“You Might Also Like”

Architecture Notebook

There is guidance within this template that appears in a style named InfoBlue. This style has a hidden font attribute that allows you to toggle whether it is visible or hidden in this template. Use the Microsoft® Word® menu **Tools > Options > View > Hidden Text** check box to toggle this setting. There is also an option for printing: **Tools > Options > Print**.

# Purpose

This document describes the philosophy, decisions, constraints, justifications, significant elements, and any other overarching aspects of the system that shape the design and implementation.

# Architectural goals and philosophy

* Data about games of different systems will need to be consistent, even though the formats retrieved will be different
* Access is not highly critical, so it doesn’t matter if the site is down from time to time
* Access isn’t required locally, only through the web
* The site is implemented as a client-server system. The server is hosted through Heroku.

# Assumptions and dependencies

* PostgreSQL will be used in database management, because Heroku provides a free PostgreSQL database, so less effort is needed when deploying live.
* PrimeFaces will be used in addition to vanilla JSF for the web components, because it has a wide variety of UI components, and it’s easier to use components from a single source.

# Architecturally significant requirements

* Security: users and admins need to be able to log in and only access the functions they are authorised
* Persistence: data must stay consistent

# Decisions, constraints, and justifications

* Application will be deployed onto a Tomcat server, as it is demonstrated that this plays nice with both Eclipse and on Heroku.
* PostgreSQL will be used, because this is demonstrated to work on Eclipse and Heroku (where a free PostgreSQL database is provided).
* Security isn’t too important, as users are not sharing any sensitive data
* Data persistence and integrity is important, so that the web app functions correctly. Data is paramount.
* The app that fetches new data must only be able to be run once at a time – if two instances are going, the database could become inconsistent with more than one process accessing it at once.

# Architectural Mechanisms

## Architectural Mechanism 1

Database access will be restricted to DAO classes that deal with only one table, or possibly one function of a table if a table covers many purposes. This is so common functions can be re-used across the application.

# Key abstractions

Please see Requirements document.

# Layers or architectural framework

As this is a JSF application, it will follow the Model-View-Controller framework. The View layer is all the xhtml files which incorporate what the users see. The controller is hidden by the JSF architecture – the FacesContext is involved in transporting data between the view and the Java classes behind it on the server. The Model layer is the Managed Beans which are involved in manipulating the data that is shown to the user or received from the user, along with all the backing Java classes that do more intensive work on it.

# Architectural views

## Recommended views

* **Logical:** Describes the structure and behavior of architecturally significant portions of the system. This might include the package structure, critical interfaces, important classes and subsystems, and the relationships between these elements. It also includes physical and logical views of persistent data, if persistence will be built into the system. This is a documented subset of the design.
* **Operational:** Describes the physical nodes of the system and the processes, threads, and components that run on those physical nodes. This view isn’t necessary if the system runs in a single process and thread.
* **Use case:** A list or diagram of the use cases that contain architecturally significant requirements.

**Use case view**

Please see Requirements document

**Logical view**

The system comprises three main packages: Game Data, User Interface, Business Services, and Business Objects.

Game Data contains self-contained apps that scrape game company servers for new data, and store it. User Interface contains all the presentation pages and the managed beans that support them. Business Services contains the application logic to support the app. Business Objects contains the entity objects used in the app.

**Operational view**

There are two separate apps in the application. One is for updating game data, and is launched by visiting a page and inputting a master password (that only the team knows). The other is the main site, where users can use the functions. Both are hosted on Heroku, deployed as war files.