MneMojno - Design and Deployment of a Semantic Web Service and a Mobile Application

Dmitry Zamula, Maxim Kolchin
University ITMO
Saint-Petersburg, Russia
{zamula.dmitry, kolchinmax}@gmail.com

***Abstract*—In this paper we discuss our experience with the design, development and deployment of MneMojno Semantic Web Service and Mobile Application. MneMojno makes use of Semantic Web Technologies to create a web service and a mobile application for rating and providing detail information about food products. This paper introduces the main components of the system: a framework to capture information about food products from various sources, a semantic web service and a framework to visualize and analyze usage statistics. Finally, we discuss challenges and problems we faced during the development and present our conclusions and future directions for exploration in terms of developing MneMojno further.**

***Keywords*—Ontology Engineering, Semantic Web, Ontology Building.**

I. Introduction

In this paper we describe the process of creation and deployment of a mobile application ”MneMojno”, based on semantic technologies. The main issue solved by the application - construction of a relative mark of the food, to simplify the process of selecting the most useful ones. Consider one of the use-case: the user choose a product in the store, scan the barcode on the package and makes an informed choice of a particular product on the specified criteria. It is required to provide a quick obtaining the maximum amount of information about the requested product on the fly, calculate a relative mark and visualize the results to the user.
Initially considered that the advice, received from the application, will be in some way, generalized, as each person has a personal preference and contraindications, but built architecture allows implementing personal assessment factors in the future.
Since the barcode does not provide information about structure of the product, we need to collect, process and provide information about products, in particular, composition, energy value, the presence of GMOs, the presence of food additives (and their origin - natural / synthetic).

A. Similar applications

To date, there are some applications on the market with similar functionality - one of the most famous in the area - ”Fooducate”. The main approach is repeated, as in the described application - user scans barcode and gets some useful information about product. Fooducate also provide additional applications for users with special needs (diabetes, allergies oriented). But, in this field a big part of information required for the analysis of the product, existing applications cover the product basket for only certain countries, and at the moment, none of these applications cover the Russian market.

B. Structure of the Paper

At the Section II the general architecture and main components of the application are presented. Next three sections describe our experience in topics of data retrieval, management of this data and recording of user activity. At the section VI we highlight and list some challenges and problems we faced during the development. And the last section we provide the conclusion.

II. Overview

A. Web Service

The initial idea of the project was to create a mobile application, but with the accumulation of a large number systematic data, it became clear that the database itself can be provided as a service, and therefore, it was decided to provide API as a RESTful web service.
Using the REST API simplifies the development of arbitrary clients (mobile application, web version, a desktop application).
PlayFramework has been chosen to implement the REST API and core module, which brings together all the server modules for the proposed approach ”convention over configuration”, reducing the amount of code, as well as a large community and use JVM as a base (the framework itself supports Java and Scala programming languages). In the future we plan to add semantic support for web service interface, but now, mainly, semantic technologies have been used in the server part of the whole system. It includes several components:

- RDF data-storage, Virtuoso Universal Server;
- Framework for RDF processing - Sesame, which provides interfaces for RDF data storage, besides multiple methods for data manipulation;
- Set of the custom SPARQL-builders, which allows to build dynamic queries
- Tuple Query to JSON serializers

Despite the existence of solutions on the market that implement ORM approach for working with ontologies, in form of the data schema, those solutions were discarded due to the lack of any information on production ready of this approach, and due to issue with difficulty of defining and optimizing the final SPARQL query.

A high-level diagram of the components is shown on Figure 1.

Application flow is as follows:

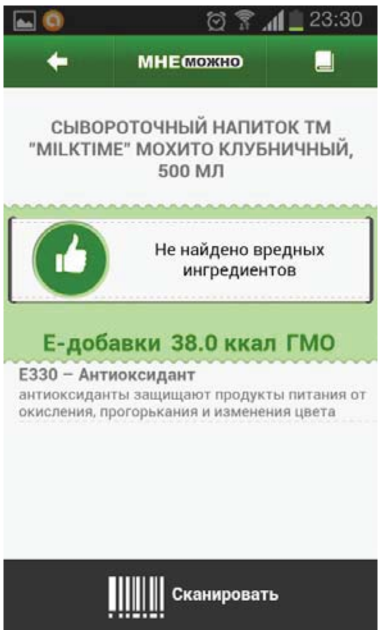
1) The user makes a query by calling a specific API method
2) After the stage of validation and transformation into internal Data Transfer Object from the request parameters, SPARQL query string is constructed
3) Request sends to the Virtuoso with the Sesame API
4) Asynchronously received triplets, serialize into JSON, which is sent to the client

B. Client

In order to develop a client application, we use the HTML5 Single Page approach, expressed in the creation of a single-page Web application, and wrap it in a container that emulates the native application. The main objective of this technique - cross-platform implementation of the application code (business logic and UI part). Phonegap framework, as the container and infrastructure for building mobile application was used. With the lack of computationally complex tasks there were no problems with performance of the application. The only one critical area - the implementation of a barcode scanner was realized using native code parts, which also allowed working directly with the mobile device hardware elements (for camera access).
The main technologies for the development of mobile applications were the HTML and CSS to mark up the user interface and JavaScript to implement business logic. Given the nature of a mobile application, there was a requirement to create a multi-template application. AngularJS, as a JavaScript framework has been selected on the basis of this requirement, because it also support the ability to link data between the dynamic context (business logic), and the presentation layer (UI). Current version of user interface is present on Figure 2

III. Data Retrieval

To implement the functionality of the products' analysis, it is necessary to have a set of data which will be base for a variety of analytical tasks, which forms the final rating of the product, with the ability to track the route to a particular value. Several web resources were selected as a data-provider, which contains product identifier (barcode or name) and one or more types of interest information (composition, the presence of GMO, food value, etc.).
Crawling was used as a method for retrieving information from the web pages, composed in writing programs that parse html code for a given page, and preserving certain blocks of data in text file (csv format). In daily practice, most crawlers are one-time programs, with single cycle of retrieving information. In case of the described application, it's originally intended to deferred supplement the available information which expressed in the creation of schedule software jobs, and updating existing information sources from the set when the data source (in this case, a website), supplemented with new information. Thus, the problem by monitoring updates of existing data was solved,



СЫВОРОТОЧНЫЙ НАПИТОК ТМ "MILKTIME" МОХИТО КЛУБНИЧНЫЙ, 500 МЛ

Не найдено вредных ингредиентов

Е-добавки 38.0 ккал ГМО

Е330 – Антиоксидант
антиоксиданты защищают продукты питания от окисления, прогорькания и изменения цвета

Fig. 2. Mobile user interface

due to the asynchronous-loading of new pieces of information. The conflicts merging process is a semi-automatic ones - in the case of receiving data on any empty field, merge will begin. The merge is doing manually if it's necessary to update existing data.

Data retrieval is implemented by defining CSS selector-like paths to the interest data and processing these paths with JSoup library. The resulting CSV files were used in semi-automatic process of refining with an OpenRefine[1]. Typos were corrected and the numbers of synonyms were reduced using a text facet and clustering to simplify the analysis of the information. This process can also be considered as an enrichment of the data, as in this case we made unification of the product data structure that allows to detect the original ingredients in the multi-level ones (in case, when the final product is composed by a multiple sub-products). After semi-automatic data processing, information was exported to RDF format, using our own ontology.

IV. Data management

As mentioned in the second section, Semantic Web technologies, such as OWL[2] and RDF[3], are used in the application extensively to store and process the data about food products, their ingredients and ratings. Similar to the development of an application using the relational approach to data management where the first step is to build a database schema, ontology for organizing the information was built and called, Food Product Ontology.

A. Food Product Ontology

The main ontology that is extensively used in the system is ontology for describing food products, called Food Product Ontology. The ontology extends the most powerful ontology for describing all of the details of products and services,

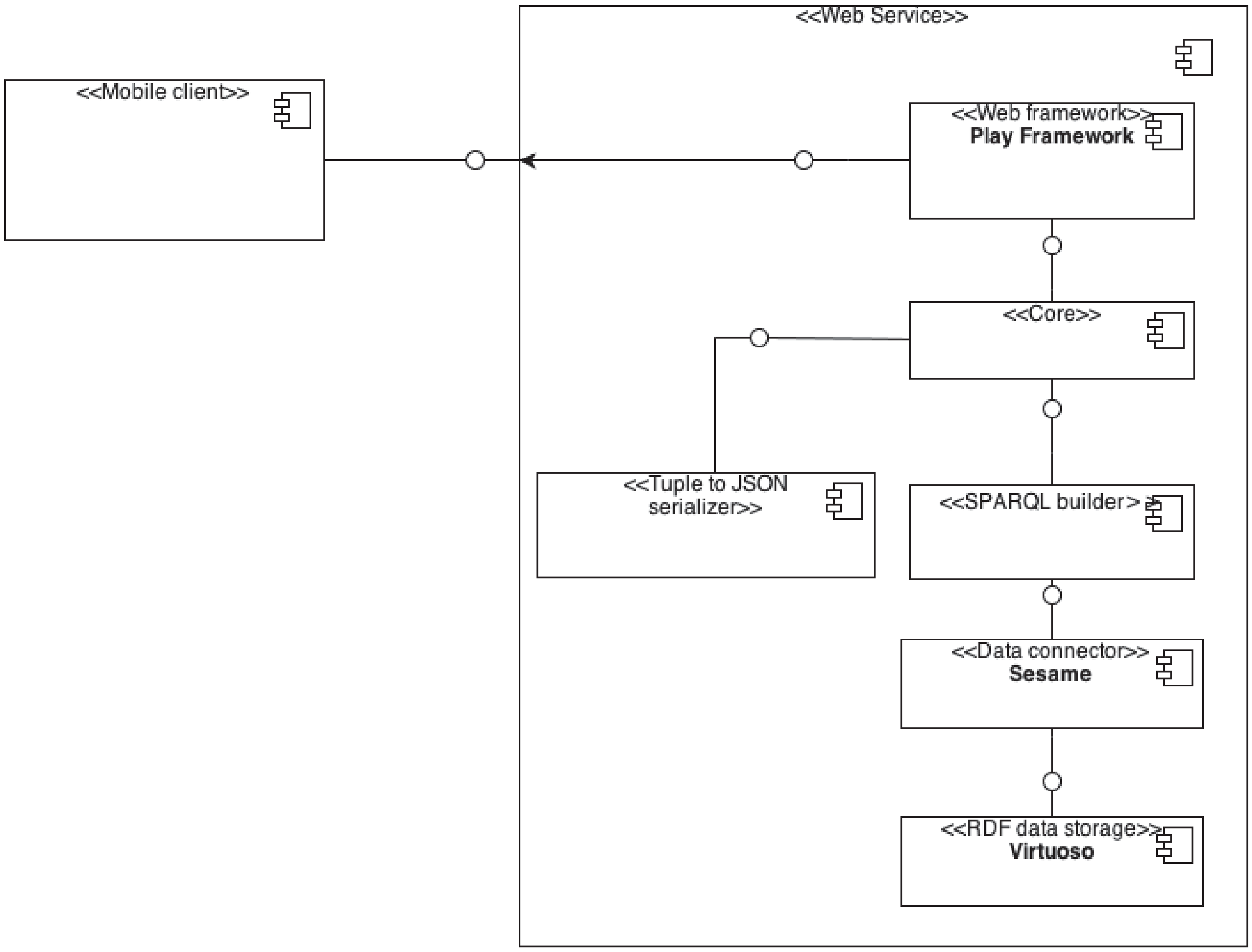


Fig. 1. Component diagram

GoodRelations[4] ontology. GoodRelations is a widely used ontology that is adopted by Google, Yahoo, Bing and other search engines.

A visualization of classes and properties in form of an entity-relationship diagram is shown on Figure 3.

The most important conceptual elements of the domain are as follows:

Product or Service - A class from the core ontology describing an actual product, product makes and models or classes of actual products that are similar in function or nature.

Food - is a concept representing a food product and a subclass of Product or Service concept in GoodRelations, therefore it inherits all needed properties of a product, such as barcode, name, description, weight and others.

Ingredient - An ingredient of a dish or a food product, i.e. a sugar or an artificial flavor. It has several subclasses describing different types of ingredients, such as Food additive and E-additive.

An example of RDF in Turtle[5] syntax describing a food product:

```
@base <http://example.org/foodproducts/> .
@prefix gr: <http://purl.org/goodrelations/v1#> .
@prefix food: <http://purl.org/foodontology#> .

<4600528346794>
 a food:Food .
 gr:name "Cheese Sauce" ;
 gr:hasEAN_UCC-13 "4600528346794" ;
 gr:description "mayonnaise sauce" ;
 food:containsIngredient food:E160a, food:E260 ;
 food:energyPer100gAsDouble "399"^^xsd:double ;
 food:fatPer100gAsDouble "42"^^xsd:double ;
 food:ingredientsListAsText "Water, sunflower oil, ..." ;
```

B. *Tools*

Several tools were used to build the infrastructure and support the management: a tool for editing and visualizing the RDF content, called OntoWiki[6]; a tool for cleaning, discovering and enriching data, called OpenRefine.

V. User Activity Statistics

As for any application, user activity statistics plays important role in determining popular and less popular features of an application. dAquin Mathieu et al showed in their paper[7]

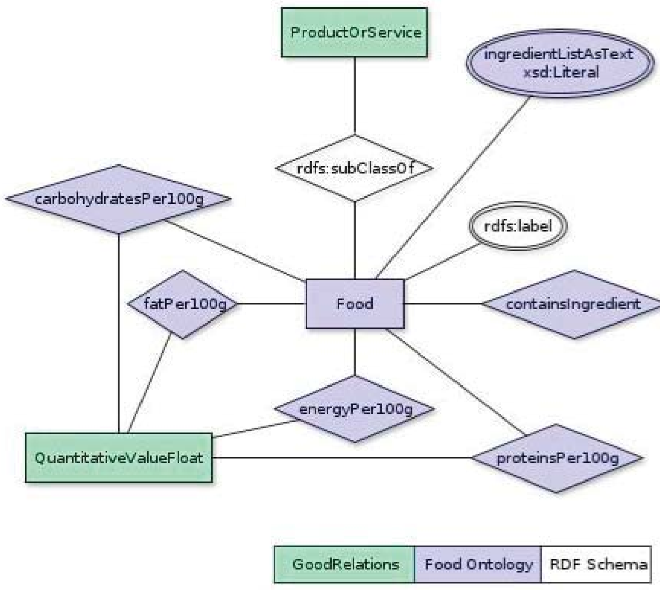


Fig. 3. Conceptual Model of the Food Ontology

that semantic technologies can provide adequate solutions for the following problems:

Fragmentation and heterogeneity - user activity data usually stored in log files that can have different formats, and might not be easily integrated between different components,

User identification - identifying a user within the data is typically a problem faced by any activity analysis,

Data analysis - the data usually are in raw, uninterpreted logs from which is difficult extract meaningful information,

Scale - Tracking user activities through logs can generate immense amounts of data. Typical systems cope with such scale through aggregating data based on clusters of users. Here, we need to keep the whole set of data for each individual user available to provide meaningful analysis of their interaction with the organization in a user-centric way.

For MneMojno project, Atom Activity Streams[8] is used, since this specification is widely used by many companies and projects, and definitely de-facto standard for recording user activity. This specification has a draft implementation of ontology that was developed for NoTude project (EU FP 7), called Atom Activity Streams RDF mapping[9] which we extended to add concepts needed at the project and make it more closely to the specification.

A. Activity Ontology

The activity ontology is an extension of the Atom Activity RDF mapping ontology to add several concepts which the original ontology doesnt have:
Mobile Application - any application which is run on a mobile device. The concept is a subclass of Actor and has a property for describing UUID of the mobile device,
Product - any product such as a food product. The concept is a subclass of Object,
Read - is a verb that indicates that the actor read the object. Its a subclass of Verb.

An example of RDF in Turtle syntax representing a user activity of requesting information about food product which barcode is 4600528346794 from mobile device which UUID is 550e-e29b-41d4-a716-446655440000 at September 29th in 3:33 pm:

```
@base <http://example.org/> .
@prefix easo: <http://mnemojno.ru/ontologies/activity#> .
@prefix aair: <http://xmlns.notu.be/aair#> .
@prefix food: <http://purl.org/foodontology#> .

</foodproducts/4600528346794>
    a food:Food ;
    a easo:Product .

</actors/550e-e29b-41d4-a716-446655440000>
    a aair:Actor ;
    easo:hasUUID 550e-e29b-41d4-a716-446655440000 .

</activities/1380486801>
    a aair:Activity ;
    aair:activityObject </foodproducts/4600528346794> ;
    aair:activityActor
          </actors/550e-e29b-41d4-a716-446655440000> ;
    easo:activityVerb easo:Read .
    aair:activityContext
         [ a aair:Time; aair:date 1380486801 . ]
```

VI. Discussion

At this section we want to highlight some challenges and problems we faced during the development of the application using semantic technologies:

- Absence of production-ready frameworks for querying RDF storage with ORM-like API.
- Poor set of tools for refining stage of data processing.
- Requirement of integrated tool for data management (with possibility of editing ontology and data together).

The problem we've faced during the development was an absence of production-ready frameworks for managing RDF data with API in ORM style. A high-level diagram of the object mapper is shown on Figure 4.

Currently, we need to manual construct of a SPARQL query in every process of data managing. Also, with this approach, there is a requirement to implement data-transformation layer between RDF and object oriented data form.

There was a project with ORM-like approach to RDF, named Sommer, which implement annotation-way mapping with following syntax:

```
@rdf(foaf + "Person")
public class Person {
  public static final String foaf =
  "http://xmlns.com/foaf/0.1/";
  @rdf(foaf + "name") private Collection<URI> names;
}
```

This project was abandoned, and last modification was about 5 years ago which prohibits its use in production. Another abandoned project with required functionality is "jenabean", which try to implement Java Persistence API:

```
public class Car {
        @Id
        private String id;
        private int value;
        public int getValue() {return value;}
        public void setValue(int i) {value = i;}
        public String getId() {return id;}
        public void setId(String id) {this.id = id;}
}

@prefix : <http://my.bean/> .

<http://example.com/javaclass>
      a       owl:AnnotationProperty .

<http://my.bean/Car/1>
      a       :Car ;
      :id     "1"^^xsd:string ;
      :value  "444"^^xsd:int .

:Car
      a       rdfs:Class ;
      <http://thewebsemantic.com/javaclass>
              "my.bean.Car" .

:id   a       rdf:Property .

:value
      a       rdf:Property .
```

There is a growing project "Empire" with required API. It provides a standard JPA style interface to RDF databases using SPARQL with custom annotations:

```
@Namespaces({"frbr", "http://vocab.org/frbr/core#",
      "dc",   "http://purl.org/dc/terms/",
 "foaf", "http://xmlns.com/foaf/0.1/"})
@RdfsClass("frbr:Expression")
@Entity
public class Book implements SupportsRdfId {
private SupportsRdfId mIdSupport = new SupportsRdfIdImpl();

@RdfProperty("dc:title")
private String mTitle;

@RdfProperty("dc:publisher")
private String mPublisher;

@RdfProperty("dc:issued")
private Date mIssued;

@RdfProperty("foaf:primarySubjectOf")
private URI mPrimarySubjectOf;

@OneToMany(fetch = FetchType.LAZY,
   cascade = {CascadeType.PERSIST, CascadeType.MERGE})
@RdfProperty("frbr:embodiment")
private Collection<Manifestation> mEmbodiments =
                               new HashSet<Manifestation>();
}
```

The project was started in 2011, but it is still in the early development stage. It is not support JPA Query Language (JPQL), so it's useless in case of complex SPARQL query.

At the stage of data refining, we faced with a poor set of tools, which can be used. There is one 'leading' tool in this field - OpenRefine (GoogleRefine in the past) with appropriate license (BSD) for our project. It covers some part of refine process:

- Importing
- Filtering / faceting
- Editing cells by Clustering
- Using regular expressions
- Exporting to RDF

The main issue with this tool is performance. OpenRefine works in browser, and with data set larger then about 5000 rows, the user interface become unusable. So, we needed to break our main data set into smallest parts, and work with these parts separately.

After completing data refine process we've managed with ontology part. In enterprise world with relation databases there are multiple handy database clients exist. In these applications we can manage database schema and also work with data (insert, find, modify, delete). It's quite uncomfortable in semantic world to work with ontologies and data separately. There are not such tools, which combine data schema (ontology in our case) management and RDF database client functionality.

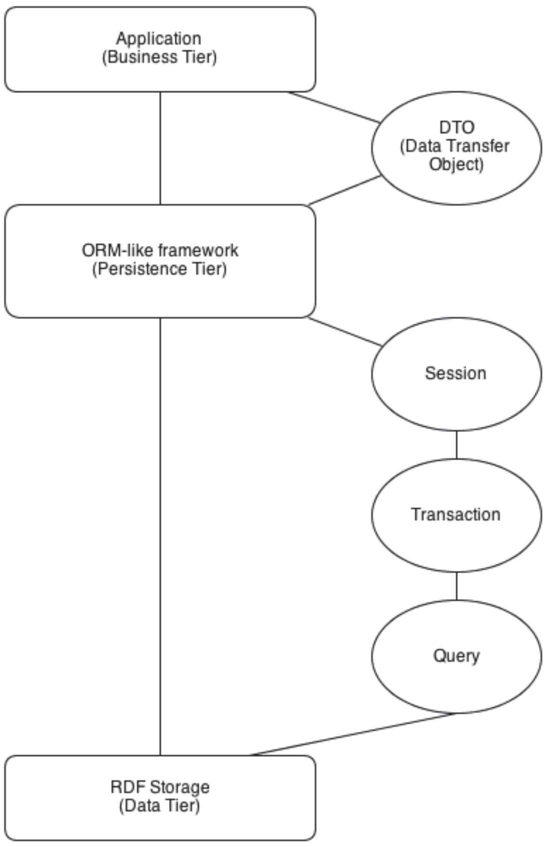


Fig. 4. Object mapper diagram

VII. Conclusion & Future Work

In this paper we described the process of MneMojno application development, which use semantic technologies in core. We described the main principles of semantic development with our architecture details, clarify the development phases and appropriate tools. The current stage of the project allows us to conclude about a success of the application, based on semantic-technology approach with existing technologies for building end-user services. Each day, we process hundreds of user requests, and plan to increase this number. As a future work, we plan to improve infrastructure part of our project - there is a multiple way to improve existing RDF storage clients with target to use them by non-developer team member.

References

[1] D. Huynh and S. Mazzocchi. ”OpenRefine”, http://openrefine.org.

[2] Hitzler, Pascal, et al., ”OWL 2 Web Ontology Language: Primer”, *W3C Recommendation 2 October 2009, 2009.*, Harlow, England: Addison-Wesley, 1999.

[3] Dave Beckett, et al. ”RDF/XML Syntax Specification (Revised)” *W3C Recommendation 10 February 2004*, 2004.”

[4] Hepp, Martin. ”Goodrelations: An ontology for describing products and services offers on the web.” *Knowledge Engineering: Practice and Patterns. Springer Berlin Heidelberg*, 2008. 329-346.

[5] Beckett, David, Tim Berners-Lee, Eric Prud’hommeaux and Gavin Carothers. ”Terse RDF Triple language.” *W3C Candidate Recommendation*, 2013; http://www.w3.org/TR/turtle.

[6] Tramp, S., Frischmuth, P., Heino, N., ”OntoWiki – a Semantic Data Wiki Enabling the Collaborative Creation and (Linked Data) Publication of RDF Knowledge Bases” *In O. Corcho, J. Voelker (eds.), Demo Proceedings of the EKAW 2010*, 2010.

[7] d’Aquin, Mathieu, Salman Elahi, and Enrico Motta. ”Semantic technologies to support the user-centric analysis of activity data.”, *Social Data on the Web: Workshop at the 10th International Semantic Web Conference*, 2011.

[8] Atom Activity Streams 1.0, http://activitystrea.ms/specs/atom/1.0/.

[9] Atom Activity Streams RDF mapping, http://xmlns.notu.be/aair/.