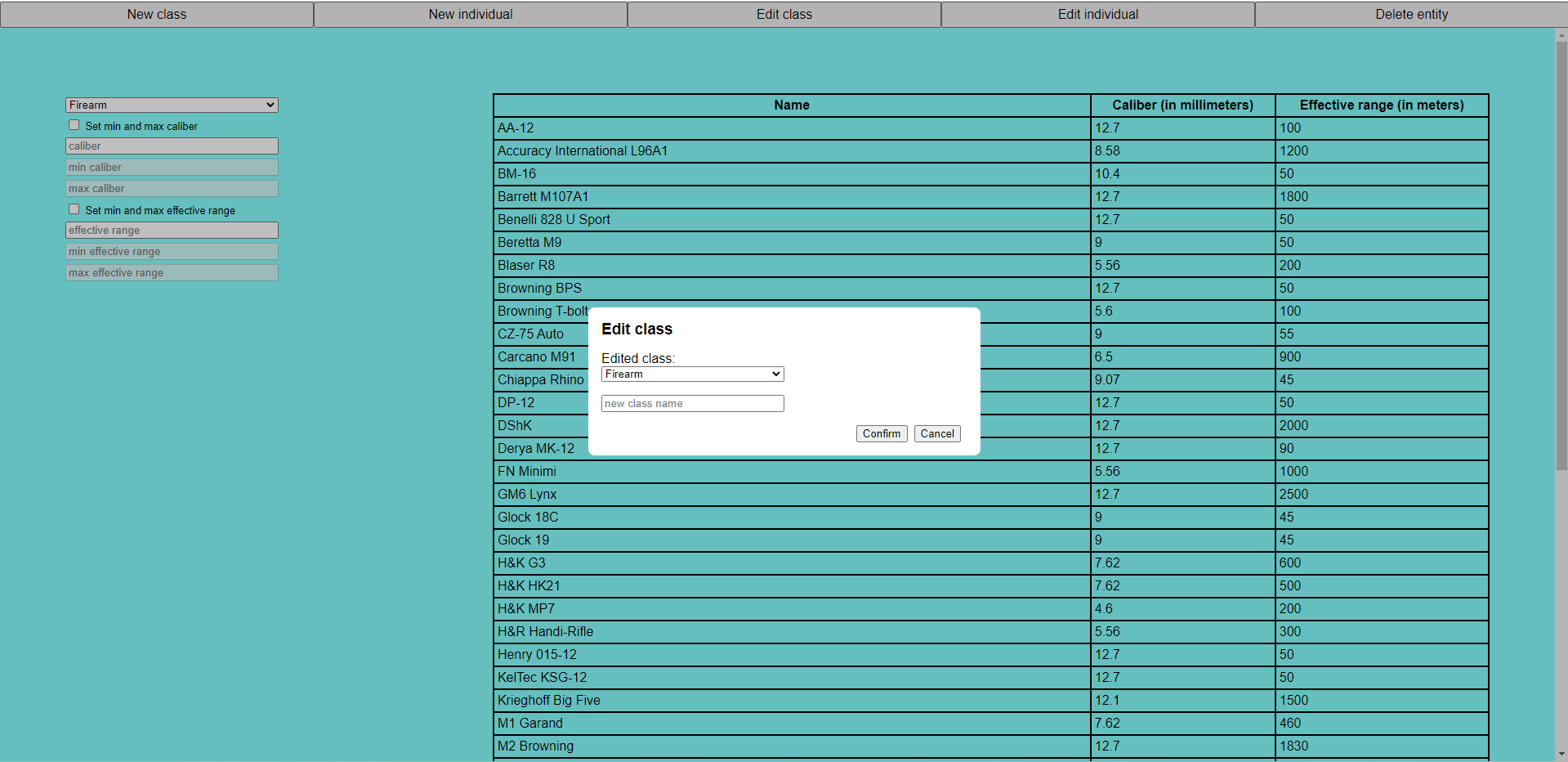
*Рис.5. Модальное окно создания нового класса*

По нажатию на кнопку “New individual” откроется модальное окно создания нового экземпляра в онтологии. Здесь можно ввести имя, калибр и эффективную дистанцию нового экземпляра, а также выбрать, экземпляром какого класса будет являться создаваемый экземпляр.



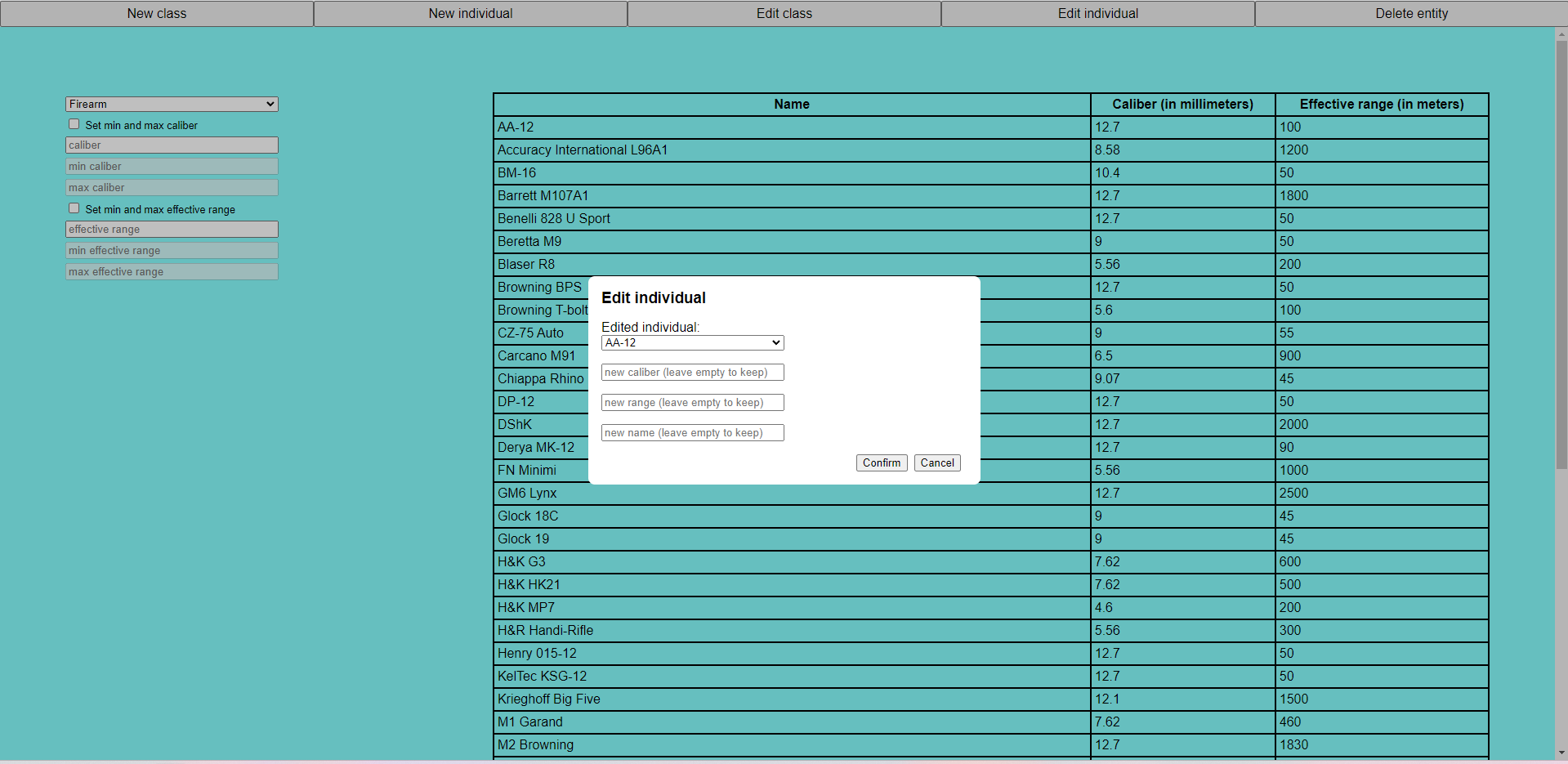
*Рис.6. Модальное окно создания нового экземпляра*

По нажатию на кнопку “Edit class” открывается модальное окно редактирования существующего в онтологии класса. Нажатие на кнопку “Confirm” после введения разрешённого имени класса изменит имя выбранного класса в онтологии на введённое.



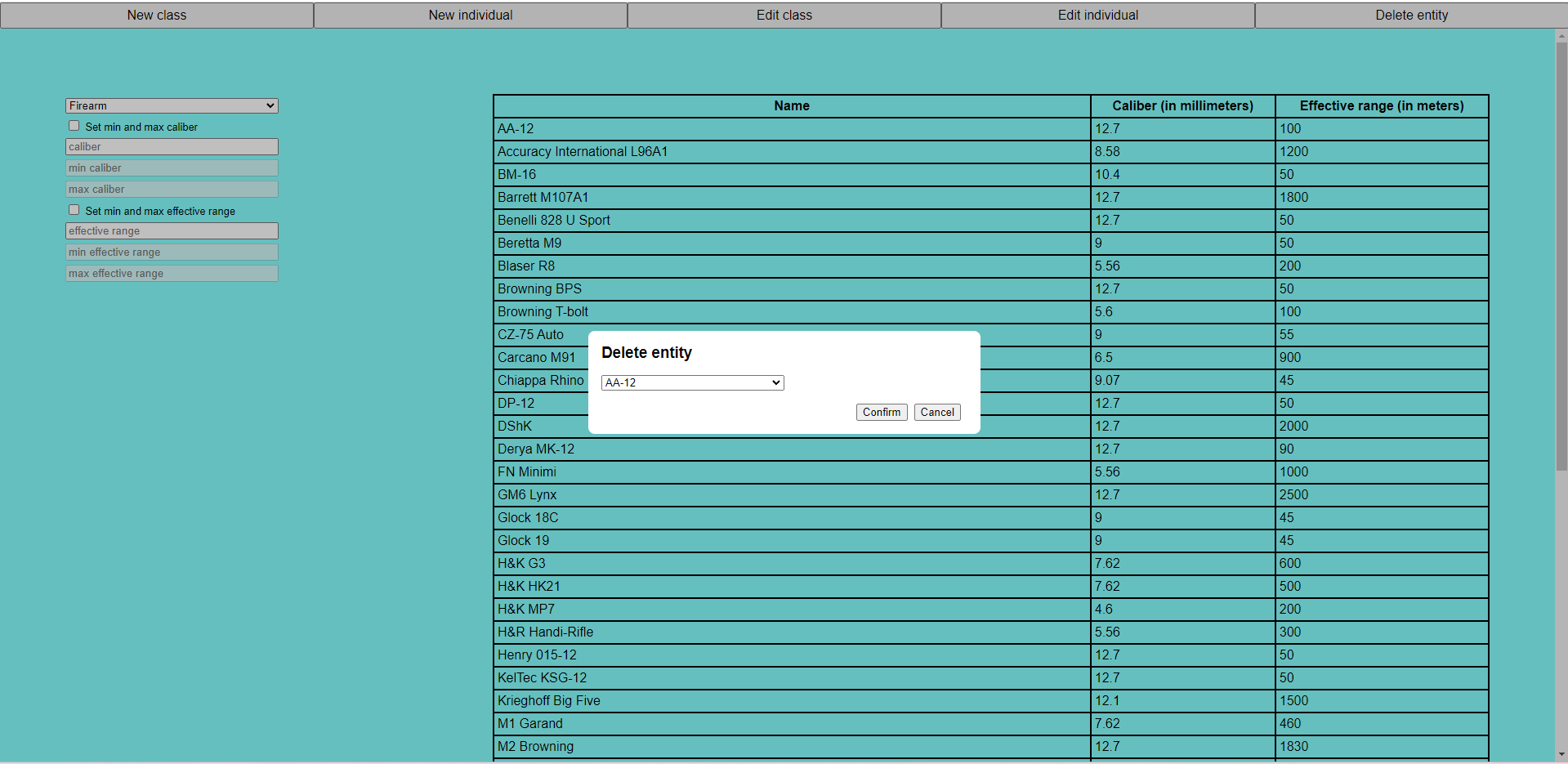
*Рис.7. Модальное окно редактирования класса*

По нажатию на кнопку “Edit individual” откроется модальное окно редактирования существующего в онтологии экземпляра. Здесь можно выбрать имя редактируемого экземпляра, а также ввести новые калибр и эффективную дистанцию нового экземпляра. Любое поле можно оставить пустым, оставить соответствующее значение тем же.



*Рис.8. Модальное окно редактирования экземпляра*

По нажатию на кнопку “Delete entity” откроется модальное окно удаления сущности из онтологии. Нажатие на кнопку “Confirm” после выбора сущности-экземпляра из выпадающего списка удалит соответствующий экземпляр из онтологии. Если же была выбрана сущность-класс, то нажатие на кнопку “Confirm” удалит из онтологии саму сущность и все её подклассы, а также все экземпляры этой сущности и её подклассов.



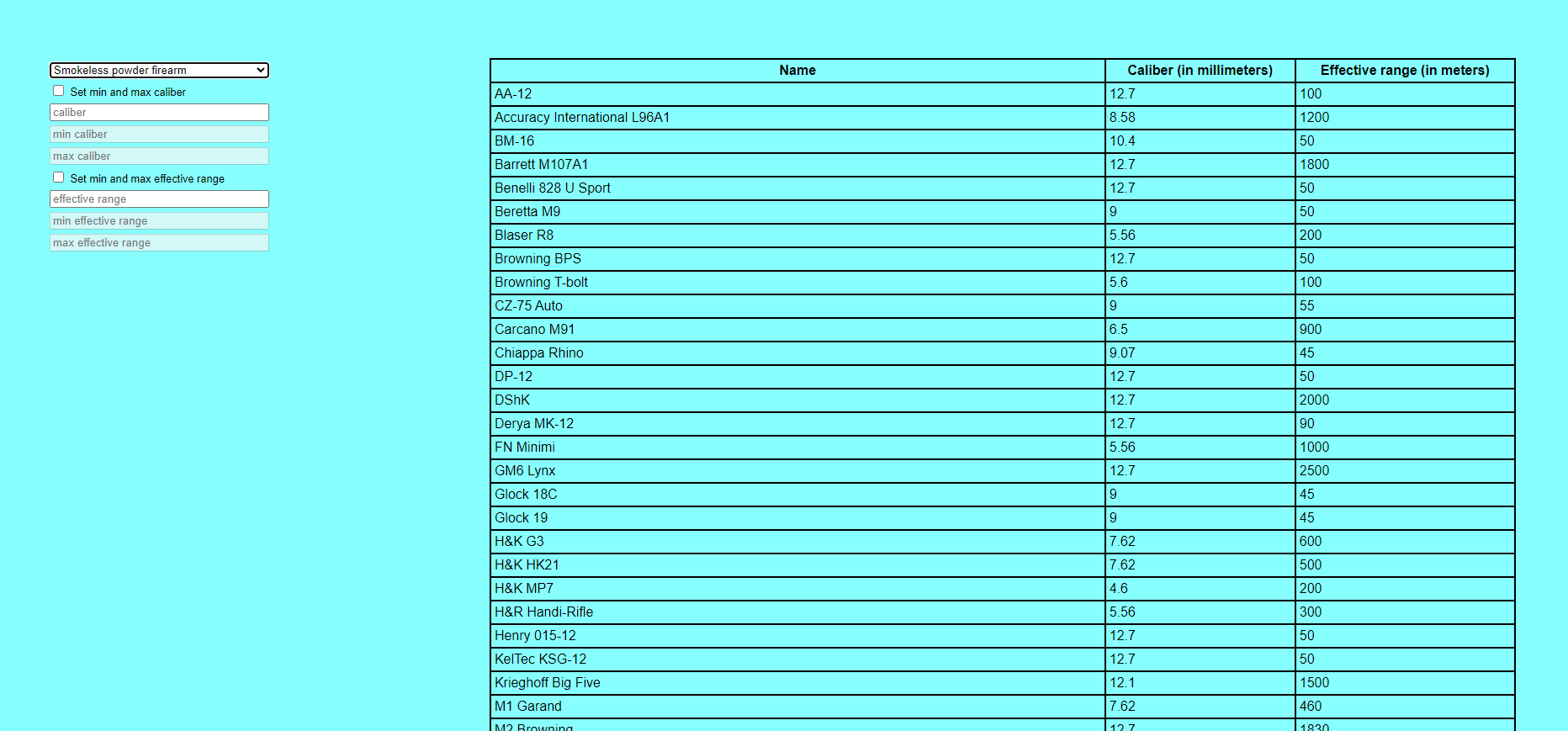
*Рис.9. Модальное окно удаления сущности*

**Описание алгоритма установки приложения**

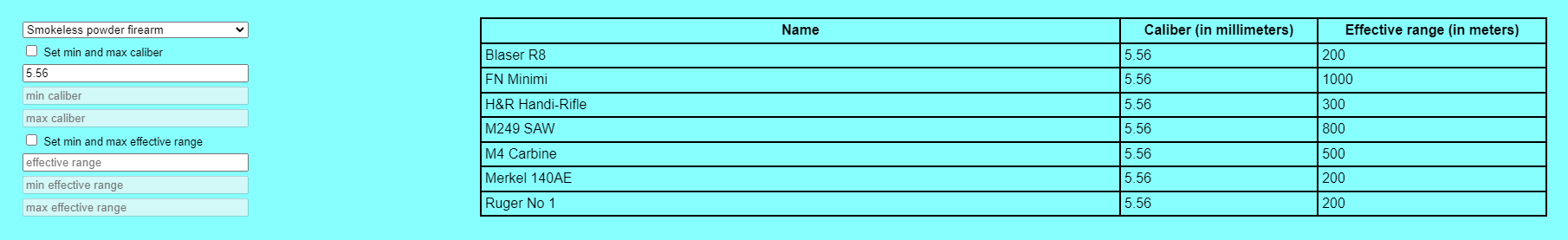
1. Зайти в папку **back**
2. Открыть файл **server.js**
3. Найти строку “**const ontologyURL = `http://localhost:3000/sparql`;**”
4. Изменить URL на необходимый URL SPARQL-эндпоинта. В лабораторной работе использовался локальный SPARQL-эндпоинт, предоставлявший доступ к локальной онтологии
5. Установить node package manager (**npm**), если он ещё не установлен
6. Открыть терминал в папке **back**
7. Ввести команду **npm i**. Эта команда установит модули, необходимые для работы локального ExpressJS-сервера
8. Ввести команду **node server.js**
9. Зайти в папку front
10. Открыть файл **index.html** в любимом браузере

**Примеры запросов к онтологии**

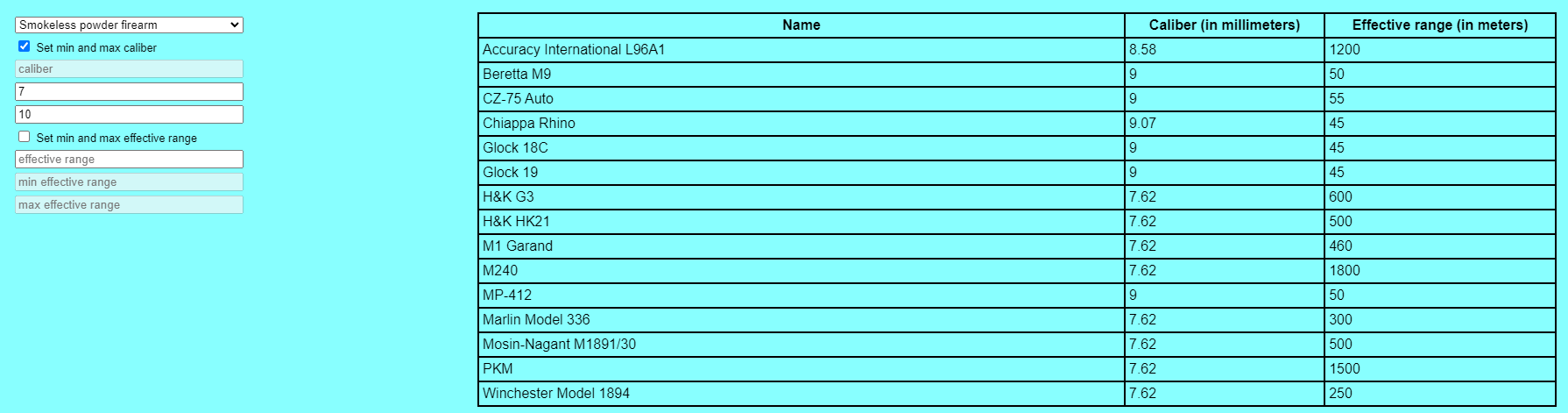
Запрос всех экземпляров огнестрельного оружия на бездымном порохе:



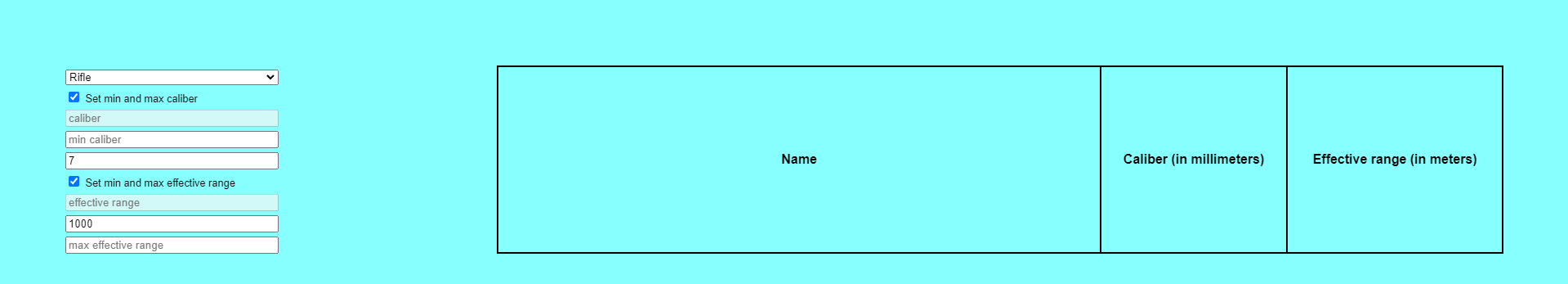
Запрос всех экземпляров огнестрельного оружия на бездымном порохе с минимальным калибром 5.56мм:



Запрос всех экземпляров огнестрельного оружия на бездымном порохе с калибром между 7мм и 10мм:



Запрос всех экземпляров винтовок с максимальным калибром 7мм и минимальной эффективной дальностью 1000м:



**Вывод:** в рамках лабораторной работы было разработано веб-приложение для работы с онтологиями: их просмотра, пополнения, редактирования. Проведён сравнительный анализ форматов описания онтологий, языков программирования, средств реализации. Изучен язык запросов к онтологиям SPARQL.

# **Источники**

[1] ScienceDirect [Электронный ресурс]. — Режим доступа: {<https://www.sciencedirect.com/topics/computer-science/ontology-language>}

[2] ResearchGate [Электронный ресурс]. — Режим доступа: {<https://www.researchgate.net/publication/269801838_Ontology_Languages_-_A_Review>}

[3] Medium [Электронный ресурс]. — Режим доступа:

{<https://medium.com/wallscope/comparing-linked-data-triplestores-ebfac8c3ad4f>}

[4] Ironhack [Электронный ресурс]. — Режим доступа: {<https://www.ironhack.com/us/en/blog/the-5-most-in-demand-machine-learning-languages-in-2022>}

[5] Sweetcode [Электронный ресурс]. — Режим доступа: {<https://sweetcode.io/how-to-choose-a-python-framework-for-web-development/>}