rouT (Smart Tourist)

Technical Report

**Abstract.** This application is a solution for every traveler or this application can be used by a citizen who don’t know very much the other place of his town or a family who want to go shopping in a new town or some students who try to find some fun in other unknown places. Also this application is a demonstration of using new technologies in a best way to guarantee information for any type of person that is using this application. rouT is giving a good infrastructure to be used in any kind of condition and you can get the information that you want in a short time.

1. Introduction

Smart Tourist is an innovative application that will help a traveler to know more information about an unfamiliar place. This application would have a great impact also to citizen of city for knowing the touristic place of their town, shopping place and also financial help or hospital and pharmacy.

This application is an easy solution for any traveler to be used in an efficient and quickly way to find any kind of information about the new place where is located. Also the application would have a friendly interface that can be used easily and the application will provide the right information in a quickly way. rouT application would have a car interface to be easily used when you drive to find quickly charger places, gas stations or shopping centers.

On the other hand, this application have a implementation structure that is easy to reused the code and to add more module, is using robust architecture that will be a good way to reused the code all to replace a module with other.

Smart Tourist (rouT) is using new technologies in an infrastructure of semantic web

1. Technologies used

### **RDF**

The Resource Description Framework (RDF) is a family of [World Wide Web Consortium](https://en.wikipedia.org/wiki/World_Wide_Web_Consortium) (W3C) [specifications](https://en.wikipedia.org/wiki/Specification) originally designed as a [metadata](https://en.wikipedia.org/wiki/Metadata) [data model](https://en.wikipedia.org/wiki/Data_model). It has come to be used as a general method for conceptual description or modeling of information that is implemented in [web resources](https://en.wikipedia.org/wiki/Web_resource), using a variety of syntax notations and data serialization formats. It is also used in knowledge management applications.

RDF was adopted as a W3C recommendation in 1999. The RDF 1.0 specification was published in 2004, the RDF 1.1 specification in 2014.

### **AllegroGraph**

AllegroGraph is a persistent Resource Description Framework (RDF) graph database. AllegroGraph uses diskbased storage, enabling it to scale to billions of triples while maintaining superior performance. *AllegroGraph* supports SPARQL, RDFS++, and Prolog reasoning from Java applications. Allegrograph provides a Representational State Transfer (REST) protocol architecture Figure 2 Allegrograph is used for Geotemporal reasoning and social networking analysis.

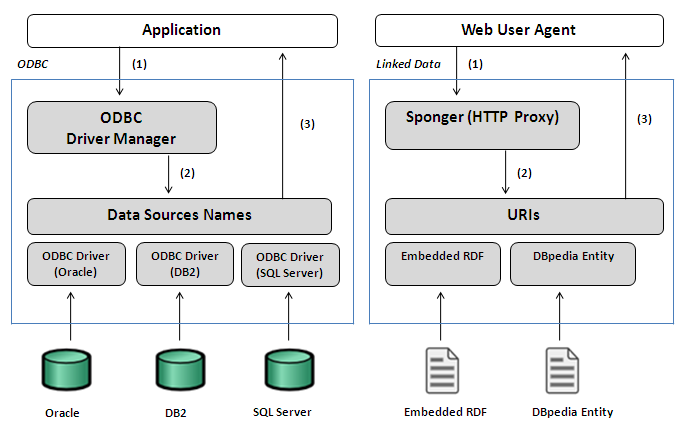
AllegroGraph comes in multiple flavors and works with multiple programming languages and environments and as an advantage for Smart Tourist (rouT) application which uses a server-side application built in NodeJs, the AllegroGraph RDF Server can also be scripted using the JavaScript Language. Moreover, it is possible to interact with AllegroGrap completely using a RESTful HTTP protocol (using GET, PUT, POST, etc…) to add and delete triples, to query for individual triples and to do SPARQL and selects using the Sesame 2.0 HTTP-interface with some extensions.

#### **Virtuoso**

Virtuoso is a modern enterprise grade solution for data access, integration, and relational database management (SQL Tables and/or RDF based Property/Predicate Graphs).

The unique hybrid server architecture of Virtuoso enables it to offer traditionally distinct server functionality within a single product offering that covers the following areas:

* Relational Tables Data Management [(Columnar or Column-Store SQL RDBMS)](http://virtuoso.openlinksw.com/rdbms-engine)
* Relational Property Graphs Data Management [(SPARQL RDF based Quad Store)](http://virtuoso.openlinksw.com/dataspace/doc/dav/wiki/Main/VOSRDF)
* Content Management [(HTML, TEXT, TURTLE, RDF/XML, JSON, JSON-LD, XML)](http://virtuoso.openlinksw.com/dataspace/doc/dav/wiki/Main/VOSDAV)
* Web and other Document File Services [(Web Document or File Server)](http://virtuoso.openlinksw.com/dataspace/doc/dav/wiki/Main/VOSDAV)
* Five-Star Linked Open Data Deployment [(RDF-based Linked Data Server)](http://virtuoso.openlinksw.com/linked-data)
* Web Application Server [(SOAP or RESTful interaction modes)](http://virtuoso.openlinksw.com/web-application-server).



### **Sesame**

Sesame has two main communication interfaces: the Sail API and the Repository API. The Storage And Inference Layer (Sail) API is a low level system API (or: SPI) for RDF stores and inferencers. It's purpose is to abstract from the storage details, allowing various types of storage and inference to be used.

The Repository API is a higer level API and is meant to be the main API that people can program against. It offers various methods for uploading data files, querying, and extracting and manipulating data. It comes in two flavours: local and remote. The local Repository API can be used for "local" repositories, as in "running in the same Java Virtual Machine". The remote Repository API can be used in client-server settings, where the application communicates with a Sesame server. The two flavours share a common interface so that applications can be developed to work with both local and remote repositories transparently.

Part of the Sail API is the Sail Query Model, which can be used to build object models representing queries. A query engine like the SeRQL engine takes care of parsing query strings and building such query object model from them. These query object models can then be passed to Sail objects, which take care of their evaluation.

Rio stands for "RDF I/O" and consists of a set of parsers and writers for various RDF file formats. The parsers can be used to translate RDF files to lists of statements, and the writers for the reverse operation. Rio is used by the Repository API for data import and export, but can also be used directly by applications.

### **AngularJS ( Client Level )**

### AngularJS (commonly referred to as "Angular" or "Angular.js") is a complete JavaScript-based [open-source](https://en.wikipedia.org/wiki/Open-source_software) front-end [web application framework](https://en.wikipedia.org/wiki/Web_application_framework) mainly maintained by [Google](https://en.wikipedia.org/wiki/Google) and by a community of individuals and corporations to address many of the challenges encountered in developing [single-page applications](https://en.wikipedia.org/wiki/Single-page_application). The JavaScript components complement [Apache Cordova](https://en.wikipedia.org/wiki/Apache_Cordova), the framework used for developing cross-platform mobile apps. It aims to simplify both the development and the [testing](https://en.wikipedia.org/wiki/Software_testing) of such applications by providing a framework for client-side [model–view–controller](https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller) (MVC) and [model–view–viewmodel](https://en.wikipedia.org/wiki/Model_View_ViewModel) (MVVM) architectures, along with components commonly used in [rich Internet applications](https://en.wikipedia.org/wiki/Rich_Internet_application).

The AngularJS framework works by first reading the [HTML](https://en.wikipedia.org/wiki/HTML) page, which has embedded into it additional custom [tag attributes](https://en.wikipedia.org/wiki/HTML_attribute). Angular interprets those attributes as [directives](https://en.wikipedia.org/wiki/Directive_(programming)) to bind input or output parts of the page to a model that is represented by standard [JavaScript](https://en.wikipedia.org/wiki/JavaScript) [variables](https://en.wikipedia.org/wiki/Variable_(computer_science)). The values of those JavaScript variables can be manually set within the code, or retrieved from static or dynamic [JSON](https://en.wikipedia.org/wiki/JSON) resources.

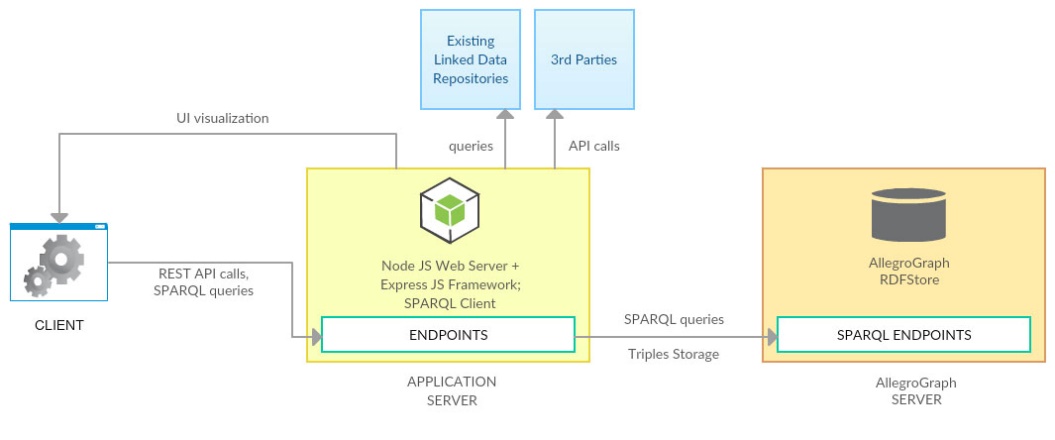
1. Application architecture

The application consists of three main modules:

* Client
* Application Server which hosts an Node JS Web Server together with Express JS Framework and a SPARQL Client
* AllegroGraph Server which hosts and AllegroGraph RDF Store along with SPARQL Endpoints

In the first phase, the client is using REST API calls and SPARQL Queries to interact with the Application Server. Here, the server is using different queries to obtain data from existing linked data repositories, and API calls to 3rd Parties, like Google Places to obtain information about points of interest (reviews, indications) or Google Directions to define the route of the trip.

After this step, the Application Server is querying the database by sending the SPARQL Queries to the AllegroGraph Server. Here, the data is stored as different triples (subject – predicate – object) to define the relationships between the entities involved.



1. Server-side

There will be involved two servers, an application server and a database server. The application server will contain Node JS Web Server along with Express JS Framework and SPARQL Client. Here, there will be used different queries to the existing linked data repositories to get as much possible information for the client, describing points of interest (hospitals, places to eat or sleep, airports, etc). To get more information, there will be used different API calls to 3rd parties like Google Places and Google Directions.

The other server will store the AllegroGraph database. The data will be find as triples (subject – predicate – object), for example Traveller | is a | person, and User 1 | is a | traveler. This data will be retrieved using SPARQL queries, like:

PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?username ?password

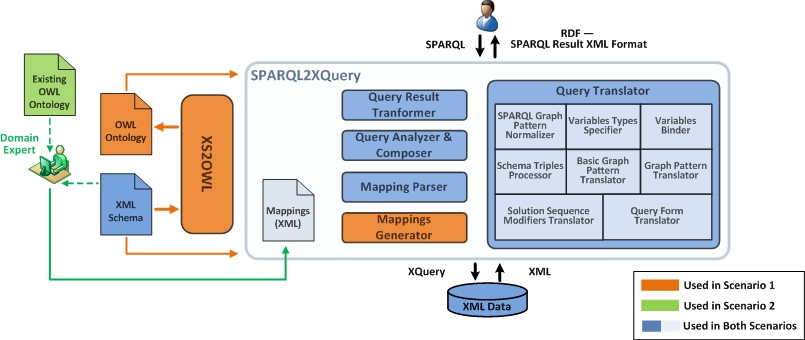
WHERE {

?person a foaf:Traveller.

?person foaf:user ?username.

?person foaf:pass ?password.

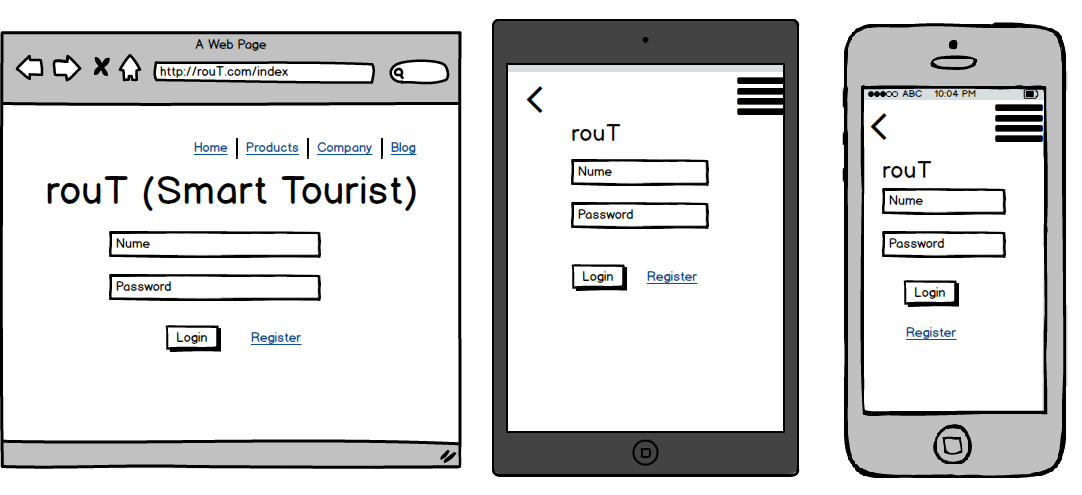
}



1. Client-side

On client side, would be a friendly interface that would help the traveler user to understand all the information that this application offers to the user traveler. The user interface have a intuitive structure that would be easy for traveler to find the right information in a quickly way. The interface is responsive that will be a great way to access this application of any kind.

Here, the client sends REST API calls to the application server, along with SPARQL queries, which will be sent later to the database server.



1. Features

The bonus features that this application will have possible to add geospatial semantic information to the World Wide Web. This features would be a strong efficient and smart way to make a great planning of a travel in any place of Earth, with this application you will have a good support to orient yourself in any unfamiliar place you get.

1. Bibliography

<https://virtuoso.openlinksw.com/>

<https://angular.io/docs/js/latest/>

<https://nodejs.org/api/>

http://www.geonames.org/ontology/documentation.html