Distributed Architecture of Mobile GIS Application Using NoSQL Database

Jonathan Rodriguez, Anthony Malgapo, Jacob Quick, and Chung-yu Huang

***Abstract*—The primary focus of our project was to investigate and build the distributed architecture of a mobile GIS (geographic information system) application. The purpose of this application would be for displaying useful information on a college campus based on the specific geographic location of our user. Information could include things such as location based events, maps, or other pertinent information that our user may find helpful or informative. A database design that offered acceptable levels of flexibility, ease of use, and quick deployment was explored and ultimately NoSQL MongoDB was chosen. We aimed to implement a database schema specifically geared towards storage of geographic data and MongoDB's flexible schema design fulfilled that requirement. Through the use of MongoDB we also aimed to investigate the tradeoffs of using NoSQL instead of SQL in regards to querying performance and ease of design/development. We explored modern technologies to implement geographical objects (polygons, points, polylines) that were to contain real-time information about geographical objects around our user (buildings, floors, classrooms). And we implemented a Node.js server to retrieve data from our MongoDB according to our user's current location and then handle that GIS information using the Google Maps API. Our project brought us to the conclusion that for the purposes of easy scalability, quick startup development, and ease of creating flexible data models NoSQL databases like MongoDB were preferable for our project over more conventional SQL databases [1]. In addition, we proved out the usefulness of a GIS application geared towards college students to provide useful campus information based on the student's location.**

***Index Terms*—GIS, MongoDB, node.js, NoSQL.**

I. Introduction

This project was undertaken to explore the possible benefits of a GIS application geared towards college campuses where the application's aim would be to provide easy and accessible location based information to students on the campus such as events, maps, or other pertinent information. We also wished to explore NoSQL databases and their use with GIS information as well as their performance compared to more traditional SQL databases we were familiar with. What resulted from this project was a practical GIS based application for our campus. Our application tracks our user's current location, and when our user enters a predefined polygon structure, such as a building stored in our database, our application will then display information relevant to that building to our user. This information could be customized to be something as broad as a building wide event or as specific as a workshop being hosted on a particular floor in a particular room.

We elected to use NoSQL, specifically MongoDB, over a more common SQL database for our backend due to the scalability and performance advantages inherent to NoSQL databases [2]. We also wished to explore NoSQL databases in order to gain a better understanding of their benefits and tradeoffs when used in development versus the more traditional SQL database that we were already familiar with. Due to the ever-rising popularity of NoSQL databases [3] we felt it was important to leverage the advantages of this relatively new technology for the benefit of our application. Due to our application's use case being for a college campus and the ever-changing layout of buildings on college campuses we wanted the flexibility of a NoSQL database to store our building location information. We felt that NoSQL would offer the greatest ease of future revisions to our data models as the campus changed over time. To further ease of use we also constructed a web-based administrative interface for adding information to our MongoDB. Our final application flow resulted in being a mobile application for displaying geo-based information to our users, a web-based administrative interface for adding information to our MongoDB for display to users, and a MongoDB sitting between them to handle the data.

II. Design and Architecture

The two main interfaces of our application are an administrative web application for adding geo-based events to our database and a mobile user application for displaying those events to our users. These two interfaces interact with each other via MongoDB, which handles the data between them. In fig. 1 a diagram of the server architecture is displayed. Both our mobile and web applications were written in AngularJS and Node.js. In order to deploy our web application for development purposes we elected to

Manuscript received August 7, 2017; revised November 14, 2017.

The authors are with School of Computer Science, Kean University, Union, NJ 07083, USA (email: rodrjon1@kean.edu, malgapoa@kean.edu, quickja@kean.edu, chuang@kean.edu).

doi: 10.18178/ijiee.2022.12.4.1

run it using gulp, an automation toolkit [4]. Should the application ever move into a production environment we would explore other options for deploying our application. AngularJS, Node.js and gulp are all housed on our application server Yoda, which handles everything aside from our database. MongoDB is hosted on our database server Vader. We elected to split our database from our frameworks/utilities in order to reduce resource contention between the entities. The servers themselves are running on the Linux distribution CentOS [5]. We choose CentOS for our servers due to its free and open source nature that we found to be very desirable for our needs in development.

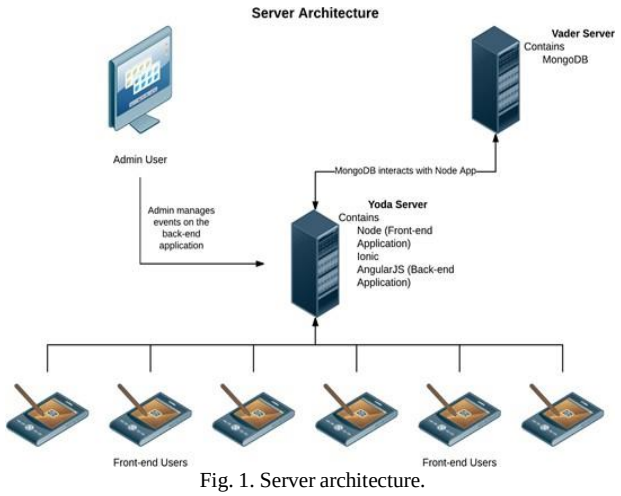

Fig. 1. Server architecture.

III. Front-End

The front-end has two components: Mobile Application and Web Application.

A. Mobile Application

It was the aim of our mobile application to provide an interface for interacting with and viewing geo-based event information for our user's, in our use case, college students on our campus. The mobile application provides the GUI (graphical user interface) for our users to view a map of their current location as well as see events pertaining to their current location, for example events pertaining to the student center on our campus should the student be within the student center's polygon. The mobile application tracks the user's GPS location via their phone's current location. These coordinates are then compared against the building polygon coordinates in our MongoDB. Once our user crosses the threshold of a polygon, in our case a building polygon, events related to that building are displayed to the user. Furthermore, once a user arrives on a certain floor of the building events pertaining just to that floor will be shown. The user's Z coordinate (height) is used to determine what floor they're on.

The front-end framework used for the mobile application was AngularJS along with the Google Maps API for handling any geographic information passed to the application, such as the coordinates that define our building polygons and tracking of the users location. Below in Fig. 2 a screenshot of the GUI is provided which shows an event and the user displayed on a Google map next to a polygon. In this particular example the building is the Computer Science building on our campus and the user is being displayed event information about an Introduction to Programming Session being hosted at the building at 3:45pm. The interface makes note that the event is not yet active. We strove to keep the layout of the mobile application as simple and as user friendly as possible. Only the information that matters to our user is displayed, namely their position on the map and any events taking place at their current location. For future expansion on the mobile application, features such as saving events to an in-app favorites folder and the ability to automatically add events to the built in calendar of the user's phone would be desirable in order to make the application more production ready.

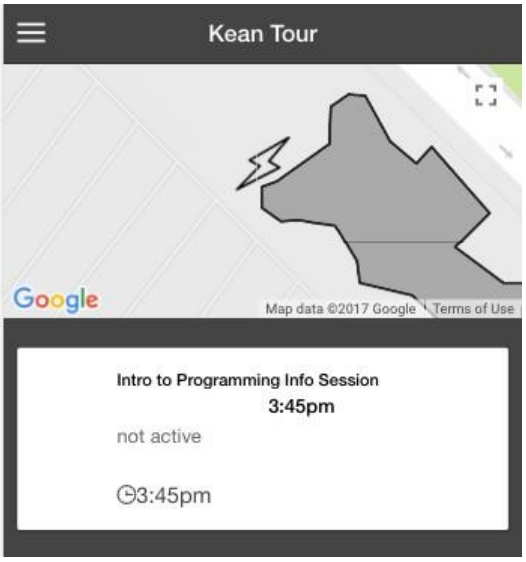

Fig. 2. End user mobile application.

B. Web Application

Our web application's job was to provide an environment for system administrators to enter event information into the MongoDB using an easy to use form that could be used by any faculty member assigned to a systems administrator role for their particular department. We elected to create a GUI for system administration instead of relying on database administrators entering new information manually as the overall goal of our application was the widespread use across our campus. Assigning a database administrator to enter all events for all

departments seemed impractical, where with this GUI anyone could enter information for an event in his or her department. Events entered into the form would be displayed to our mobile users. The web application allows our system administrators to pick a date, time, building, floor and room for the event they are entering as well as being able to enter a title and provide information about the event for the user to see. The web application also displays all events currently in the database to our administrators. This was in order to provide a simple means of giving system administrators the resources they needed to prevent them from entering duplicate events into the database and also to provide a means for scheduling events without conflicts. In Fig. 3 a sample event is shown being entered into the event data form. Following in Fig. 4 the table of events currently in the database is shown, with the event from Fig. 3 being displayed.

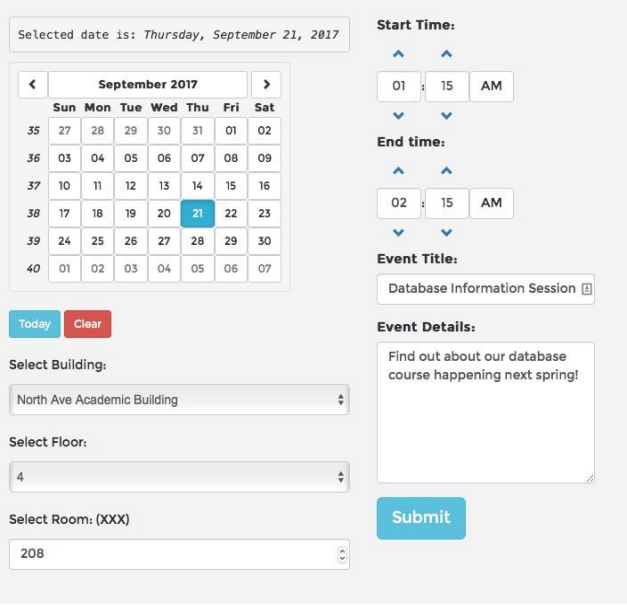

Fig. 3. Event data form.

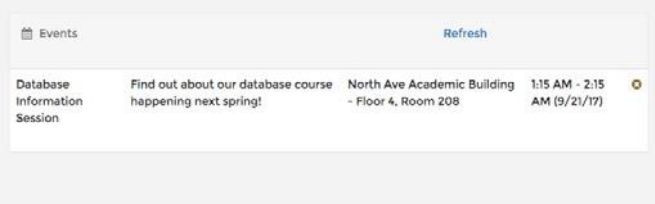

Fig. 4. Database events display.

Our web application, like our mobile application, was built in AngularJS and specifically for the web application the rdash-angular dashboard (an angular implementation of the RDash admin dashboard) was used as a base to build our dashboard up from [6]. The majority of the default functionality of the RDash admin dashboard was cleaned out and a custom form and database display table were created to better fit the needs of our use case. What was left from the initial RDash admin dashboard was the styling and sidebar. In Fig. 5 the home page of our application is shown along with the RDash styling. The name of our application SKIIwalker-web is an acronym for Simple Kean Interactive Information.

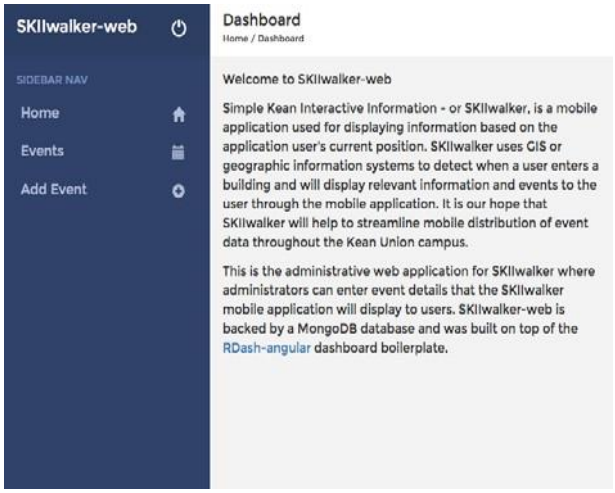

Fig. 5. Dashboard home.

For future expansion of the web application, should the project move into a production environment, an interface for mapping and creating new polygons would be planned in order to offer full administration of the application from the web-app. As of now the only means for adding new polygons to the MongoDB is for a database administrator to manually create the polygon and pass the coordinates into the database. In a production environment this would be unacceptable, but for the purposes of this pilot program we deemed fleshing out a polygon creation GUI to be unnecessary as we are only using a single polygon for piloting the program within our department.

IV. Back-end

The back-end has two components: Database and Application Server.

A. Database

Along with proving out the usefulness of a GIS mobile application the other aim of our project was to explore the use of NoSQL databases in development. In order to store our geographic and event information MongoDB was chosen. Due to the flexibility of the JSON (JavaScript Object Notation) [7] documents MongoDB uses to store data we felt that it would be the best choice for our program as flexibility, ease of development and the ability for easy future expansion of data models was a requirement. MongoDB also offers specific data models geared towards geographic data, which was a required trait for the database

we were going to use for development due to the nature of the data we were storing. The JSON documents MongoDB uses are a human and machine-readable standard that facilitates data interchange and in the case of MongoDB BSON, a binary derivative of JSON, is used as it offers both the flexibility of JSON with the speed of a lightweight binary format. Though for clarity we will continue to refer to the documents MongoDB uses as JSON as the BSON documents are only used behind the scenes when encoding the JSON [8].

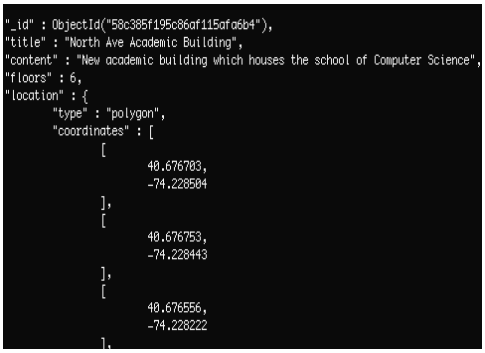

Fig. 6. NoSQL GeoJSON document describing a building and its coordinates.

The layout of buildings is constantly in flux at a college campus due to class changes each semester or even building renovations and new additions, with the flexibility offered by MongoDB's JSON documents we are able to quickly and easily update these documents should the information for a building change. Such as what department is located in that building, or what department's classes primarily take place in that building. In addition to that we have the quick flexibility to add new buildings to the database as they're built and even modify our entire schema to accommodate these new additions should that need arise. Something like this just wouldn't be possible with traditional SQL databases, at least not in a timely fashion [2].

Since we're using a college campus as our pilot program for the application we felt it was necessary that our database provide the flexibility and ease of use that MongoDB offers. Additionally, when compared to traditional SQL databases there were no noticeable degradations in typical database querying speeds, this allowed us the freedom of the flexible data modeling that NoSQL offers without having to sacrifice the traditional reliability of querying an SQL database in a timely manner [9]. Fig. 6 is an example of a typical JSON document used to store information in MongoDB, you can see in this document we are defining the name of the building, information about that building, and all the important GIS information, in this case coordinates, which define the polygon of the building for our mapping API to take care of.

B. Application Server

Our server environment is split into two different systems, we have a database management system housing solely our MongoDB, this is run on our Vader server, and we host our front-end frameworks and utilities, like Node.js, AngularJS, and gulp on our Yoda server. Both of these servers run the open source CentOS Linux distribution.

The database management system (MongoDB – Vader Server) is separated from the rest of the environment to eliminate the resource contention between the application and the database, and to increase security by removing the database from the Application Server (Yoda). Moving forward with this approach, application and database do not contend for the same server resources. This, we found, overall leads to better performance and easier development.

V. Conclusions

We found in our research that for our purposes the flexibility offered by a NoSQL database, in our case MongoDB, proved to be more advantageous over the more common SQL database. As well as offering the flexibility of JSON documents MongoDB [10] also offered easy scalability and schema change should we need to expand the database in the future for more buildings, different layouts, or different locations entirely such as different college campuses or even museums, to offer a new use case.

Overall this research opportunity has allowed us to better understand the practical uses of GIS information and its relation to providing users with real time information about their environment. We have constructed a proof of concept mobile application that utilizes GIS information to help users find relevant information based on their location and display it to them in a way that is quick and convenient. In addition to that we effectively explored NoSQL technology and its benefits and tradeoffs for our application. It is our hope that in the future this application will be improved upon to make it production ready for use on our campus full time, as well as on other campuses and in similar situations where such an application would be useful.

References

[1] J. Han, E. Haihong, G. Le, and J. Du, "Survey on NoSQL database," in *Proc. 2011 6th International Conference on Pervasive Computing and Applications*, Port Elizabeth, 2011, pp. 363-366.

[2] MongoDB and MySQL. [Online]. Available: https://www.mongodb.com/compare/mongodb-mysql

[3] S. Venkatraman, K. Fahd, S. Kaspi, and R. Venkatraman, "SQ L Versus NoSQL movement with big data analytics," *International Journal of Information Technology and Computer Science* (*IJITCS*), vol.8, no.12, pp. 59-66, 2016.

[4] Gulp automation toolkit. [Online]. Available: https://github.com/gulpjs/gulp

[5] CentOS linus distribution. [Online]. Available : https://wiki.centos.org

[6] RDash rdash-angular dashboard. [Online]. Available: https://github.com/rdash/rdash-angular

[7] L. Bassett, *Introduction to Javascript Object Notation: A To-the-Point Guide to JSON*, O'Reilly Media, 2015.
[8] JSON and BSON. [Online]. Available: https://www.mongodb.com/json-and-bson
[9] Z. Parker, S. Poe, and S. V. Vrbsky, "Comparing NoSQ L MongoDB to an SQL DB," *Proceedings of the 51st ACM Southeast Conference (ACMSE '13), ACM*, Article 5, New York, NY, USA, April 4-6, 2013.
[10] G. Fourny, M. Brantner, and F. Cavalieri, "JSONiq: The SQL of NoSQL," *CreateSpace Independent Publishing Platform*, 2013.

Jonathan Rodriguez was a senior student at Kean University in Union, New Jersey and graduated in 2017 with a BS in computer science. He is currently a software developer in the financial industry working on legacy applications and next generation architecture.

Anthony Malgapo graduated from Kean University with a B.S. in computer science information systems and was a senior at the time of writing this paper. He is currently a systems engineer for an IT firm mainly working on infrastructure architecture and network monitoring.

Jacob Quick was a senior student at Kean University in Union, New Jersey and graduated in 2017 with a BS in computer science. He is currently working as a software developer specializing in database design and administration in both NoSQL and SQL as well as back-end web development.

Ching-yu Huang is an assistant professor of the Department of Computer Science at Kean University since September 2014. Dr. Huang received a Ph.D. in computer & information science from New Jersey Institute of Technology, Newark, New Jersey, USA.

Prior to joining Kean University, Dr. Huang had more than 16 years of experience in the industry and academics in software development and R&D in bioinformatics. His research focuses SNP genotype calling and cluster detection; image processing and pattern recognition, especially in microarray and fingerprint; geotagged images and location information reconstruction; database application development; data processing automation; E-learning, educational multimedia, methodology, and online tools for secondary schools and colleges. Dr. Huang has more than 40 publications in journals and conferences and more than 20 presentations in workshops and invited lectures.