Orange Dating: Developing Modern Java Web Applications

William Mercado Millán
Master of Engineering in Computer Engineering
Alfredo Cruz, Ph.D.
Computer Engineering Department
Polytechnic University of Puerto Rico

Abstract – *Developing a Java web application can be considered too complex for most modern developers, mainly because Java has a reputation of being difficult and requiring lots of coding. However, there are modern tools and frameworks that are designed to facilitate the development of web applications in Java. These frameworks are focused on reducing the amount of code needed to make the basic functionality of an application. One of the frameworks used in this project is Spring. This framework provides a MVC architecture designed to facilitate the creation of a modern Java web application with a minimal configuration and code. This document describes an online dating system that was developed in a short time, using Spring framework and Semantic UI to achieve the goal of developing a whole system in a limited amount of time. The online dating system suggests friends based on answers to a questionnaire after the registration. The goal of the system is to make accurate suggestions to a user that is interested in identifying an ideal partner for a date.*

Key Terms – *Java, Online Dating, Spring Framework, Web Application.*

INTRODUCTION

Developing an online dating system can be very complex and could take a lot of time. This could be a challenge due to the limited amount of time and resources available. However technology offers many ways to solve these problems. Part of the focus of this project is to use the technologies offered in the Java environment to create web based applications in a short time. To achieve this goal, a careful study of new technologies has been done in order to create the full functionality of a complete web application, reducing the amount of code needed to a minimum.

In the past, creating Java web applications would have required a lot of technical knowledge and time, but now there is the advantage of modern frameworks which reduce the amount of code and configuration, allowing the developer to focus on the design and not so much on coding. Reducing code and configuration has been a crucial step in the creation of this system and it has been the key for completing the development on time, in a very tight schedule.

SYSTEM OVERVIEW

The name of the system is Orange Dating and the word orange comes from the "half orange" term in Spanish that it is used to refer to the ideal partner for a person. The system was created by studying different online dating sites and taking the best features of each of them as the requirements for this system. The system was developed using Java and the enterprise framework Spring. As mentioned before, Spring provides a series of tools that help to reduce the amount of code, and it also provides a MVC architecture for web applications [1]. One of the objectives of the development is to demonstrate how Spring can facilitate the development of web applications in Java. The system was developed according to Spring's best practices on how to create a modern Java web application.

The system was developed on the basis of other existing sites. One of the sites that was evaluated is eHarmony, which is one of the top online dating sites available [2]. It was carefully studied and tested; and some of its good features were also implemented in the system.

SYSTEM REQUIREMENTS

The system's requirements consist of the list of functionalities that the system will have. This section describes the functional requirements for the system developed.

Questionnaire

After a new user completes the registration, to know the user better the system will be asking the user a series of questions, this questions will be used by the system to determine some aspects of that user so that the system can suggest contacts for that user. Questions can be configured in the database, the amount of questions can be configured as needed. Each question has a category, when the user answers the question the category score will be increased. Questions can be negative or positive, if it's a negative question it means that the answer to that question will impact the category in a negative way, if the question is a positive question it will impact in a positive way the category. The current categories are: romantic, organized, passive and religious. Each category has a set of questions that will deter-mine the top categories for that user. If the user is determined to be romantic and religious, it will be matched with other users that are also romantic or religious.

Questions and categories can be configured in the database, the amount of questions can be increased as well as the amount of categories, and this can provide a more accurate result of the answers. But for the purpose of this project, we will limit the amount of questions to 28 and the categories to 4.

Recommendations

One important requirement for this system is to be able to suggest or recommend new friends to users. Our users are looking for their perfect match, and when users try to find their match in other places the match might be full of surprises. Orange Dating system will provide accurate recommendations to users based on the questions that were answered. In order to suggest a match, the system will first determine the place of residence, if the user lives in San Juan the system will try to find matches that are from San Juan. The system will also match the preferences that were specified in the registration, if the user wants to be matched with females from a range of 18-24 years old the system will take that into consideration as well. From all the matches that are found after the initial filtering the user will only be recommended with the users that are a match for him based on the answer to the questions that were asked when the profile was created.

Profile

The profile is the home screen when the user logins to the system, the profile screen will have the details of the user, the details that will be shared to other users. These details can be easily updated by clicking the pencil icon. Anything that is displayed in the profile can be updated from the users profile screen. For example smoke and alcohol preferences, maybe the user started smoking again and wants to update that field so that other users know, or maybe the user doesn't want to drink more alcohol and changes its preference so that other users are well aware.

Subscription

Orange Dating system has subscription business model, in which the actions the user can make are limited by the features that are included in the subscription that the user selected. Users are able to register and use the system for free so that they can have a glimpse of the features that the system has. In the free subscription the system will ask the questions to the users and will display recommendations based on the answers from the user but the user will not be able to see the suggested user profile picture. The only information that will be available to the users is the name, age and city.

The purpose of the free subscription is to engage the users and motivate them to upgrade a paid subscription. There are two types of paid subscriptions *Premium* and *Unlimited*. Premium subscription will open the main features of the system to users the users will now be allowed to see the profile picture of other users, visit their profile and have a vote in the attractive score. Basically most of the important features are available in this subscription however one important feature will be left for the unlimited subscription and that is the messaging system only users that are subscribed to the unlimited subscription will have the ability to send messages to other users. Unlimited users will have the full functionality of the system.

Subscriptions are charged monthly and users will have the option to upgrade to a greater subscription at any time. Once the subscription is charged it will be available for one month.

Messages

Messages will be the mechanism available for users to engage in a conversation with another user but this feature will only be available to unlimited subscription users. The message feature will allow users to write a 160 character message to another user in order to engage in a conversation. A user can only send a message to another user once, and only after the other user replies the user can send another message. This will avoid spam. Messages will help users to get engaged with the suggested users that were matched based on the questions answered. The only users that will be available to send messages are those that matched based on the answer to the initial questions.

Attractive Score

The attractive score is based on rating that users give to other users when they visit there suggested user profile, this score will tell how attractive the user is. The attractive score will be the average number that most people score for that user. Users are limited to only rating other users once. The purpose of the attractive score is to let other users know how attractive that particular user is to other users. To achieve a higher score users should write interesting information on their profile.

Functional Design

This part of the document describes the functional part of the system. Basically what will be described next are the screens designed to fulfill the requirements of the system that were described in the previous section.

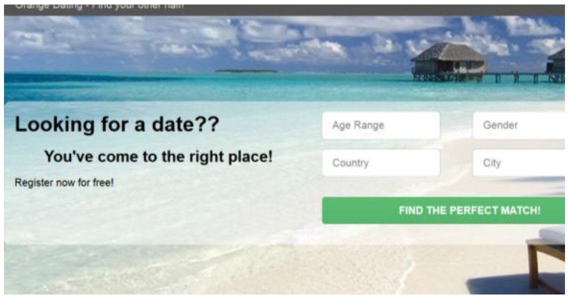
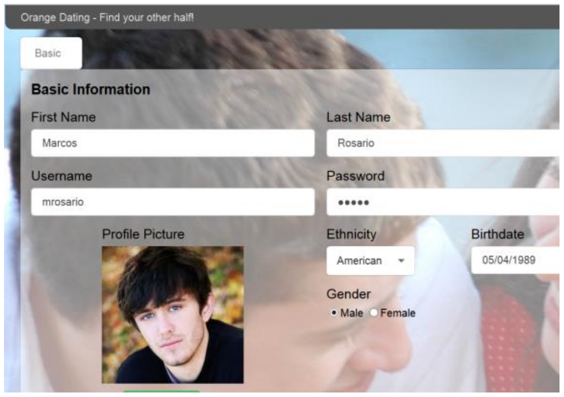


Figure 1
Home Screen

The home screen in Figure 1 was designed in an attractive way so that new users can be engaged to join Orange Dating system. The page suggest that a user can search for friend right away but it's only a catch up to get users to register. The suggestion for your area is an example of the real suggestions that the system can make when the registration is done and the questions are answered. The suggestions area in the home screen was designed to detect the IP address of the user and display random profiles that live near the physical location of the IP address.



Basic Information

First Name: Marcos
Last Name: Rosario
Username: mrosario
Password: •••••
Ethnicity: American
Birthdate: 05/04/1989
Gender: [✓ Male] Female

Figure 2
Registration

The registration wizard will be gathering the basic information of users including the profile picture which can be uploaded in the first step as seen in Figure 2. The information obtained from the registration process will be later used to fine tune friend suggestions and will also obtain information about the subscription and payments methods that the user wants to have.

After the registration process is completed the user will be sent an email to confirm the registration. The activation code will have a time limit for the user to activate its profile. If the email was confirmed on time the profile will be activated and the user will be able to log in to the system, otherwise the system will not recognize the profile as a valid one.

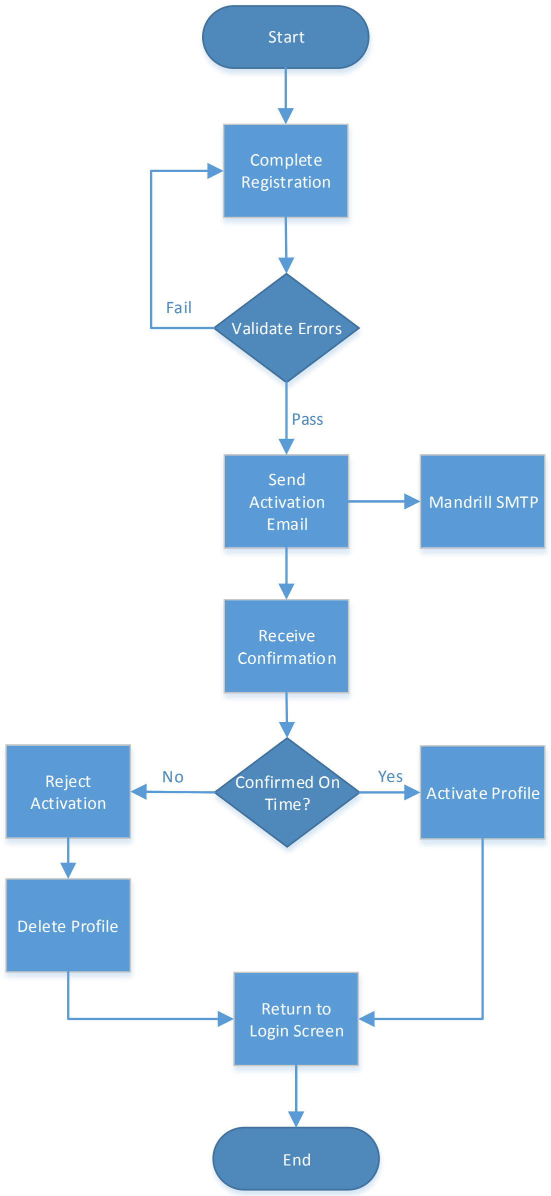
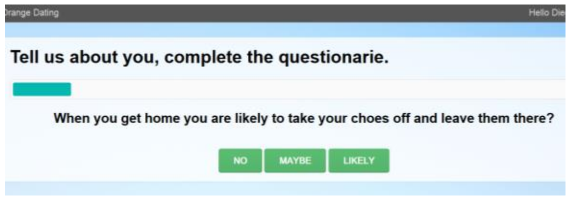


Figure 3
Registration Flow

Figure 3 shows the registration flow logic that is followed after the user has finish the registration process, the registration flow is designed to avoid bots creating fake accounts, as there is a need for human intervention to activate the profiles

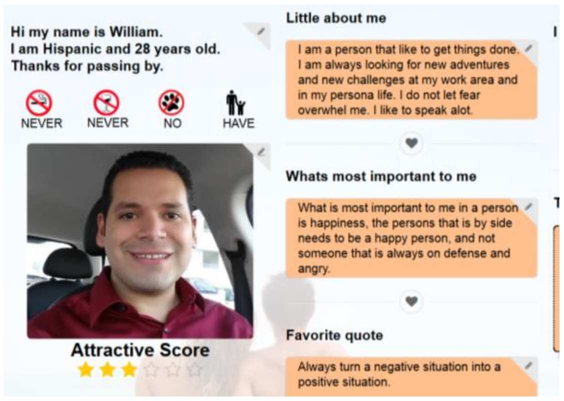


Tell us about you, complete the questionarie.

When you get home you are likely to take your choes off and leave them there?

Figure 4
Questionnaire

The questionnaire is the set a questions that is presented to new users when they register in system. This questionnaire will help the system determine which of the recommended users is more compatible with the user. The goal is that the recommended users be a compatible match, because they would both be on the same category of thing they like to do. Figure 4 is presenting an example questions on how the questionnaire is presented to the user. The user has 3 options as answer and depending on the answer chosen the amount of points that will be added to each question category. If the user select answer *No* and the question is positive would mean that no score will be added for that category. If the user answer is *Maybe* then only 1 point is added to the category. If the user answer is *Likely* and it's a positive questions the score that will be added is 2 points. If it's a negative question the *Likely* answer would then be zero points and the *No* answer would be 2 points. There are currently 20 questions configured that the user has to answer, and the questionnaire will display to the user the progress of the questions with a progress bar. If the user gets tired of answering questions, the user can exit the system and then come back and continue answering the questionnaire. The system will force the users to answer the questions before they can see their profile, because it is based on the answers that are given in those questions that the system is able to recommend suggestions for that user. The goal of the system is to make accurate suggestions and that is why the system must ensure that the users answer the questionnaire before they can continue to their profile.



Hi my name is William.
I am Hispanic and 28 years old.
Thanks for passing by.

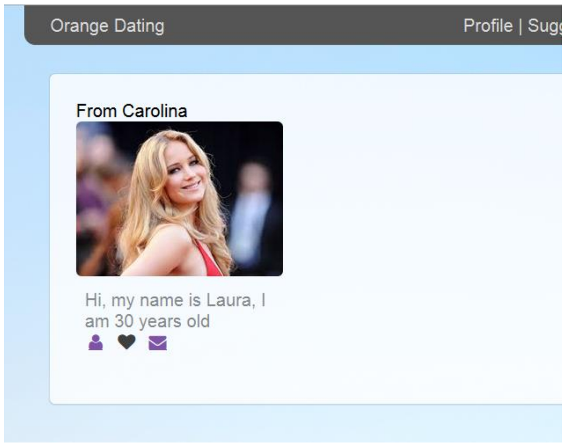
I am a person that like to get things done. I am always looking for new adventures and new challenges at my work area and in my persona life. I do not let fear overwhel me. I like to speak alot.

What is most important to me in a person is happiness, the persons that is by side needs to be a happy person, and not someone that is always on defense and angry.

Always turn a negative situation into a positive situation.

Figure 5
Profile Screen

Figure 5 presents the profile screen that is the main screen presented to the user after logging in and answering the questions. This screen will display the information that was gathered during the registration plus some additional information that will be set when the user arrives at the profile screen for the first time. Every field displayed in the profile screen can be edited by clicking on the pencil icon in the right corner of the object. The profile picture can also be updated in this screen.



Hi, my name is Laura, I am 30 years old

Figure 6
Suggestions Screen

Figure 6 shows the suggestions screen that will be displayed after selecting suggestions from the top menu. The users that appear in the suggestions page were specifically matched by the answer to the questions of the questionnaire. From this screen the user can see the suggested user profile or send a message to that user to initiate a conversation. The user can also visit the suggested friend profile to read all the information that the user published in the profile and has the opportunity of rating the user to affect the attractive score of that user. The user will only be allowed to score in the attractive score once.

TECHNICAL DESIGN

The technical design section consists of a description of the implementation of the system, the section includes information about the programming tools, frameworks, and architecture of the system.

Programming Tools

Programming tools are used to ease the development of the application and if used properly they can be beneficial to reduce the development effort to provide faster results. The tools used to develop the system will be described next.

Spring Tool Suite

Spring Tool Suite abbreviated with STS is an integrated development environment (IDE) provided by Spring that includes a set of tools for Java developers using Spring Framework. STS is based on Eclipse meaning that all Eclipse plugins are compatible and also comes pre-loaded with many important plug-ins already installed [3]. STS comes with many features out of the box and one of the features that is being used in this project is Maven. STS includes an embedded version of Maven which is necessary for development of the system. Maven is used to resolve dependencies and using a combination of plugins can be used to create *Jar* and *War* files. [4].

Spring Boot

This project was developed using the latest technologies from Spring. Spring contains a set of sub projects that are designed to address specific programming issues. One of the most important project of the new version of Spring is Spring Boot. Spring Boot project is focused on the reduction of configuration needed to create and run a project, it also removes the requirement of using a separate application server. Out of the box Spring Boot comes preloaded with Apache Tomcat latest stable version. Spring Boot projects can be executed from a single *Jar* file [5].

JRebel

One of the biggest problems with Java development is the constant need to restart the application server to load the new changes. Restarting the application server could take up to one minute. Every change that is done in Java files require an application server restart to take effect. So this basically adds a lot of time to the development. For this reason there is a product called JRebel. This product is focused in avoiding constant restarts on the application server and thus reducing the development time [6].

JPA and Hibernate

Spring Data is the Spring project used to connect to different data sources. It is an abstraction layer that relieves the developer from having to implement specific implementations for popular databases like MySQL and Oracle. Java Persistence API (JPA) is a Java standard used to map Java objects to database tables. JPA is implemented with annotations which are used to specify how an object will be mapped to a table in the database. So basically the complete database structure can be created by combining different Java objects and mapping them with tables using JPA annotations.

Finally, after the objects have been mapped, Spring uses Hibernate to interact with the database. Hibernate is an Object Relational Management (ORM) tool that understands JPA standard [7, 8]. Hibernate can be implemented alone or can be implemented with Spring. The project was implemented using the Spring project called Spring Data that adds a layer that abstracts the implementation of Hibernate. Abstracting Hibernate allows flexibility, if for some reason the framework needs to be replaced in the future, the abstraction helps reduce the code impact. One good feature that Hibernate has is that it can create the schema of the database based on the domain objects that defined the database and that are mapped to tables using JPA annotations.

Spring MVC

Spring MVC is part of the Spring and it is very easy to use. It is prepared to work with redirects by mapping REST addresses to actions methods. It sup-ports the different REST operations like (POST, GET, PUT, DELETE) for example. It also provides ways to upload file to a controller and handle Multi-Part parameters. With an additional library Spring MVC support JSON request and responses. The controller can receive a JSON string and convert it to a java object which is very useful when dealing with jQuery requests.

Application Architecture

The system was designed following the recommended architecture for web applications in Spring. Spring follows the MVC architecture and also follows a three-tier architecture in which the layers are: Presentation, Logic and Data. This layers are well followed in system. The upper layer cannot directly communicate with the lower layers. Spring recommends the use of these layers and gives each layer a different responsibility.

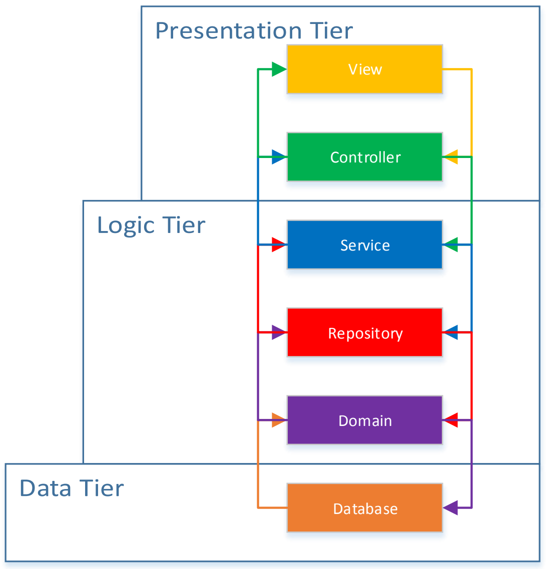


Figure 7
High Level Architecture

Figure 7 shows the high level architecture in place for the developed system. This architecture was done in accordance to MVC architecture and three tier architecture. It is important for the developer to understand this architecture because the flow of data needs to happen in this specific order, and each layer has its own set of responsibilities that must be respected when making code changes. Each tier is explained in detail in the next sections.

Presentation Tier

The tier shown in Figure 7 is the presentation tier and is in charge of handling the interaction with the front end. This tier is very important because it's the face of the application. This tier has the responsibility of handling human interaction with the system and needs to make sure that the information entered by humans is correct. To achieve correctness the presentation tier needs to have a set of validation tools to make sure that user input is correct. Sometimes by mistake and sometimes in purpose, users tend to enter bad information in the input fields, and those fields must be validated before they are processed by the rest of application layers.

It is also the responsibility of the presentation layer to be user friendly. The pages must be intuitive so that users can easily understand the actions intended for them to execute. It is a challenge to make a page user friendly. Sometimes it requires the use of other frameworks that can provide alternatives to simple web pages, like for example a modal window. There are tons of libraries that are designed to facilitate the design of user friendly web pages.

Presentation Tier: View

The view layer is part of the presentation tier. This layer is very important, because it manages HTML documents. That is the front end that users will be interacting with. The view layer can incorporate a set of tools and plug-ins that can add valuable functionality to the system. In the case of Orange Dating a library called Semantic UI was introduced. One of the goals of the development of this project is to reduce the amount of development time. One of the biggest challenges that developers face when they are developing a web application is in the View layer. Creating intuitive HTML pages is a good way to make the application more user friendly. Sometimes in order to make something look exactly the way we need, we are forced to spend a lot of time tweaking CSS, Javascript and HTML files to achieve our goal. HTML was become more much better in these days with the introduction of HTML 5 but there are still some challenges that developers face. Semantic UI css library is designed to be very intuitive for developers, the names states that the library is implemented in a semantic way, meaning that with the use of simple English words, a developer can achieve great looking interfaces [9]. The following code segment is an example of Semantic UI:

```
<%@ include file="header.jsp" %>

<div class="ui three column grid">
        <div class="column">
        </div>
        <div class="column">
           <div class="ui segment">
           </div>
        </div>
        <div class="column">
        </div>
</div>
<%@ include file="footer.jsp" %>
```

The previous code segment presents an example of what Semantic UI can achieve. As noted in the example there are a set of division objects from HTML with some classes specified. These classes are invoking Semantic UI instructions to format our web page the way we need it. The previous code example intends to create a 3 column web site with the content in the middle. To achieve this we declare a division with a class "ui three column grid" meaning that our HTML code will be divided in 3 columns. After declaring this division, we then create three divisions inside, this divisions will have the class of type "column" which means that it's a column of the declared grid. Since we want to have our content on the middle, we are only going to use the center column and leave the other two empty. In the center column we then insert the HTML code we need to display. The “ui segment” div creates a box in which we can put our form or HTML tags that we need.

Presentation Tier: Controller

View are very important, but they have a set of responsibilities that limit their functionality, MVC architecture states that *Views* and *Controllers* need to be separated. To display information the view needs to interact with the server. The most basic example is when an HTML form is created with an input tag type submit. When this button is pressed the HTML page will request the server to process that form and wait for a response. The server must then return the new page to be displayed along with the data.

Spring uses JSP tags to bind data from the controller to HTML pages. JSP tags simplify the work of the controller because the data is received in the corresponding objects. The following code segment shows an example on how to bind an object to an HTML input:

```
<form:input path="profile.username" …
```

This example uses the “form:input”. When the web page is served to the browser, the JSP tags are removed and converted to HTML code so that browser can correctly display. Take a look at the path attribute, this attribute specifies which programmatic object the control is binded to. The previous code example a form input tag is tied to the object profile and attribute username. When this form is submitted it is going to bind the user input for this field to the “profile.username” value, and then it is the responsibility of the controller to take the necessary actions and validations on the field.

Spring MVC makes it very convenient and easy to develop a dynamic website using Java. It is a crucial part of this project to reduce development efforts. Controllers also have the responsibility of receiving the data from the view layer and validating the data before sending it to the next layer, there are many ways to validate the data and Spring has an “Errors” objects that its associated with the object received from the View that can be used to reject the value and return the object to the View.

When data retrieval is needed the controller will need to communicate with the next level which is the Service level. In order to communicate with the service level, the controller needs to include in its definition an instance of each service that needs to invoke, this can get very repetitive if the same service needs to be reused in many controllers so in this case the Service instances were instantiated in the “AbstractController” so that every controller has access to the Services. To instantiate a service Spring provides an annotation called “@Autowired” that basically injects an instance of that component or service in this case. As mentioned earlier the controller is responsible of validating the data that came from the View before sending the data to the next level. When the controller finally send the object to the Service the controller needs to interpret the result of the Service request and take an action on the View. The controller will decide which HTML page will be displayed next.

Logic Tier

The logic layer is where the intelligence of the application lives, this layer is also known as the business logic layer, all the complexity that is not related to the View or user interactions should be managed in this layer. For example if the user tries to login to the application the layer that needs to determine if the credentials submitted by the user are correct is the logic layer which will then return a result to the controller so the controller knows what action to take on the View.

Logic Tier: Service

The service layer is one of the most important layers in the whole application. This layer is responsible of managing the most advanced logics in the system. All business logic and business rules should be developed in this layer. The more deeply we go down the layers the closer we are to the database access, the Service layer has the responsibility of invoking database queries when necessary and taking actions based on the result of the queries to the database. The following code segment is an example of a service:

```
@Service("countryService")
public class DataCountryService
implements CountryService{

       @Autowired
       private CountryRepository
countryRepository;

       @Override
       public Country find(Long id) {
          return
countryRepository.findOne(id);
       }
```

The implementation of "Country Service" as can be noted in the previous code example is responsible of communicating with the *Repository* layer. The Repository layer is very close to the Data Tier which will retrieve the data from the database and provide it to the Service layer. In most cases as noted in the example the Service layer will only be a bridge to jump to the next layer the Repository layer.

Logic Tier: Repository

The Repository layer is an important part of the architecture used in the System. Repository is part of Spring Data and provides an easy and fast way to develop queries in the database. Spring Data handles the connection to the database and converts the result of queries to objects. Spring Data uses Hibernate to communicate with the database a convert queries to object results. Spring Data provides an interface that basically implements the basic CRUD operations needed for an object or table in the database [10]. So by simply creating a Repository object associated with a Domain object the Repository object provides an interface to make queries to the database to the object that is associated with. It is important to note that even though Repositories are interfaces we do not need to implement them, we just use them with "@Autowired" annotation. Repositories have the responsibility of providing basic operations like CRUD and some more advanced queries too, Repositories can be modified to provide custom queries.

Logic Tier: Domain

The Domain layer will contain domain object definitions, in the domain layer there will be an object per table in the database. Domain objects are table representation in objects. Mixed with the domain objects are the some "Enum Types" that are not mapped to tables in the database, and some Many to Many relationship tables in the database are not mapped to a domain object, only if it's needed it will be mapped. Domain objects are mapped with tables in the database using a library called Java Persistence API (JPA) this library provides a series of annotations to map an object to a table. For this project in particular a Hibernate option was used to automatically create the database schema, Hibernate uses the JPA annotations to convert the object into a database table, if the domain object is properly annotated it will create an identical database to the object set. This is another key factor used in this project to reduce development time.

Data Tier

The data tier is basically where the database is. This layered system is designed so that it can be scalable and adaptable. Since we are using JPA standard to define our domain objects, this means that if the database has to be changed from MySQL to Oracle is very simple. The developer doesn't need to change anything, only the driver that Hibernate is going to use to connect to the database. Is that simple.

Database

The database for this project was modeled with object oriented programming and JPA annotations. This is very useful approach to synchronize java code with the database. Hibernate has a configuration to automatically create the DDL by reading the objects defined in code along with the annotations and converting them to a relational database. Hibernate was configured to use MySQL so that it knows how to define the DDL for the database. So basically the developer doesn't need to be an expert in databases just needs to know how the objects are going to be used in code and Hibernate makes sure to translate that into a relational database. There are some known issues with this approach, but in general it works great.

CONCLUSION

The main objectives of this research and development was to prove that Spring Framework and tools like Semantic UI could reduce the develop-ment effort. Spring Framework provides an easy way to develop the backend system and creating a new query to the database would only take a couple minutes with only a few lines of code however there was a downfall, the learning curve was stiff, at the beginning of the development a lot of time was spent learning how to use the frameworks and tools, it took a lot of time to understand the concept of Spring Boot, and learn how to use Semantic UI, however after the setup was completed and the understanding of the frameworks was more advanced, the developed result were increased and to prove the point after the base HTML pages were designed creating the rest of the pages only took a minimal effort and adding new functionality to the system was easier. Spring Framework has improved many thing in their latest release but the documentation is not easy to understand and can be hard for new user.

REFERENCES

[1] Spring Platform, *Spring Documentation*, 2014. Retrieved on April 28, 2014 from http://spring.io/docs.

[2] eHarmony, *Company Overview*, 2014. Retrieved on May 5, 2014 from http://www.eharmony.com/about/eharmony.

[3] Spring Platform, *Spring Tools*, 2014. Retrieved on May 25 2014 from http://spring.io/tools.

[4] Apache Maven, *What is maven?*, 2014. Retrieved on April 15, 2014 from http://maven.apache.org/what-is-maven.html.

[5] Spring Platform, *Spring Boot*, 2014. Retrieved on April 23, 2014 from http://-projects.spring.io/spring-boot/.

[6] Zero Turnaround, JRebel, 2014. Retried on April 15, 2014 from http://zeroturnaround.om/software/jrebel/.

[7] J. Community, *Hibernate ORM Documentation*, 2014. Retrieved on April 20, 2014 from http://hibernate.org/orm/documentation/.

[8] Hibernate, *What is ORM?*, 2014. Retrieved on April 16, 2014 from http://hiber-nate.org/orm/what-is-an-orm/

[9] Semantic UI, *Overview*, 2014. Retrieved on April 15, 2014 from http://semantic-ui.com/introduction/overview.html.

[10] Spring Platform, *Spring Data JPA*, 2014. Retrieved on April 25, 2014 from http://projects.spring.io/spring-data-jpa/.