Anwenden der Low Level Design Patterns

Gruppe H

17. Dezember 2017

Inhaltsverzeichnis

Teil I.

Welches Low Level Design Pattern eignet sich für unseren Faithinator?

1. Client-Dispatcher-Server (CDS)

Da dieses Pattern hauptsächlich von den zentralen Rolle des Dispatchers abhängt, der allerdings das Interface für die Clients darstellt, wären künftige Anpassungen am Interface äußerst komplex und risikoreich. Die großen Vorteile dieses Patterns (Server-Redundanz, geringer Migrationsaufwand, Fehlertoleranz) sind für unseren Faithinator nicht die wichtigsten Eigenschaften. Hier wären eher Erweiterbarkeit und Wartbarkeit zu nennen.

2. Command Processor

Das Command-Processor Pattern eignet sich (fast) ausschließlich zur Protokollierung von Ereignissen (d.h. bspw. Erstellung von Logs oder Ermöglichen des Undo-Befehls). Daher eignet es sich denkbar schlecht für unseren Faithinator.

3. Forwarder Receiver

Das Design Pattern "Forwarder Receiverïst ebenfalls nicht geeignet für unsere Applikation. Es bietet eine Peer-to-Peer Kommunikation, sodass verschiedene Knoten in einem Netzwerk kommunizieren können. In unserer App sollen gibt es jedoch eine zentralen Server, auf dem alle Anfragen bearbeitet und alle Daten gespeichert werden.

4. Master-Slave

Das Master-Slave Pattern wird genutzt um Arbeit zu partitionieren, dies wird zum Beispiel genutzt um Probleme mit langer Laufzeit zu lösen. Im Fall unseres Faithinators ist dies nicht der Fall, somit ist es nicht sinnvoll das Master-Slave Pattern anzuwenden.

5. Proxy

Das Proxy Pattern lässt sich gut für unseren Faithinator nutzen. Da es eine Zwischeninstanz zwischen Clients und Original bildet um direkte Kommunikation zu vermeiden. In unserem Fall könnte das Proxy Pattern einen Übergang zwischen Clients/Admins und der Datenbank bilden. Damit kann sichergestellt werden ,dass die Clients und Admins nicht unbegrenzten Zugriff auf die Datenbank haben.

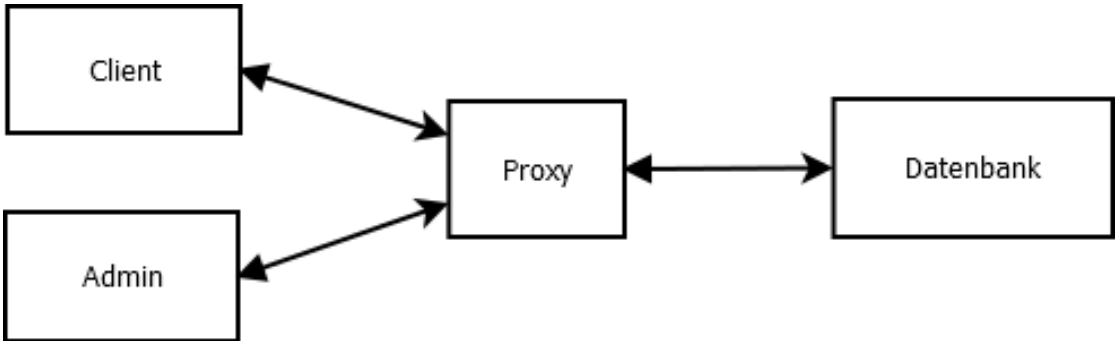

Abbildung 5.1.: Übersicht: Zusammenarbeit der Komponenten beim Proxy-Pattern

6. Publisher Subscriber

Das Publisher-Subscriber-Pattern würde sich dazu eignen, das Versenden der Fragen an die Clients zu verwalten. Clients können sich dafür entscheiden, an einer angebotenen Fragestellung teilzunehmen, in diesem Fall können sie beim Publisher (Server) als Subscriber eingetragen werden. Der Publisher kann dann Benachrichtungen zum Start oder Ende der Fragezeit und den Chatrunden verteilen. Wenn Einladungen zu Fragen zielgruppenspezifisch versendet werden sollen, ließe sich das ebenfalls über dieses Pattern realisieren, ebenso wie ein Benachrichtigungssystem für die Admins einer Frage.

7. View Handler

Das Design Pattern "View Handlerïst für unsere Applikation nicht notwendig oder von Vorteil. Es gibt für jede Aktion eine eigene grafische Benutzeroberfläche, Daten müssen nicht in verschiedenen Fenster in verschiedenen Formaten dargestellt werden. Das Potenzial des View Handler Patterns würde nicht genutzt werden.