PhoS

- Phobia Smart Assistant -

Team: Tiba I. Georgiana Isabela

Tiba P. Bianca Madalina

Table of contents

1. Introduction

PhoS (Phobia Smart Assistant) is an application designed for people suffering of different phobias. It helps them to discover dangerous contexts that might interfere with their fears, offering in the same time possibilities to see treatment suggestions and tips to avoid them. They can also read more information about any phobia, including their meaning, origin and causes. If a user finds himself in a phobia's characteristics, he can add it to his profile's phobias list. If he considers that a certain phobia is treated, he can remove it from his list.

The users can connect to each other and can use the others' profiles information to avoid situations like visiting a place together, watching a movie or practicing a sport, that can be discomfortable for their phobias.

For building PhoS, we have chosen common, well known and innovative technologies in order to make it flexible, extensible and easy to maintain.

The data and data relations in this application are built based on RDF data and ontologies within SPARQL queries on the web data.

2. Technologies

2.1. HTML 5

HTML 5 is a markup language used for structuring and presenting content on the World Wide Web. In this application, HTML5 is used for the user interface part, to offer a dynamic interactive communication.

2.2. Bootstrap

Bootstrap is a framework used for developing responsive web applications. It is a combination of HTML, CSS, and Javascript code designed for the user interface components.

2.3. Javascript

Javascript is most commonly used as a client side scripting language. For Phos, its main use is to interpret and manage RDF-related data. With Javascript, the RDF data will be parsed, queried and processed.

2.4. Java J2EE

Java is a high level, object-oriented, platform independent language. Java Enterprise Edition is a platform that offers an API that allows the developer to create complex applications with information stored in databases and also, most important for PhoS application, getting and managing user data.

2.5. Apache Jena. Fuseki

Apache Jena is an open source Java framework used for building Semantic Web and Linked Data applications. This framework offers the developer the possibility to use Fuseki, a SPARQL standalone server used to query the rdf data. In this application is used in this application to extract data and to write them to RDF graphs.

2.6. SPARQL

SPARQL is a query language intended to be used to retrieve and manipulate data stored in RDF format. In PhoS development process, it is used in Fuseki server requests for getting and manipulating data from Dbpedia, Schema.org, FOAF and its own ontology.

2.7. API

In order to create HTTP requests between PhoS clients and the server-side, it is used an API developed in Java which allows retrieving data about users, phobias and user profile. The endpoints used for this communication are:

/phobias

The Phobias endpoint returns a list of phobias. The response includes the display name, a long description, resource URI, the title and a photo if exists.

/phobias/add/{phobiaResource}

This endpoint allows a user to add a new phobia (given as parameter) to his list of phobias.

/phobias/remove/{phobiaResource}

The Remove endpoint allows a user to remove a phobia (given as parameter) from his profile list of phobias.

/phobias/myPhobias

The myPhobias endpoint returns the list of phobias that the user added on his profile.

/getActvities

The getActivities endpoint returns a list of predefined activities. The response includes the type for each activity.

/getSubactivities

The getSubactivity endpoint returns a list of subactivities. The response includes the name of the subactivity and the type for each activity.

/findPhobias

The findPhobias endpoint allows a user to check the safety of the activity he wants to do and returns the participants's phobias that might interfere with the action.

/context/getTreatment

The getTreatment endpoint allows a user to get a list of linked treatments and tips related to the selected phobia.

/user/friends

The friends endpoint returns a list of people. The response includes the display name, a photo url and an email for each person.

/user/addFriend

The addFrieds endpoint allows a user to add a new person to his friends list.

/user/saveUserData

The saveUserData endpoint updates the user profile information with the birthdate, gender, animals and if the user has children.

/user/userDetails

The UserDetails endpoint returns information (photo, name, email) about the user that has authorized with the application.

/user/getUserData

The getUserData endpoint returns the user's profile data: gender, if the user has children and the user's animals.

/user/getAnimals

The getAnimals endpoint returns a list of pets that a user can have.

/login

The Login endpoint allows an user to log in.

/logout

The Logout endpoint allows an user to log out.

2.9. Protégé

Protégé is an editor and framework for creating and updating custom ontology, and validating the models to infer the information structured on the ontology rules. The Protégé platform supports two main ways of modeling ontologies via the Protégé-Frames and Protégé-OWL editors. Protégé ontologies can be exported into a variety of formats including RDF, RDFS, OWL, and XML Schema.

2.10. FastJSON

FastJSON is Java library for allowing developers to work with JSON data, like sending and sharing data with the client within request (GET, POST). JSON is a way to store human-readable collection of data that is organized in a logical manner.

2.11.JQuery

JQuery is a JavaScript library that helps the developers on HTML document traversing, event handling, animating, and Ajax interactions for rapid web development. In this application its main role is for internationalizing the messages.

3. Data Sources and Vocabularies

To retrieve various information about phobias and bindings between them, PhoS application uses Wikipedia based data sources described below:

3.1. DBpedia.

DBpedia is a project that allows users to semantically query relationships and properties associated with Wikipedia resources, including links to other related datasets. The language used for queries is SPARQL, a SQL-like query language for RDF (Resource Description Framework).

3.2. Schema.org

Schema.org is the centralized home on the web for the Schema project which contains microdata that makes it easier for searching engines to parse and interpret information.

3.3. FOAF

FOAF is a project that allows users to semantically associate and query relations between person and other concepts/entities using the web. FOAF integrates three kinds of network: social networks of human collaboration, friendship and association; representational networks.

3.4 Google API

Google API is a service built for developers in order to allow them to use Google as a resource in their software application within query over web documents. In PhoS application the Google API is used to get treatments and tips linked sources for any phobia.

4. PhoS ontology

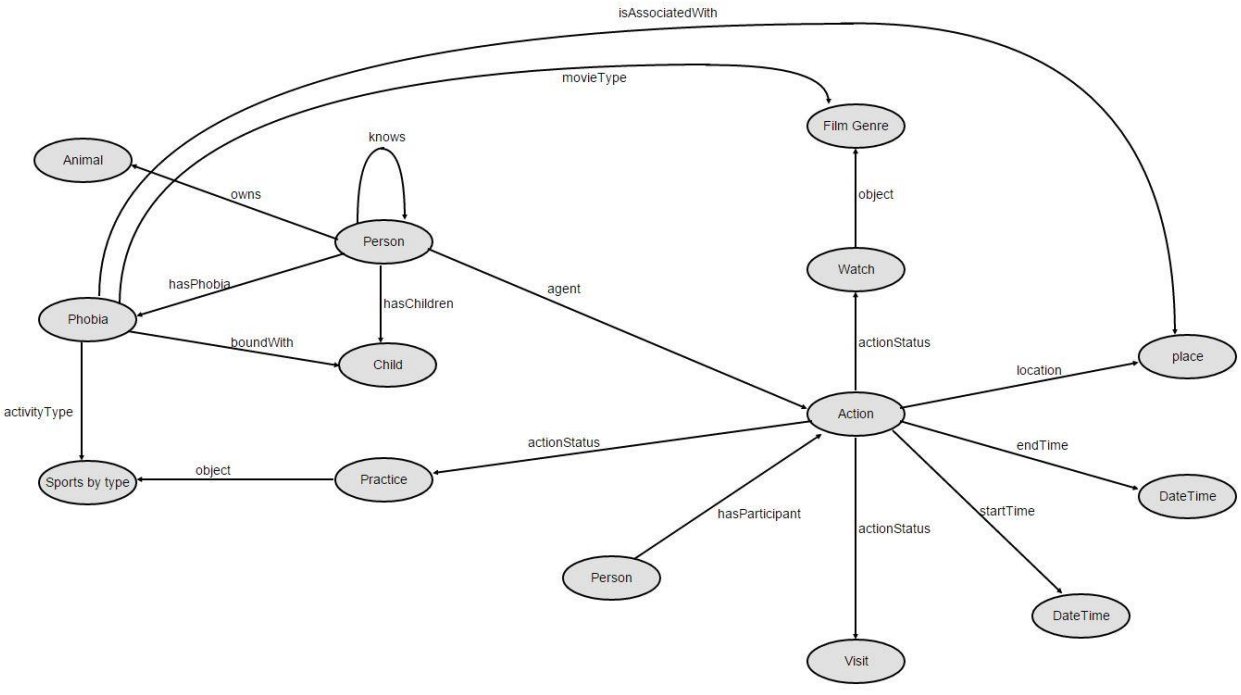
For developing PhoS application, to create the ontology (OWL) for rdf rules between information we used Protégé.

An ontology is a description language of the concepts and relationships that can exist betweenn database entities that may be used. The ontology allows the user to define classes, taxonomy hierarchy (subclass-superclass) and creating instances.

The information from PhoS application is based on a custom ontology that uses Dbpedia, Schema.org, FOAF data to create the rules and the enities.

The PhoS ontology uses Person from FOAF to define the user that at that time need to fulfill data in the database. The foaf:Person defines the ‘knows’ predicate which can be used to make connections between users. Based on a custom predicate, ‘hasPhobia’, the foaf:Person can be associated with a certain Phobia, also a custom entity created with Protégé editor. A person who owns an animal is associated with the entity Animal from Dbpedia, by the predicate “owns” from Schema.org. Another custom predicate is “hasChildren”, added to allow the application to discover the phobias related to children.

The person is defined with the schema.org attribut as being agent of an Action (schema.org entity). At that Action another participant must be a person in ‘foaf:Person knows’ relation with the user. The ‘actionStatus’, defined as an attribut of the Action entity can have the following values: “visit”, “watch”, “practice”. The user can visit a city from Romania, defined as Dbpedia entity. Also, the person can watch a movie with another person, action possible by



indicating the movie type (Dbpedia resource category Film_genre) or he can practice a Sport, categorized and represented by Sports_by_type from Dbpedia resource category.

After the user enters the context, the application makes SPARQL queries to the database to find the phobias that might appear for the user, but also for the user's chosen friend. Those queries search in the action's agents phobias lists, those phobias associated with the action selected by the user.

For example, if an user wants to have a trip ("visit") with another user in "Constanța", and he has "Aquaphobia" and his friend has "Fear of driving", then the application should list both of these phobias, with a few linked suggestions of treatments and tips to avoid them.

Another example, could be shown in a situation in which an user has children, and the selected friend has a fear of children. Then, the application should detect the user's phobias which can be found in a "subject predicate object" triple like: "?phobia phos:boundWith dbo:Child", where "phos" is the prefix for Phos Ontology.

SPARQL interrogation for example 1 (for user's phobias):

```
PREFIX dbo: <http://dbpedia.org/ontology/>

PREFIX db: <http://dbpedia.org/>

SELECT (str(?p) AS ?phobiaName)

WHERE {

<userResource > <http://www.semanticweb.org/tibabianca/ontologies/2016/0/phos#hasPhobia> ?p.

{ ?p  <http://www.semanticweb.org/tibabianca/ontologies/2016/0/phos#isAssociatedWith> dbo:Beach.}

UNION

{ ?p <http://www.semanticweb.org/tibabianca/ontologies/2016/0/phos#activityType> db:Place. }

}
```

Another example of SPARQL interrogation, used to obtain the list of Sports type from Dbpedia source is in the following example:

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

PREFIX dbc: <http://dbpedia.org/resource/Category:>

PREFIX skos: <http://www.w3.org/2004/02/skos/core#>


SELECT DISTINCT ?sportType ?sportName

WHERE {

?sportType skos:broader dbc:Sports_by_type.

?sportType rdfs:label ?sportName.

FILTER (langMatches(lang(?sportName), "en"))

 }
```

Bibliography

- https://en.wikipedia.org/wiki/HTML5
- http://tympanus.net/codrops/2011/11/24/top-10-reasons-to-use-html5-right-now/
- http://searchenterpriselinux.techtarget.com/definition/MySQL
- http://dev.mysql.com/doc/refman/5.7/en/what-is-mysql.html
- https://jena.apache.org/
- http://www.disgenet.org/web/DisGeNET/menu/home
- http://www.wordstream.com/blog/ws/2014/03/20/schema-seo
- http://common-phobias.com/
- http://wiki.dbpedia.org/
- https://schema.org/
- http://xmlns.com/foaf/spec/#
- http://wallpaper.pickywallpapers.com/1280x800/screaming.jpg