

Stories

Verteilte Systeme und Komponenten

Gruppe 5 (Bucher, Kiser, Meyer, Sägesser)

19.03.2018

1 Stories (nach Epics)

1.1 Epic Log-Ereignisse

1. Aufzeichnung Spielverlaufs: Als *Anwender* möchte ich *persistent festgehaltene Spielereignisse*, um *den Spielverlauf bei Bedarf nachvollziehen zu können*.
2. Aufzeichnung von Problemen: Als *Anwender* möchte ich *persistent festgehaltene Fehlermeldungen*, um *auf tretende Probleme nachweisen zu können*.
3. Aufzeichnung von Exceptions: Als *Entwickler* möchte ich *persistent festgehaltene StackTraces*, um *geworfene Exceptions nachvollziehen zu können*.

1.2 Epic Log-Level

4. Message-Level: Als *Entwickler* möchte ich *zu jeder Log-Meldung ein Message-Level setzen können*, um *Log-Meldungen auf verschiedenen Levels definieren zu können*.
5. Codiertes Filter-Level: Als *Entwickler* möchte ich *zur Laufzeit einen Filter-Level setzen können*, um *die Ausgabe der Log-Meldungen steuern zu können*.
6. Konfiguriertes Filter-Level: Als *Administrator* möchte ich *vor der Ausführung einen Filter-Level konfigurieren können*, um *die Ausgabe der Log-Meldungen steuern zu können*.

1.3 Epic Log-Interface

7. Logger-Interface: Als *Entwickler* möchte ich *ein definiertes Logger-Interface zur Verfügung haben*, um *eine Logger-Komponente umsetzen zu können*.
8. Logger: Als *Entwickler* möchte ich *einen Logger zur Verfügung haben*, um *Log-Meldungen aus dem Game ausgeben zu können*.
9. Logger-Setup: Als *Entwickler* möchte ich *ein LoggerSetup zur Verfügung haben*, um *Zugriff auf eine Logger-Instanz zu bekommen*.

1.4 Epic Architektur

10. Komponenten-Austausch: Als *Administrator* möchte ich eine austauschbare und plattformunabhängige *Logger-Komponente*, um die *Logger-Komponente* zur Laufzeit ohne Code-Anpassungen und Neukompilation austauschen zu können.
11. StringPersistor: Als *Entwickler* möchte ich eine *StringPersistor-Implementierung* für *Textdateien* (*StringPersistorFile*) zur Verfügung haben, um *geloggte Meldungen* persistent in *Textdateien* festhalten zu können.
12. Mehrere Log-Clients: Als *Administrator* möchte ich einen *Logger-Server*, auf den mehrere *Logger-Komponenten* parallel loggen können, um einen *Server* für mehrere *Clients* zur Verfügung stellen zu können.
13. Payload-Adapter: Als *Entwickler* möchte ich einen *Adapter* für die Übergabe des *Payloads*, um *Daten* in *strukturierter Form* dem *StringPersistor* übergeben zu können.
14. Adapter-Tests: Als *Entwickler* möchte ich *Unittests* für den *Adapter* haben, um zu sehen, ob *Änderungen am Adapter* zu *Problemen* führen.
15. Speicherformat-Strategien: Als *Entwickler* möchte ich über leicht austauschbare *Log-Strategien* verfügen, um das *Speicherformat* für *Textdateien* einfach auf *Code-Ebene* wählen zu können.

1.5 Epic Persistenz

16. Log-Aufzeichnung: Als *Administrator* möchte ich eine verlässliche *Aufzeichnung der Log-Ereignisse* auf dem *Server*, um die *Ereignisse* aus dem *Spiel* in korrekter Reihenfolge nachverfolgen zu können.
17. Unterscheidung Log-Clients: Als *Administrator* möchte ich nach *Client-Instanz* unterscheidbare *Logdateien*, um die *Log-Meldungen* verschiedener *Clients* auseinanderhalten zu können.
18. Textdatei-Log: Als *Administrator* möchte ich *Log-Ereignisse* in einfachen *Textdateien* festhalten, um diese mit gängigen Werkzeugen betrachten und auswerten zu können (*grep*, *tail*, etc.).

1.6 Epic Format

19. Speicherung Log-Quelle: Als *Administrator* möchte ich die *Quelle einer Logmeldung* sehen, um verschiedene *Clients* und *Sessions* voneinander unterscheiden zu können.
20. Speicherung Log-Level: Als *Administrator* möchte ich den *Level einer Logmeldung* sehen, um diese je nach Bedarf nachträglich filtern zu können.
21. Speicherung Ereignis-Zeitstempel: Als *Administrator* möchte ich den *Zeitstempel der Message-Erstellung* sehen, um den *genauen Zeitpunkt des Ereignisses* zu kennen.
22. Speicherung Log-Zeitstempel: Als *Administrator* möchte ich den *Zeitstempel des Message-Eingangs* auf dem *Server* sehen, um *Verzögerungen beim Logging-Vorgang* erkennen zu können.
23. Speicherung Message-Text: Als *Administrator* möchte ich den *Text der Log-Message* auf dem *Server* sehen, um das aufgetretene Ereignis erkennen zu können.

1.7 Epic Konnektivität

24. Konfiguration Serverzugriff: Als *Administrator* möchte ich *die Erreichbarkeit des Servers mittels Konfigurationsdatei definieren können*, um *die Verbindung zum Log-Server ohne Quellcodeanpassung definieren zu können*.
25. Robuste Netzwerkanwendung: Als *Anwender* möchte ich *eine Anwendung mit robuster Netzwerkkonnektivität*, um *auch bei Netzwerkunterbrüchen einen ununterbrochenen Spielfluss ohne verlorene Log-Meldungen zu haben*.
26. Message-Queue: Als *Entwickler* möchte ich *auf tretende Log-Ereignisse in einer Message-Queue zwischenspeichern können*, um *Log-Ereignisse bei einem Verbindungsunterbruch erneut senden zu können*.

1.8 Epic Viewer

27. Viewer: Als *Anwender* möchte ich *einen Viewer für Logmeldungen haben*, um *Logmeldungen ohne Kenntnis des physischen Speicherorts auf dem Server betrachten zu können*.

2 Rollen

- Ein *Entwickler* hat Zugriff auf den Quellcode und verfügt über die Fähigkeiten und Berechtigungen um Änderungen daran vorzunehmen.
- Ein *Administrator* hat Zugang auf beide konzeptionell involvierten Systeme (Client und Server) und Zugriff auf die Konfigurationsdateien aller involvierter Komponenten.
- Ein *Anwender* kann das Spiel auf dem Client ausführen, kann aber keine Änderungen an Quellcode und Konfigurationsdateien vornehmen und nicht auf den Server zugreifen.