BAI5-VSP	Praktikum Verteilte Systeme – Aufgabe 2	AZI/BEH/KSS
WiSe 15	Step 2: Clients Calling	

Vorbereitung

Rufen Sie sich die verschiedenen HTTP-Methoden und Arten der Parameter-übergabe ins Gedächtnis, oder noch besser, suchen Sie sich eine schöne Übersichtsseite und schreiben sich die raus, die Sie brauchen und wozu gleich mit!

Informieren Sie sich außerdem über die gebräuchlichsten HTTP-Status-Codes und deren Bedeutung.

Einführung – Client-Service-Communication

Im ersten Aufgabenblatt wurden nur einfache Anfragen an die Services gestellt, welche leicht über den Browser überprüfbar waren.

In diesem Aufgabenblatt sollen Sie die zurückgegebenen Werte weiter verwenden und mittels eines Clients verarbeiten und damit neue Anfragen an andere Services stellen.

Außerdem werden nun weitere HTTP-Methoden benötigt und verschiedene Arten, Daten an Services zu übergeben.

Materialien

Die für dieses Aufgabenblatt benötigten Interfaces finden Sie unter:

https://pub.informatik.haw-hamburg.de/home/pub/prof/kossakowski_klaus-peter/wise2015/verteiltesysteme/step2.raml

Eine anschauliche Darstellung ist zu finden unter:

https://pub.informatik.haw-hamburg.de/home/pub/prof/kossakowski_klaus-peter/wise2015/verteiltesysteme/step2.html

Um Interoperabilität zu sichern, müssen die Services diese Interfaces anbieten.

Die Api für den in Aufgabe 2.2 angesprochen Dienst ist zu finden unter

https://pub.informatik.haw-hamburg.de/home/pub/prof/kossakowski_klaus-peter/wise2015/verteiltesysteme/services.raml bzw die HTML unter

https://pub.informatik.haw-hamburg.de/home/pub/prof/kossakowski_klaus-peter/wise2015/verteiltesysteme/services.html

Bei Risiken oder Nebenwirkungen, wird der Apotheker nicht helfen. Deshalb bei Vorschlägen zum Interface direkt an Andrej.Zieger@haw...

Aufgabe 2.1: Spieler registrieren

Implementieren Sie einen Client so, dass dieser über den /games-Service die folgenden Funktionen anstoßen kann:

- Benutzer können mit dem Client ein neues Spiel eröffnen
  `post /games`
- Benutzer können sich mit dem Client als Spieler registrieren
  `put /games/{gameid}/players/{playerid}`
- Benutzer können mit dem Client melden, dass sie fertig sind und das Spiel losgehen kann

Wenn alle Clients bereit sind, kann das Spiel beginnen – die erste Person muss anfangen zu würfeln! Achten Sie darauf, dass für die verschiedenen Spielkomponenten auch unterschiedliche Hosts über- bzw. angegeben werden können.

Aufgabe 2.2: Verzeichnis Dienst / Yellow Pages

Alle Services müssen bei einem Verzeichnis-Dienst angemeldet werden. Dieser Dienst muss nicht von Ihnen implementiert werden, sondern nur genutzt werden und wird unter http://vs-docker.informatik.haw-hamburg.de:8053/service bereitgestellt.

Sie können diesen Dienst nutzen, um weitere Spielkomponenten des Systems zu finden. Hierbei ist zu beachten, dass die Root-Resourcen der Services genutzt werden als „service“ z.B. „/dice“, „/games“, „/boards“.

Aufgabe 2.2 A: Keep on rolling!

Implementieren Sie den /boards-Service, so dass

- der Client einen Würfelwurf übergeben kann mit
  `post /boards/{gameid}/players/{playerid}/roll`
- die Position des Spielers vom Service verändert wird
- die Position des Spielers abgefragt werden kann mit
  `get /games/{gameid}/players/{playerid}`
- der Zustand des Brettes abgefragt werden kann mit
  `get /games/{gameid}`

Synchronisation: Achten Sie darauf, dass nicht zwei Spieler gleichzeitig Würfeln können! Nutzen Sie hierzu einen verteilten Mutex. Ein Spieler darf nur mit dem /boards-Service agieren, wenn gilt

- Der Mutex kann erworben werden mittels
  `put /games/{gameid}/turn`

- Es kann abgefragt werden, ob ein Spieler den Mutext hält mit
  `get /games/{gameid}/turn`
- Freigegeben wird der Mutex durch
  `delete /games/{gameid}/turn`
- Der Mutex sollte frei gegeben werden direkt bevor der Zug beendet wird durch
  `put /games/{gameid}/players/{playerid}/ready`

Aufgabe 2.2 B: Die /Bank

Implementieren Sie den `/banks`-Service, so dass

- ein Konto erstellt werden kann mit
  `post /banks/{gameid}/players`
- der Kontostand abgefragt werden kann mit
  `get /banks/{gameid}/players/{playerid}`
- Geld von der Bank überwiesen werden kann mit
  `post /banks/{gameid}/transfer/to/{to}/{amount}`
- Geld eingezogen werden kann mit
  `post /banks/{gameid}/transfer/from/{from}/{amount}`
- Geld von einem zu anderen Konto übertragen werden kann mit
  `post /banks/{gameid}/transfer/from/{from}/to/{to}/{amount}`

(Lokale) Transaktion: Achten Sie darauf, dass ein Geldtransfer immer als eine Transaktion zu verstehen ist: entweder wird diese ganz – oder gar nicht – ausgeführt.

Frage:
Wie könnte man eine at-most-once Fehlersemantik implementieren?

Optional 2.4: Bilder braucht das Land

Implementieren Sie eine Repräsentation des Spielstandes bzw. Spielbretts für den Client. „It's nice to have a GUI“ ;)

Optional 2.5: Waiting Lounge

Implementieren Sie eine Anzeige, die darstellt, welche Spieler das Spiel bisher betreten haben, während gewartet wird. Die Lounge-Musik ist keine Pflicht hierbei.

Optional 2.5: List Games

Implementieren Sie, dass der Client den `/games`-Service nach aktuellen Spielen abfragen kann mit: `get /games`