Oregon State University

CS 461

Fall 2017

Tech Review AgBizClimate

Author:
Thomas Noelcke

Instructor:
D. Kevin McGrath
Kirsten Winters

Abstract

The purpose of this document is to research and consider different technical options for our application. In this document we research different options for data storage, HTTP request frame works, and testing frameworks. I will consider three possible choices for each section of the application. For each of these options I will weight the pros and cons of each. After comparing the different options I will select the option I would like to use for the AgBizClimate application.

1 Data Storage

1.1 Overview

For the AgBizClimate©application we will need a way to store data so it can be easily retrieved later. For this application we will store a variety of data including budget data, weather data, and user information. This information will need to be quickly recalled so it can be used in our application. Generally, we will want to select a data storage option that will be easy to set up and allow us a lot of flexibility with what kind of data that can be stored. We will also want a data storage option that will quickly recall stored data so it can be used by the application.

1.2 Criteria

To determine the best choice for our application i will analyze the performance of the data storage based on, Ease of Development, and Ease of Set Up and Flexibility. I will analyze run time speed however, this will not be one of the criteria as it is not critical that our data is retrieved as quickly as possible. Generally, its much more important to consider Ease of development, Set up and flexibility because the primary concern is being able to get the application up and running quickly. Ease of development and ease of set up will be a subjective measures for each option. In those sections I will use the opinion of other software developers along with my own experience to compare each option.For these criteria I will rate each option on a scale from very easy to very hard. Flexibility will be a measure of how easy it is to store different kinds of data using each option. For Flexibility I will measure each option from flexible to rigid.

1.3 Potential Choices

1.3.1 PostGreSQL

PostGreSQL is an open-source relational database management system. PostGreSQL uses the Sever Querying Language (SQL). PostGreSQL uses the relational database model. This model sets up tables that represent a certain type of data. We can then run SQL quires on this data base to get the data we need for our application.
SQL databases are great for run time performance. PostGreSQL will have superior run time performance. Generally, If you are looking for a way to store and retrieve data as quickly as possible PostGreSQL is one option you should consider(Citation needed).
Though SQL is quick, In my experience it is not as developer friendly as the other options in this analysis. Generally, writing raw SQL queries is difficult and time consuming. This is especially true when your data models are very complicated. Using Raw SQL also requires the developer to manually figure out how to make the mapping between the database and the models used at the application level.
Another problem with SQL in my experience is that it is more difficult to set up. The configuration process can be complicated. Additionally, if you choose to use raw SQL you must also set up your data base as third party application separately from your actual application.
Another problem with SQL is that it is rather rigid in the way that you must store data. For example if you want to store a list object in an SQL data base you must create a new table where one row represents one item in the list. This item must have its own unique id. Additionally, if you want to have a list of lists it gets even more complicated. Now you need to create another table to represent a name and ID for each list you want to store and you must relate every item you want to put in that list back to the parent item in another table. This can get very complicated in a hurry.
Another problem is if you don't know what the structure of the data is going to look like before run

time it is impossible to store this data in an SQL database. This makes PostGreSQL rather rigid in terms of flexibility.

1.3.2 Python ORM

Python ORM or Object Relational Mapper does not substitute for an SQL database. There will still need to be an SQL database running on the back end. However, the ORM framework allows developers to create objects in python that then map to the data base. Often times this type of frame work allows the developer to create the objects first and let the framework deal with creating the SQL database on the backend. Given that this is not a replacement for an SQL it does make working with an SQL database much easier.
This type of frame work has several advantages, it allows for easy development and set up. The ORM frame work allows the developer to be completely insulated from the SQL data base on the back end. This means that instead of writing SQL queries to get data from the database the developer is able to use objects in python to access data that is stored in a database. This makes development and set up Easier on the developer. This is because the developer doesn't have to worry about writing complicated SQL statements or setting up complicated relationships between tables. This frame work also allows the developer to develop the code first and let the framework worry about creating the database the data will ultimately be stored in(citation needed).
The ORM framework also allows for great flexibility in what you can store in a database. This type of frame work allows for mappings between python objects and the data base. So nearly anything you can store in an object in python, you can also store in a database. However, it should be noted that you must know the structure the object you are trying to store before run time.
Though this approach is very easy for the developer it isn't with out cost. The ORM approach does take a hit in terms of run time performance. The ORM framework will be slower than raw SQL. Another problem with this type of framework is that it doesnt leave the developer very much control over the database. This means that you are stuck with what you get. If the frame work structures something in the database in a way you don't like you don't have a ton of choice about that. Additionally, making database chances can cause you to loose data in the database if you are not careful about how you handle the migrations (citation needed).

1.3.3 MongoDB

MongoDB is an open source database NoSQL database. This means that instead of using tables and rows like an SQL relational database, MongoDB uses collections and documents. Documents are individual data members where there is a key value associated with each document. Collections contain multiple documents. Collections allow the developer to store many items in a database. This structure allows for the structure of the data being stored to be determined after run time. This allows for greater flexibility because you can store dynamically generated objects (cite source here). MongoDB is also fairly simple to set up. This is because of the way that objects are stored we don't need to set up and complicated tables. We can simply set up our data base and insert the data. This also means that if we want to change the structure of the data down the road we don't need to make massive changes to the data base. This makes MongoDB much more developer friendly and also very flexible (Cite Source).
However, It should be noted that the usability and flexibility of MongoDB is not with out cost. The biggest cost of the flexibility of MongoDB is that it is harder to relate two items in a database. For instance if want to have one entity that represents a user and another that represents a user role, this becomes a challenge in MongoDB. There are work around and ways to solve this problem however it should be mentioned that this is a problem. Another trade off in using MongoDB read and write operations are generally asynchronous. This means that you can tell the database to do something and then move on other tasks and the data base will return the result of your query later.

On the surface this sounds nice but if you are doing a lot of reading and writing operations in the same block of code this can cause problems. For instance if you write several items to the data base and then need to read items out that depend on those items you just added you may get errors because the first write hasn't finished yet(cite here). Finally, MongoDB may also be much slower for some operations than traditional SQL. One example where this becomes apparent are aggregate functions where you want to preform some sort of manipulation of the order of the data (Source here).

1.4 Discussion

As mentioned earlier we can divide the types of data we need to store into two distinct groups, weather data and user data. The user data will generally have relationships between different parts of the data. For instance a user will have an address and a phone number. The user will also have various different budgets. These relationships make the data easier to store in an SQL database such as PostGreSQL. To farther simplify storing the data we could also use Python ORM. This would allow us to create relationships in the data, easily develop the database and store the data. The weather data on the other hand is much more like a large list. This sort of data can be sotred in an SQL data base such as PostGreSQL however, this would make the development of the project more difficult. A more suitable choice for storing the weather data would be MongoDB. This is because MongoDB allows for easy storage of lists with little complexity than PostGreSQL. MongoDB provides the easiest and most flexible way to store the weather data.

1.5 Conclusion

For this project we will use PostGreSQL and MongoDB. We decided to use these frame works at the request of our client.

2 HTTP Request FrameWork

2.1 Overview

The *AgBizClimate* project will need a way for the client on the front end of the application to communicate with the API at the back end of the application. We have already determined that we will use HTTP request to facilitate this communication. However, we have not decided which framework we should use to make the HTTP requests with. Generally, the frame work we use to make HTTP requests should be easy to use, easy to read and allow us to quickly right requests.

2.2 Criteria

For the purposes of this analysis we will consider

2.3 Potential Choices

2.3.1 Ajax

2.3.2 XML Http Request

2.3.3 choice 3

2.4 Discussion

2.5 Conclusion

3 Testing Frame Work

3.1 Overview

For the AgBizCliate system we will be required to write unit tests for our code. In this section I will discuss the various testing frame works we might use for our project. I will restrict the conversation to frameworks that we can use in Python as parts of our clients application are already in python. It is important for use to write tests for our code for several reasons. Firstly, We will want to ensure that our code actually does what we intended. Secondly, We will want to make sure that when we make changes we do not break our code. Thirdly, Our unit tests will become useful when we want to do integrated builds for our project.

3.2 Criteria

3.3 Potential Choices

3.3.1 Choice 1

3.3.2 Choice 2

3.3.3 choice 3

3.4 Discussion

3.5 Conclusion

4 References