Report

1.1 How My Design fits with the 3-tier architectural model and the model-view-controller software pattern

The implemented systems fits within this described model and this structured using packages. These packages are:

- webapps2019.jsf (Tier 1/Presentation tier)
- webapps2019.ejb (Tier 2/Business Logic tier)
- webapps2019.entity (Tier 3/Data Access Logic tier)

The Presentation Tier (Tier 1)

The presentation tier is made up of two sets of components: Java Server Faces (JSFs) and a number of backing beans that are linked to these JSFs. Each of the backing beans reacts to any user interaction by calling the methods of the business logic tier which will then in turn provide the JSFs with up to date data to display. Any form of data entry is first dealt with by the JSFs before it is passed in the business methods as a parameter. The navigations between these pages are explicitly defined in a faces-config.xml file

When considering the MVC software pattern these components could also be considered to behave somewhat like a controller and view. The JSFs 'render' the information from the model and displays it to the user. Moreover, it passes on any user gestures/interactions to the controller in the form of the backing beans associated with these JSFs. These could be considered controller like as they define application behaviour with calls to business methods in the EJBs and maps user interaction to updates in the model.

The Business Logic Tier (Tier 2)

The business logic tier is made up of a number of Enterprise Java Beans that interact with the data access tier by creating named queries. Any persistent data that is received from the data access tier is wrapped in simple classes that are then passed onto the presentation tier to be presented to the user. The tier performs the registration logic, login logic, and gets data regarding funds raised as well as querying the different tables in the database through the third tier.

The Data Access Tier (Tier 3)

The data access tier is made up of a number of JPA entities each with a number of defined queries that can be used by the EJBs to access the stored data. These entities populate a database that is running on a Glassfish server.

Both business logic and data access tiers could be considered to make up the model in this instance of the MVC design patterns. They expose the application functionality and encapsulate the application's state and keep the view constantly updated of this.

1.2 Security

How is the system secured?

- Secure digest function (SHA-256)
- JDBCRealm-based authentication
- Declarative and programmatic J2EE security
- Web-resource collections
- Authorisation constraints
- Secure connections

Advantages

- With use of a digest function the user-password is never sent to the server in plaintext
- JDBCRealm-based authentication using forms allows the dynamic registration of new users and the storing of their credentials to a relational database system unlike using a file-realm which does not provide the latter
- Web-resource collections utilise URL-patterns to place restrictions on the available resources to different actors. This can protect confidential information and ensures that actors only interact with the system within a defined scope

Disadvantages

- Declarative and programmatic J2EE security require web.xml and application server configuration
- Web-resource collections and authorisation constraints become impractical as the number of roles etc... grows
- Digest functions can be vulnerable to man-in-the-middle attacks. Other alternatives such as SSL are less vulnerable to this

1.3 Extend design as to not be a single point of failure

To ensure the system does not act as a single point of failure replications and redundancy could be introduced into the design.

Replication

Given the sole reliance on a single server for all data storage and access its failing would be catastrophic for the system in its current state. If the system were to be extended to introduce some replication of this data across two severs, if one of the servers were to malfunction and no longer be operational, traffic could be routed to the second server and once the original server is back up and running the data could be replicated from the second server to ensure that there are no inconsistencies between the data. With two servers handling traffic at the same time this may also increase the load that the system can handle.

Redundancy

In a similar vein to replication, redundancy could be introduced into the system to ensure there is no or minimal downtime if there is a failure within the system. This redundancy could manifest itself in the form of a backup server, backing up at regularly intervals so that in the face of an issue with the database, a known good environment can be restored with no to minimal data loss.

1.4 Dealing with concurrent users accessing functionality and data

Concurrency is dealt with within the system by J2EE transactions in the form of entity managers. The transactions within the system are managed by a container-based entity manager. This ensures that the sharing of resources is handled in an automatic and effective way. Each interaction with the business logic tier of the system spawns a new transaction. A transaction, if change to date in the data access tier, is not committed until the transaction has gone through its lifecycle i.e. the resource is no longer in use and the data change is valid and safe to persist to the database. While a resource is in use it briefly becomes unavailable, must like resource management in an OS. However unlike in an OS this isn't always the case. In the case of this system, the EJB containers and persistence context would ensure a number of things. If any data is access, it will only be persisted back into the database if what was returned from querying the database is the same state as the database is just before the commit. If this is not the case, then the persist is rolled back and the database will be queried again. This always for functions and data to be accessed currently without causing inconsistent retrievals or the loss of persisted data.