A Trivia like Mobile Game with Autonomous Content That Uses Wikipedia Based Ontologies

Bogdan IANCU
The Bucharest University of Economic Studies
bogdan.iancu@ie.ase.ro

The mobile devices are everywhere and the gaming industry has an important market share on this field. This paper aims to present a new approach of game which generates autonomous its content, without human intervention, by using existing facts from the semantic version of Wikipedia, called DBpedia. At the beginning a short introduction of the studied field and of the used terms and technologies is presented. There are also remembered similar approaches and different struggles from this domain. In the middle part the architecture and the logic of the system are shown. The paper ends with some conclusions and future plans.
Keywords: *Semantic Web, Ontologies, Open Linked Data, Mobile Games*

1 Introduction

The mobile industry is literally huge. In the US, 55% of the adult population owns a smart phone and it is predicted that by the end of 2016 we will have 1.4 smart devices per person on the planet. Even if the market is so big, its granularity is still high from an OS point of view. The leader is the Android OS, closely followed by the iOS. Another players are the Blackberry OS, Windows OS and the last but not to ignore, the recent launched Firefox OS.

A mobile device owner uses in average 26.8 apps per month and 57% of these people use their apps daily. The most used apps are from the "Search, Portal & Social" category (with 10 hours and 56 minutes spent per month per user), closely followed by the "Entertainment" apps which are used 10 hours and 34 minutes monthly per user [12].

Unfortunately the most apps from the Entertainment category, which the games are part of, involve a whole team of developers for the engine of the game at the beginning and then, for developing the content. With investments that sometimes overtake the budgets of many movies, the games consume more and more money.

But what if we can use an automatic way of providing the content? Robots that write press articles already exist [10], so why not developing a game that uses a semantic web approach in providing its content in general, or to be more specific, general knowledge questions in particular.

The path for semantic answering questions in a gaming environment was paved since February 2011 when the Watson Question Answering system built by the IBM Research team challenged two human champions in the American TV quiz show Jeopardy! and bested them. However, the questions being played for the quiz had been created by human authors. The information used to solve these questions came from the Linked Open Data (LOD) and analysis of large amount of documents like newspaper articles. Even though Watson has won the quiz show, the knowledge that can be drawn automatically from the LOD cloud is far from being perfect [8].

But this thing doesn't mean it can't be used for entertainment purposes like question source for a trivia like game. In fact its correctness can be adjusted by players. If more players report a question as being incorrect then the system can consider that most probably it is and ignore it from future shows. Maybe it is a better approach to have an almost infinite number of possible question with an accepted error ratio, than to have a limited number of 100% correct questions that will repeat themselves.

DOI: 10.12948/issn14531305/19.1.2015.02

2 Technologies Used

As we saw in the previous section, right now there isn't a unique approach in developing a native mobile application. Each platform and operating system has its own development environment and its own programming language for writing apps.

But, even so, there is an alternative in the name of hybrid apps. They are usually responsive web applications that run in a native web browser control. There are multiple platforms for developing this kind of apps, but the most important is PhoneGap.

PhoneGap is a platform based on HTML and JavaScript which can be used for developing hybrid mobile applications (Figure 1). Basically a PhoneGap application implements platform specific webView like controls which run JavaScript code in order to access specific elements of the mobile device like network connectivity, contacts, file system, multimedia or camera. The graphic interface is defined exclusively by using HTML5 and CSS3 and, in order to display it on a mobile device, the browser features are used. This approach unfortunately makes the PhoneGap apps slower than similar native apps, but the advantages of writing code just once and knowing just one programming language make it a reliable solution [2].

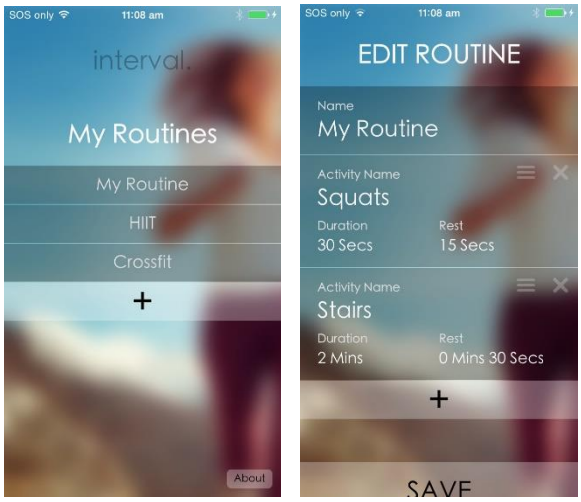

Fig. 1. Example of a PhoneGap application design

Node.js, called sometimes simply – Node, is a server-side JavaScript environment that is based on Google's runtime implementation — the aptly named "V8" engine. V8 and Node are mostly implemented in C and C++, focusing on performance and low memory consumption. But, whereas V8 supports mainly JavaScript in the browser (most notably, Google Chrome), Node aims to support long-running server processes. Unlike in most other modern environments, a Node process doesn't rely on multithreading to support concurrent execution of business logic; it's based on an asynchronous I/O event model (Figure 2). Think of the Node server process as a single-threaded daemon that embeds the JavaScript engine to support customization. This is different from most event systems for other programming languages, which come in the form of libraries: Node supports the event model at the language level. JavaScript is an excellent fit for this approach because it supports event callbacks. For example, when a browser completely loads a document, a user clicks a button, or an Ajax request is fulfilled, an event triggers a callback. JavaScript's functional nature makes it extremely easy to create anonymous function objects that you can register as event handlers [9].

DOI: 10.12948/issn14531305/19.1.2015.02

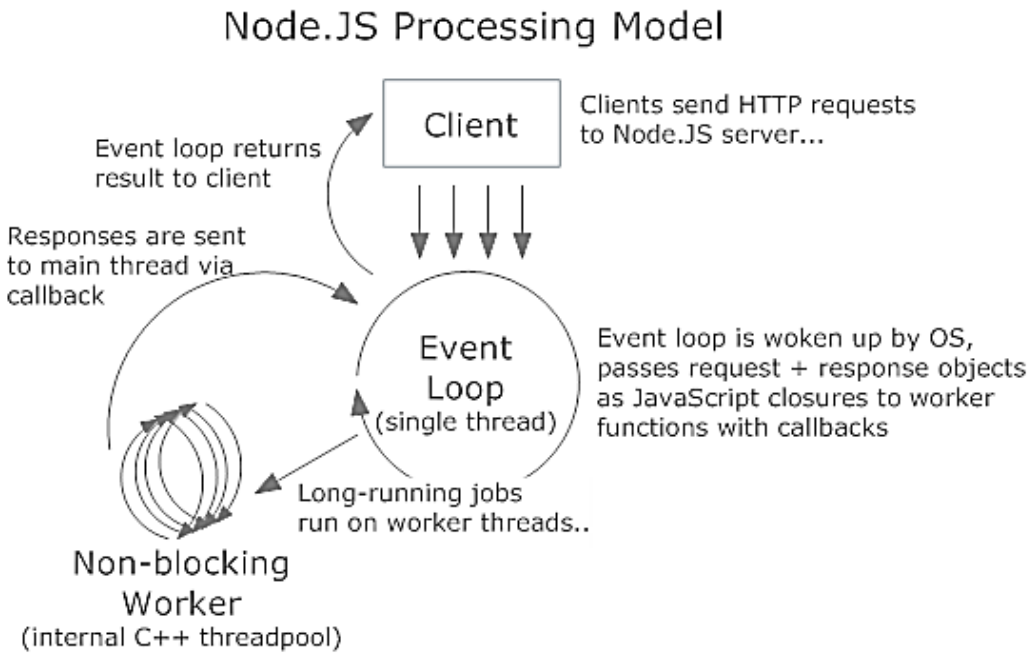

Fig. 2. Node.js processing model

Express is a web application framework for Node.js. It is minimal, flexible and great for building web APIs. Express extends Node and has a robust set of features for web and mobile applications. It can be installed via npm package manager. For details about Express.js and npm please see [14] and [15].

Angular.js is a client-side JavaScript framework for adding interactivity to HTML pages. It is maintained by Google and a community of individual developers and corporations to address many of the challenges encountered in developing single-page applications. Its goal is to simplify both development and testing of such applications by providing a framework for client-side model–view–whatever (MVW) architecture, along with components commonly used in rich internet applications.

The library works by first reading the HTML page, which has embedded into it additional custom tag attributes. Those attributes are interpreted as directives telling Angular to bind input or output parts of the page to a model that is represented by standard JavaScript variables. The values of those JavaScript variables can be manually set within the code, or retrieved from static or dynamic JSON resources. For more information about Angular please consult [16].

MongoDB is an open-source document database written in C++ and the leading NoSQL database. NoSQL refers to non-relational databases promising to handle large volumes of structured and unstructured data. Unlike relational databases the NoSQL databases do not require schemas to be defined beforehand. These fits perfectly fine with the agile technologies as each time a new feature is developed, the schema of the database is needed to be changed. The data-structure of non-relational database is not fixed. This allows the insertion of data without a predefined schema. This ultimately proves helpful in making significant changes in applications in real-time, with no interruption in service, leading to faster development, reliable code integration and lessens the amount of time by database administrators. NoSQL also referred as “Not only SQL” to emphasize that these also support SQL like queries. Its architecture is based on a set of collections that hold each a set of documents. A document further is a set of key-value pairs. These documents have a dynamic schema which means that documents in the same collection do not need to have the same set of fields or structure.

DOI: 10.12948/issn14531305/19.1.2015.02

The key feature for which MongoDB is used is its flexibility which is understood as that the data is stored in JSON documents [3]. JSON will be in fact the keyword in the presented paperwork.

An analogy between the leading SQL database (MySQL) and MongoDB from a query point of view is presented in Figure 3. For more details about MongoDB please visit [17].

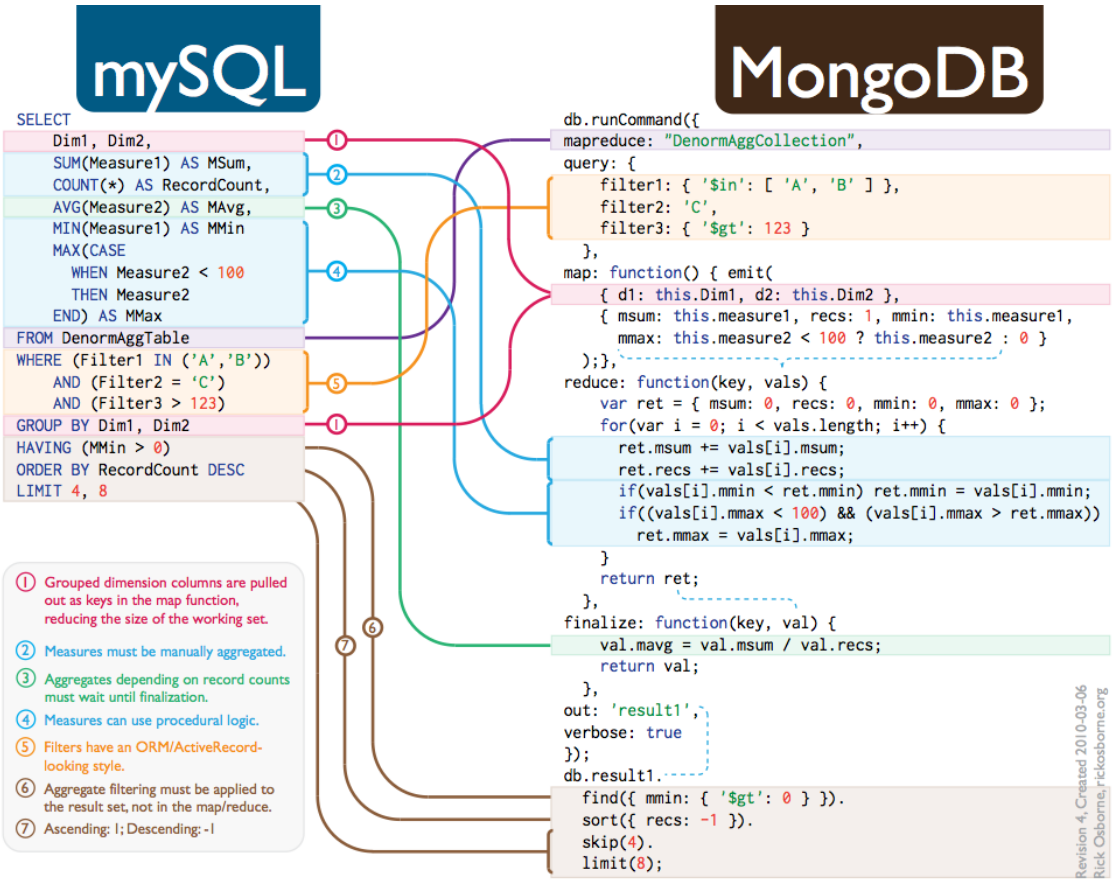

Fig. 3. Running a query in a SQL environment (MySQL) vs running the same query in a NoSQL environment (MongoDB)

The last but not the least technology used is Linked Open Data. It is also the most important part of the system, the feature that makes the designed architecture to work autonomous.

The Linked Data is the process of applying the general architecture of the World Wide Web to the task of sharing structured data on a global scale [1]. It uses the architecture of the classic document Web with Uniform Resource Identifiers (URIs) as globally unique identification mechanism, the Hypertext Transfer Protocol (HTTP) as universal access mechanism and different XML or JSON derived formats for presenting or formatting the data, like RDF, OWL or JSON-LD [11]. When the data that is presented is publicly available we call it Linked Open Data (LOD).

There is a graphical representation of how URIs in the LOD link to each other provided by [4], named the LOD cloud diagram (Figure 4). As we can see there, the nucleus of LOD on this moment is DBpedia, the data source chosen for this system.

RDF (Resource Description Framework) is a standard model for data interchange on the Web. RDF has features that facilitate data merging even if the underlying schemas differ, and it specifically supports the evolution of schemas over time without requiring all the data consumers to be changed. RDF extends the linking structure of the Web to use URIs to name the

DOI: 10.12948/issn14531305/19.1.2015.02

relationship between things as well as the two ends of the link (this is usually referred to as a subject–predicate–object "triple") [13]. Using this simple model, it allows structured and semi-structured data to be mixed, exposed, and shared across different applications.

SPARQL for RDF [18] is a query language that can be used to retrieve information across diverse data sources, whether the data is stored natively as RDF or viewed as RDF via middleware. SPARQL contains capabilities for querying required and optional graph patterns along with their conjunctions and disjunctions. SPARQL also supports extensible value testing and constraining queries by source RDF graph. The results of SPARQL queries can be results sets or RDF graphs.

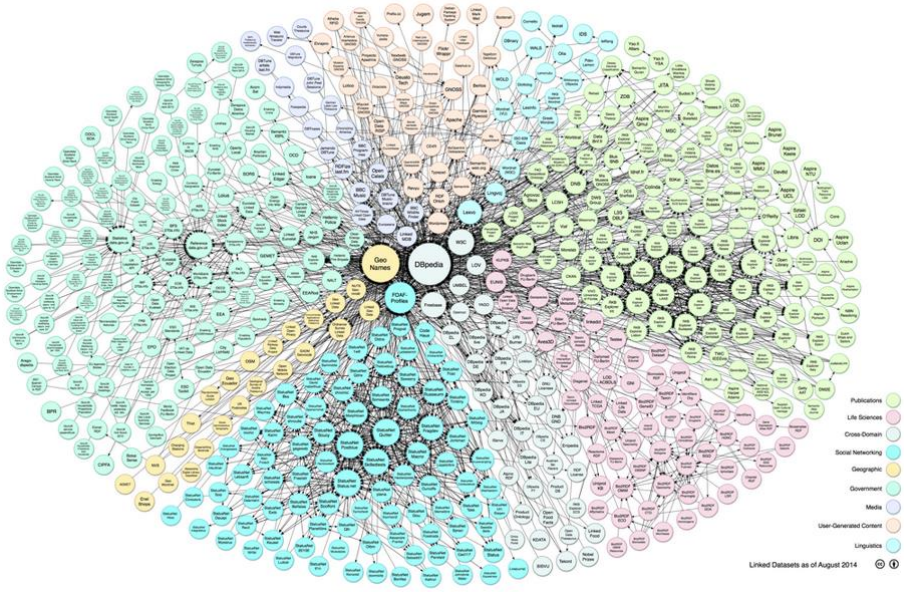

Fig. 4. LOD cloud diagram

DBpedia is the semantic version of Wikipedia, the largest encyclopedia ever, which was constructed by a very large open community. It uses the information from Wikipedia's info boxes and makes it available for machines in ontological formats like RDF and OWL. It also makes the structured data queryable against a SPARQL endpoint.

DBpedia contains all the RDF specific elements like RDF Subject, RDF Property and RDF Object, elements that we need for the proposed system.

3 The Trivia Like Mobile Game

The designed game aims to involve smart device users from around the world in testing their general knowledge in 1 vs 1 battles in order to gain points and respect of their friends. The application uses gamification techniques and social features to increase the users' commitment.

As [7] show the social network users are concerned about what their friends do and care about what their friends think about them. Thus the application will recommend the players to login with their Facebook account so they can be able to compete with their social network friends in the pursuit for points (Figure 5 left side).

The system provides the users with simple questions that begin with one of the four "W"s ("What is…?", "Who is…?", "When was…?" and "Where is…?") continued by a single term and 4 possible answers (each one composed from a word or more that represent together a single entity). Based on their answer speed and the accuracy of their choices a score is computed. The main screen shows the picture of the player (if available),

DOI: 10.12948/issn14531305/19.1.2015.02

his overall score and his abilities together with the win/loss ratio (Figure 5 right side).
The game itself consists in a 1 minute battle between two random selected players for answering as many question as they can. The players are selected by the server based on their overall points so the match can be balanced. The winner will be the player who answers the most question in the given time (Figure 6 left side). If both players report the same question as being incorrect or strange, then the app ignores it when counting the score. If the user logins with a facebook account then he has access to a scoreboard that contains all his friends who are playing the game together with their scores (Figure 6 right side).

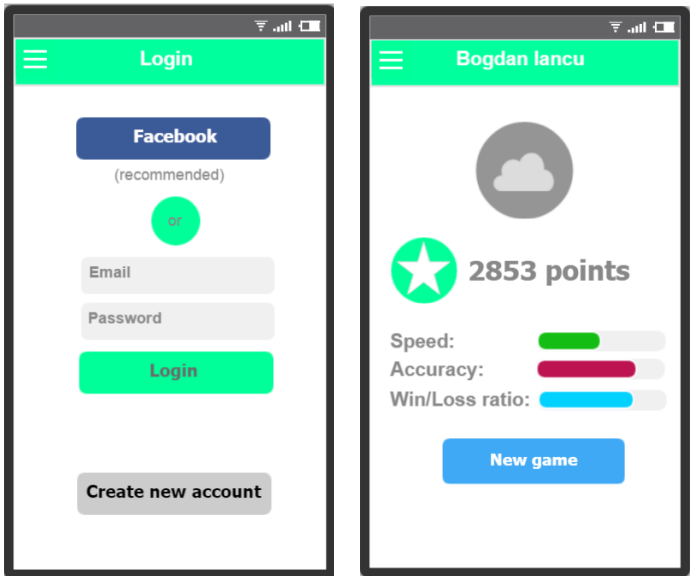

Fig. 5. The user interface: the login screen (left side) and the main screen (right side)

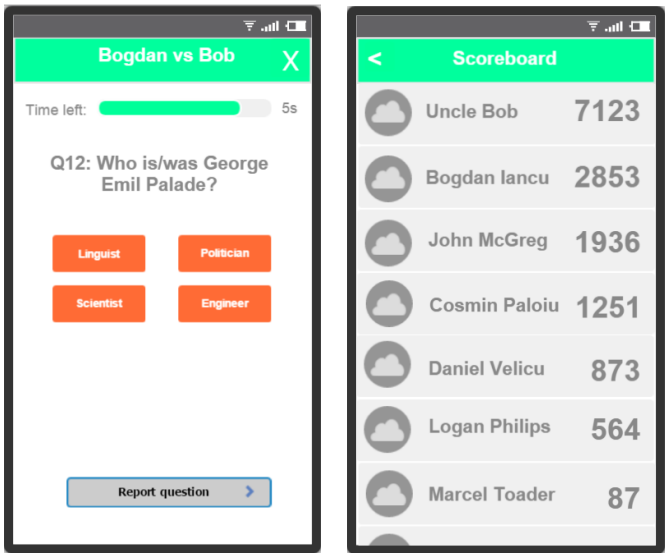

Fig. 6. The user interface: the game view (left side) and the scoreboard (right side)

4 The Architecture of the System

The server, written in Express.js uses a RESTful approach. It accept HTTP GET calls for services like: `loginUser`, `getProfile`, `findSimilarPlayer`, `getPackOfQuestions`, `getScoreBoard` and other similar functions. HTTP POST methods are used for creating a new account (or when the user logins with facebook for the first time), for reporting a question as incorrect and for saving the game result (together with the statistics).

DOI: 10.12948/issn14531305/19.1.2015.02

If almost all the services presented above are self-explanatory, the most interesting ones are `findSimilarPlayer` and `getPackOfQuestions`. For finding a similar available player *y* for the one that requests the challenge – *x*, the following formula is applied: $y = min(|P_x - P_y|)$, where P_x is the score of the player *x*.

Other formulas used on the server are:

$Sx = \frac{\Sigma\left(\frac{qx}{nx}\right)}{\max(\Sigma\frac{qi}{ni})}$, where *S* is the player's speed, *q* is the number of questions asked in a game by that player and *n* the total numbered of questions asked in that game or $max(q_1, q_2)$ and $Ax = \frac{\Sigma rx}{\Sigma qx}$, where A_x is the player's accuracy and *r* is the number of correct answers.

Regarding the other service, the one that gives a pack of questions, things are a little bit more complicated. First of all, why a pack a question and not a single one? Because the game is time based and no human player could answer more than 100 questions in 1 minute. Thus for optimization purposes the server gives to the client application a set of 100 questions not a single one. This thing ensures that no service calls are made during the game so it can be as smooth as possible.

One might ask now, how the questions are automatically and autonomous computed? The server calls the DBpedia SPARQL endpoint, running queries in search for RDF object, RDF specific properties and RDF subjects. If the properties are aprioristically known (they are given by the four types of questions), the subject is selected randomly together with the correspondent object(s). In case of more objects, the answer will be the one that is more precise (lower in class hierarchy) and in case of more possible items in this situation, it will be selected randomly. After that a new call is being made that request 3 random object from the same class as the correct object, that don't share the same property with the object.

Let's take the question "Who is/was George Emil Palade?" as an example. So the server selected randomly that the next question will be of type "Who is…?". This means that the queried property will be `rdf:type` and the object will be of type `dbpedia-owl:Person`. Now the first call is made to the SPARQL endpoint with the following query (where the offset number is generated randomly by the server):

```
SELECT ?subject WHERE {
?subject rdf:type dbpedia-owl:Person
} OFFSET 43509 LIMIT 1
```

The response will have the subject `http://dbpedia.org/resource/George_Emil_Palade`. Now a new query is made that ask for an object correspondent to the selected subject that is also a subclass of `dbpedia-owl:Person`:

```
SELECT ?object WHERE {
<http://dbpedia.org/resource/George_Emil
_Palade> rdf:type ?object .
?object rdfs:subClassOf dbpedia-
owl:Person
}
```

The result will be `dbpedia-owl:Scientist`. After that a third call will be made with the next query to ask for another 3 objects that have the same superclass as `dbpedia-owl:Scientist`. Those will be `dbpedia-owl:Linguist`, `dbpedia-owl:Politician` and `dbpedia-owl:Engineer` let's say, all of them subclasses of `dbpedia-owl:Person` and none of them applicable to George Emil Palade:

```
SELECT ?object WHERE {
?object rdfs:subClassOf dbpedia-
owl:Person .
FILTER ( !EXISTS{
<http://dbpedia.org/resource/George_Emil
_Palade>  rdf:type  ?object })
} ORDER BY RAND() LIMIT 3
```

A similar approach is used for the 3 other types of questions. In order to keep up with the players, the server will continuously generate new questions and save them in a question cache in MongoDB.

When a new game is started the server will return a block of 100 questions to the client and then erase them from the NoSQL database. The server is optimized to have all the time a sufficient number of questions and to stop generating new ones when the

DOI: 10.12948/issn14531305/19.1.2015.02

maximum storage limit for the questions' file is reached.
The MongoDB contains also the Model and the Business Logic of the application: users, game, statistics, etc.
The mobile app uses PhoneGap and Angular JS in order to bind the views' properties and commands to server's API. It consists in HTML5 pages that make asynchronous calls to the Express.js server when a specific event happens.
The overall picture of the designed system can be deducted from Figure 7.

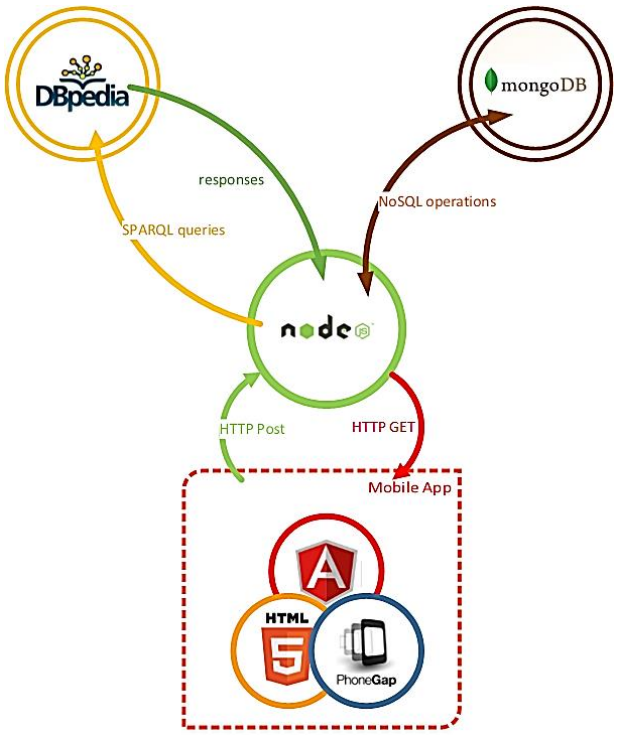

Fig. 7. The designed system's architecture

4 Conclusions

In this paper was presented a simple trivia like mobile game system that uses semantic web technologies. The knowledge base used is DBpedia because it is the largest open data resource and the center of the LOD cloud. I chose a loosely tied architecture based on RESTful services, Node.js, MongoDB and HTML5. Even if there are other similar works like [5] or [6], they provide just simple ideas to be implemented. The proposed system is a fully-described one, ready to be launched on the mobile devices marketplaces. Future plans include testing the system on a group of students and the support of more social networks.
Note: Parts of this work were presented in "Wikipedia based short answer quiz system using semantic web technologies", in the Proceedings of the 4th Multidisciplinary Academic Conference in Prague 2015, Czech Republic, February 2015, ISBN 978-80-905442-9-1.

Acknowledgments

This paper was co-financed from the European Social Fund, through the Sectorial Operational Programme Human Resources Development 2007-2013, project number POSDRU/159/1.5/S/138907 "Excellence in scientific interdisciplinary research, doctoral and postdoctoral, in the economic, social and medical fields -EXCELIS", coordinator The Bucharest University of Economic Studies.

References

[1] T. Heath, C. Bizer, "Linked Data: Evolving the Web into a Global Data Space (1st edition)," *Synthesis Lectures on the Semantic Web: Theory and*

Technology, Vol. 1, No. 1, pp. 1-136, Morgan & Claypool, 2011.

[2] Pocatilu P., Ivan I., Vișoiu A., Alecu F., Zamfiroiu A., Iancu B. (2014), Programarea aplicațiilor Android, Editura ASE, pp 62

[3] Punia, Y., & Aggarwal, R. (2014) Implementing Information System Using MongoDB and Redis, International Journal of Advanced Trends in Computer Science and Engineering, 3(2), pp 16 - 20

[4] Linking Open Data cloud diagram 2014, by Max Schmachtenberg, Christian Bizer, Anja Jentzsch and Richard Cyganiak. http://lod-cloud.net/

[5] Waitelonis, J., Ludwig, N., Knuth, M., & Sack, H. (2011). Whoknows? Evaluating Linked Data Heuristics with a Quiz that Cleans up DBpedia. Interactive Technology and Smart Education, 8(4), 236-248.

[6] Balog-Crisan, R., Roxin, I., & Szilagyi, I. (2009, July). Ontologies for a semantic quiz architecture. In Advanced Learning Technologies, 2009. ICALT 2009. Ninth IEEE International Conference on (pp. 492-494). IEEE.

[7] C. Delcea, L.-A. Cotfas, & R. Paun (2014). Grey Social Networks A Facebook Case Study. In Computational Collective Intelligence. Technologies and Applications, vol. 8733, D. Hwang, J. J. Jung, and N.-T. Nguyen, Eds. Springer International Publishing (pp. 125–134).

[8] L. Wolf, M. Knuth, J. Osterhoff, & H. Sack, (2011, September). RISQ! Renowned Individuals Semantic Quiz: a Jeopardy like quiz game for ranking facts. In Proceedings of the 7th International Conference on Semantic Systems (pp. 71-78). ACM.

[9] Tilkov, S., & Vinoski, S. (2010). Node. js: Using JavaScript to build high-performance network programs. IEEE Internet Computing, 14(6), 0080-83.

[10] Y. Eudes (2014, September 12). The journalists who never sleep [Online]. Available:http://www.theguardian.com/technology/2014/sep/12/artificial-intelligence-data-journalism-media

[11] W3C Wiki (2014 January 28). Semantic Web Standards [Online]. Available: http://www.w3.org/2001/sw/wiki/Main_Page

[12] The University of Alabama at Birmingham's Collat School of Business infographic, available online at http://businessdegrees.uab.edu/resources/infographic/the-future-of-mobile-application/

[13] W3C Wiki (2014 March 15). Resource Description Framework (RDF) [Online]. Available: http://www.w3.org/RDF/

[14] Express - Node.js web application framework Official Site [Online]. Available: http://expressjs.com/

[15] npm Official Site [Online]. Available: https://www.npmjs.com/

[16] AngularJS Official Site [Online]. Available: https://angularjs.org/

[17] MongoDB Official Site [Online]. Available: http://www.mongodb.org/

[18] S. Harris, A. Seaborne. (2013, March 21). SPARQL 1.1 Query Language. W3C Recommendation [Online]. Available: http://www.w3.org/TR/2013/REC-sparql11-query-20130321

Bogdan IANCU has graduated The Faculty of Cybernetics, Statistics and Economic Informatics from The Bucharest University of Economic Studies in 2010. He has a master in Economic Informatics (2012) and he is a PhD Candidate in Economic Informatics starting from 2012 in the field of Ontologies and eLearning. He is a teaching assistant in The Department of Economic Informatics of The Bucharest University of Economic Studies. His current work focuses on the analysis of semantic web and ontologies innovations. Other fields of interest include data mining, multimedia, mobile devices programming and Big Data.

DOI: 10.12948/issn14531305/19.1.2015.02