Anwenden der Design Patterns auf unseren Faithinator

Gruppe H

4. Januar 2018

Unser Faithinator hat, grob abstrahiert, folgende Bestandteile:

- Datenbank: Die Datenbank enthält sämtliche Daten über die User, die Clients und die Fragen.
- Code bzw. Logik: Der Code implementiert alle Funktionalitäten der App.
- Grafische Benutzeroberfläche: Sowohl für die App als auch für die Weboberfläche gibt es eine grafische Benutzeroberfläche, die die Funktionalitäten repräsentiert.
- Chatverbindungen: Die Kommunikation zwischen den Usern findet über Chats statt.
- Fragen: Die Fragen müssen von den Admins erstellt und ausgewertet und von den Clients beantwortet werden.

Um diese organisieren und umsetzen zu können, haben wir die folgenden Pattern ausgewählt.

1 Architectual Pattern

1.1 MVC

Das “Model - View - Contoller” - Pattern können wir für die Grundstruktur des Faithinators benutzen. Die **Models** stellen die direkte Schnittstelle zur Datenbank dar und werden ausschließlich über die Controller angesteuert. Die **Views** zeigen den Benutzern die Daten in angemessener Weise auf der Benutzeroberfläche an. Es gibt Views für die Fragen, für die Chats und für alle anderen Bestandteile der Applikation. Die **Controller** reagieren auf die Benutzereingaben und leiten gegebenenfalls Änderungen an den Daten an die Models weiter. Die Models führen diese Änderungen dann direkt in der Datenbank aus.

Beispiele für die Anwendung im Faithinator:

- Anzeige der Fragen für die Clients und Möglichkeit der Beantwortung
  - View: Anzeige in geeigneter Form
  - Controller: Weitergabe der Antwort an das Model
  - Model: Eintragen der Antwort in die Datenbank
- Benutzeroberfläche zur Erstellung von Fragen
  - View: Anzeige des Formulars zur Definition der Frage durch einen Admin
  - Controller: Validierung und Weitergabe der eingegebenen Daten an das Model
  - Model: Eintragen der neuen Frage in die Datenbank

- Login für die Benutzer
  - View: Anzeige des Logins mit „Benutzername“, „Passwort“, etc.
  - Controller: Validierung und Weitergabe der eingegebenen Daten an das Model
  - Model: Abgleich der eingegebenen Daten mit den Daten in der Datenbank

2 Design Pattern

2.1 Proxy

Der Proxy regelt den Zugriff von verschiedenen Benutzergruppen auf die Daten in der Datenbank.
Konkret bedeutet das, dass Clients durch ihre Zugriffseinschränkungen nur für sie vorgesehene Fragen sehen. Admins können beispielsweise alle Fragen anschauen und diese auch bearbeiten.

Vorteile:

- Sicherheit (Unveränderbarkeit bestimmter wichtiger Daten durhc Clients, z.B. der Fragen)
- Effizienz (Anzeige der gleichen Daten für verschiedene Clients mit gleicher Berechtigungsstufe)
- Rechte (siehe oben)

2.2 Forwarder Receiver

Für den Chat haben wir kein Pattern gefunden, welches alle nötigen Funktionen des Chats optimal unterstüzt. Die Chatfunktion kann aber aus unserer Sicht über das Forwarder Reveiver Pattern realisiert werden. Chats bedeuten immer eine 1:1 Verbindung zwischen Clients, in denen sie Nachrichten und/oder Dateien austauschen können. Denkbar wäre, dass die IPC mithilfe des Servers realisiert wird, der Chats automatisch in der Datenbank protokolliert.