Cognitivity - The Backend

Introduction:

The previous week it was decided among the team that I - Ophir Katz will take responsibility on the investigation on the backend logic of our software.
As I was scoping the web for information about which information I need to collect in order to create this report, I realized I actually have two types of jobs here, that both need to be aligned with the works of Daniel and Guy, since our duties are somewhat intertwined.

1. I need to find a backend technology that is implemented in one of the allowed programming languages that fits the “backend/server side” logic, which has great features for development, and **can be used to talk to the frontend technology** that Daniel will choose to be the best (for now, it seems that angularJS is the best pick for frontend).
2. I need to find comfortable ways to use different types of libraries that help talking to a database. Since we will most likely be using some sql database, the programming language I’ll use for backend **must** supply easy enough ways to use this type of db. This should be verified with Guy, since he is working on deciding which type of data server we will use eventually.

As the most commonly used language for backend in web applications is **Java** (among the allowed PL’s), this report will mainly focus on that language. Other language (such as C#) will be covered at the end of this report, if Daniel covers a technology that requires it (such as ASP).

Java technologies for developing the backend side:

Spring Web MVC (or just Spring MVC) is the original web framework built on the Servlet API and included in the Spring Framework from the very beginning. The Java Servlet API is just a standard for implementing Java classes that respond to requests. It is designed in a way that web based application (which naturally involve a lot of request based web page changes) could use it very easily. The Spring MVC framework allows for extensive configurations, either using the web.xml file or even via the Java code. Moreover, the MVC module provides default configuration (that can be changed easily) suitable for most applications. Spring offers a neat way to define REST clients for transferring data between the web application and the server side (for various purposes, such as storing data, or handling web pages requests). Spring MVC also offers a very comfortable way to test the entire backend, using Mocks, which is incredibly convenient for testing Java classes.

Spring's modularity for use with AngularJS:
First of all, as I stated before, Spring is entirely modular. This makes it possible (and easy) to program the backend logic without regard to the web technology used for the UI.
You can define controllers, adapters and more objects to implement logic of the backend, all in Java, while the configuration of the web technology used, is only done in the xml configuration files. For instance, declaring that a certain service of the UI is implemented in a file called service.js (or service.ts), can be easily be done by inserting a <script> tag in the corresponding .jsp file (like homepage.jsp, for example).

Integrating with Angular's services is also easily done: Spring MVC defines a controller, which defines interfaces for conversing with a frontend, and Angular can define how to use these interfaces when implementing different types of services, such as pulling data from a data server (which is done via the server side), or requesting a web page, and so on… The RESTful services Angular calls are also integrated in the Spring framework, so it makes for an even easier development (because the design is already implemented).
The way it is done is by defining interfaces and then implementing these interfaces as services to offer from within the controller class. The controller is accessed from Angular.

Learning curve with Spring MVC:
As stated, this is just a Java library, so it is written in only Java, using some configurable content in xml. So, All we really need to know is Java, and to read a lot on the framework (it has alot of features, but what's good is that we probably won't use them all).

Pro's

- Spring MVC is FREE!
- It is highly used in building web applications. Actually, it is the most used in Java, grabbing over 40% of the market.

- It is written in only Java - easy to learn + all of the team mates already know a portion of Java (as a prerequisite to this course).
- It has a lot of guide online, and a bunch of examples for building web applications, even using AngularJS for frontend.
- As the most used library out there, it is very unlikely that it will stop being supported. Since it is also open source, that is probably impossible.

Java's way to interact with a SQL data server:

Since we are definitely using a SQL data server, Hibernate ORM is a great option:
Hibernate ORM (Hibernate in short) is an object-relational mapping tool for the Java programming language. It provides a framework for mapping an object-oriented domain model to a relational database.
The ORM model is THE way to use any relational data servers, for creating data, and reading from it. It interacts with such servers using JDBC (= Java Database Connectivity is an application programming interface for the programming language Java, which defines how a client may access a database. It is Java based data access technology and used for Java database connectivity).
Hibernate was designed to work in an application server cluster and deliver a highly scalable architecture, which answers, in theory, any questions about number of users the application will have, regarding access to the data server.
Just like Spring, Hibernate is highly configurable.
Hibernate is well known for its excellent stability and quality, proven by the acceptance and use by tens of thousands of Java developers, which means it will be supported for a long time..
Hibernate is database agnostic. It doesn't care if you use Oracle, SQL Server, MySQL, or about a dozen other relational databases. This means that without regard

to the database we will use, Hibernate will be good with it. Specifically, as we will most likely be using a kind of SQL server, Hibernate is perfect.

Configuring Hibernate to work with Spring:
This configuration is done via just xml config file in the project directories. It is easy, and does not need further dependencies.

Configuring Hibernate to work with the data server:
This configuration is done via just xml config file in the project directories. It is easy, and does not need further dependencies. We also need to write a sql script file to define the tables in our database.

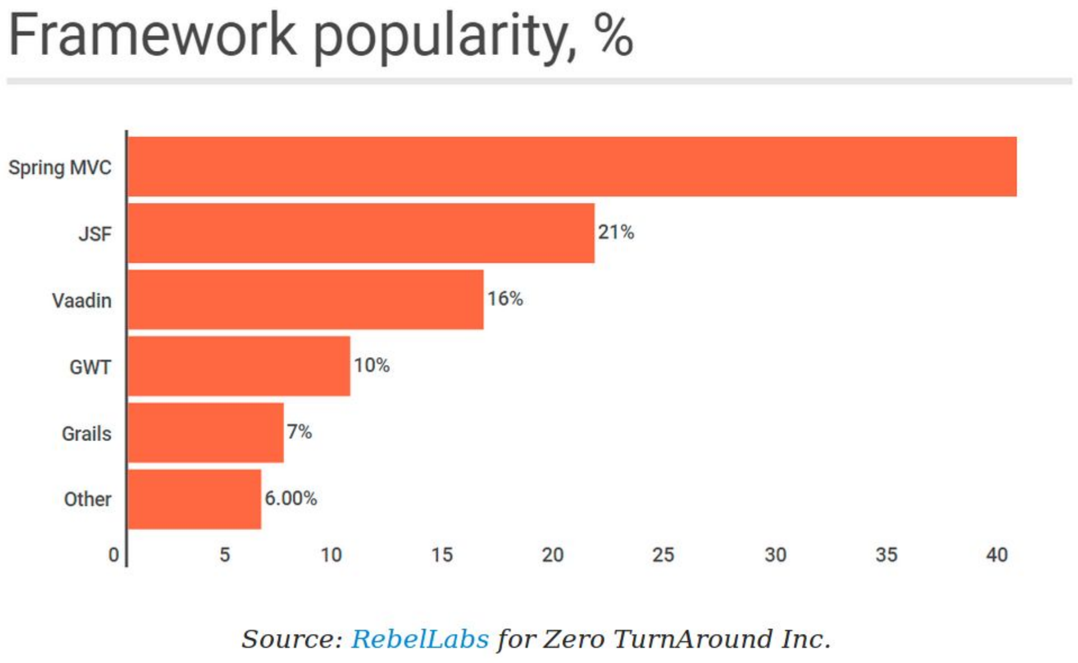
To summarize:
There really is no fault with using the Hibernate service. Both it, and the JDBC are highly used for accessing different types of data servers, and the only thing we need to do is to configure the correct one with the system, and implement the objects about which we want to converse with the server.

References:

- https://blog.angular-university.io/developing-a-modern-java-8-web-app-with-spring-mvc-and-angularjs/
- https://examples.javacodegeeks.com/enterprise-java/spring/mvc/angularjs-spring-integration-tutorial/
- https://www.journaldev.com/2882/hibernate-tutorial-for-beginners
- https://www.journaldev.com/3524/spring-hibernate-integration-example-tutorial
- https://www.journaldev.com/3531/spring-mvc-hibernate-mysql-integration-crud-example-tutorial

Notes:

- C# will be covered if and when Daniel covers technologies like ASP.
- Other technologies like Spring is JSF. It isn't as recommended as Spring (this I read immediately when looking for other technologies), but will be covered if Tanya or other team members require it.


Framework popularity, %
Spring MVC
JSF
Vaadin
GWT
Grails
Other
Source: RebelLabs for Zero TurnAround Inc.