Architecture Notebook: Turnierauswertungssoftware

Inhaltsverzeichnis

1. Links zur Erläuterung

HTW Seite

OpenUP Doku

2. Zweck

Dieses Dokument beschreibt die Philosophie, Entscheidungen, Nebenbedingungen, Begründungen, wesentliche Elemente und andere übergreifende Aspekte des Systems, die Einfluss auf Entwurf und Implementierung haben.

3. Architekturziele und Philosophie

- Da die Anzahl der Nutzer nur maximal 200 beträgt, wird keine besondere Architektur gebraucht. Es reicht ein einfacher Webserver.
- Die Webseite soll auch offline nutzbar sein und beim späteren synchronisieren eine Möglichkeit bieten, konfligierende Datensätze zu beheben.

4. Annahmen und Abhängigkeiten

5. Architektur-relevante Anforderungen

6. Entscheidungen, Nebenbedingungen und Begründungen

Wir haben uns für eine Client-Server-Webanwendung aus folgenden Gründen entschieden: - Da derselbe Datenbestand von mehreren unabhängigen Benutzern geändert wird, wurde die Datensynchronisation bei einer Peer-To-Peer-Architektur einen zu großen Aufwand erfordern. - Die Aufwand für die Konfiguration von den Clientgeräten ist gering, wenn zur Nutzung des Softwaresystems auf der Benuzerseite nur ein Webbrowser benötigt wird.

6.1. Auswahl der zuverwendenden Komponenten

1. Das Software-Produkt wird später mithilfe von Docker ausgeliefert, damit sichergestellt ist, dass jedes System die gleichen Voraussetzungen erfüllt.
2. Zur Speicherung der Daten wird PostgreSQL benutzt, da die Speicherung in Datenbanken übersichtlicher und robuster ist, als die Speicherung der Daten in Dateien.
3. Für eine einfachere Gestaltung der Website wird des weiteren TailwindCSS benutzt.
4. Für das Back-End wird Java benutzt. Dies vereinfacht die Einarbeitung, da bereits jeder Informatik-Studiengang Java behandelt hat.

7. Architekturmechanismen

Doku "Concept: Architectural Mechanism"

1. Mehrbenutzer-Zugirff.
   Gleichzeitige Bedienung der Anwendung durch mehrere Nutzer.
2. Nutzer-Identifizierung.
   Eine Folge von Anfragen eines Nutzers, soll demjenigen Nutzer zugeordnet werden können.
3. Daten-Persistenz.
   Dienste zum Lesen und Schreiben von gespeicherten Daten.
4. Systemredundanz
   Bei Nichterreichbarkeit des Hauptsystems, soll die zweite System die volle Funktionalität übernehmen.

8. Wesentliche Abstraktionen

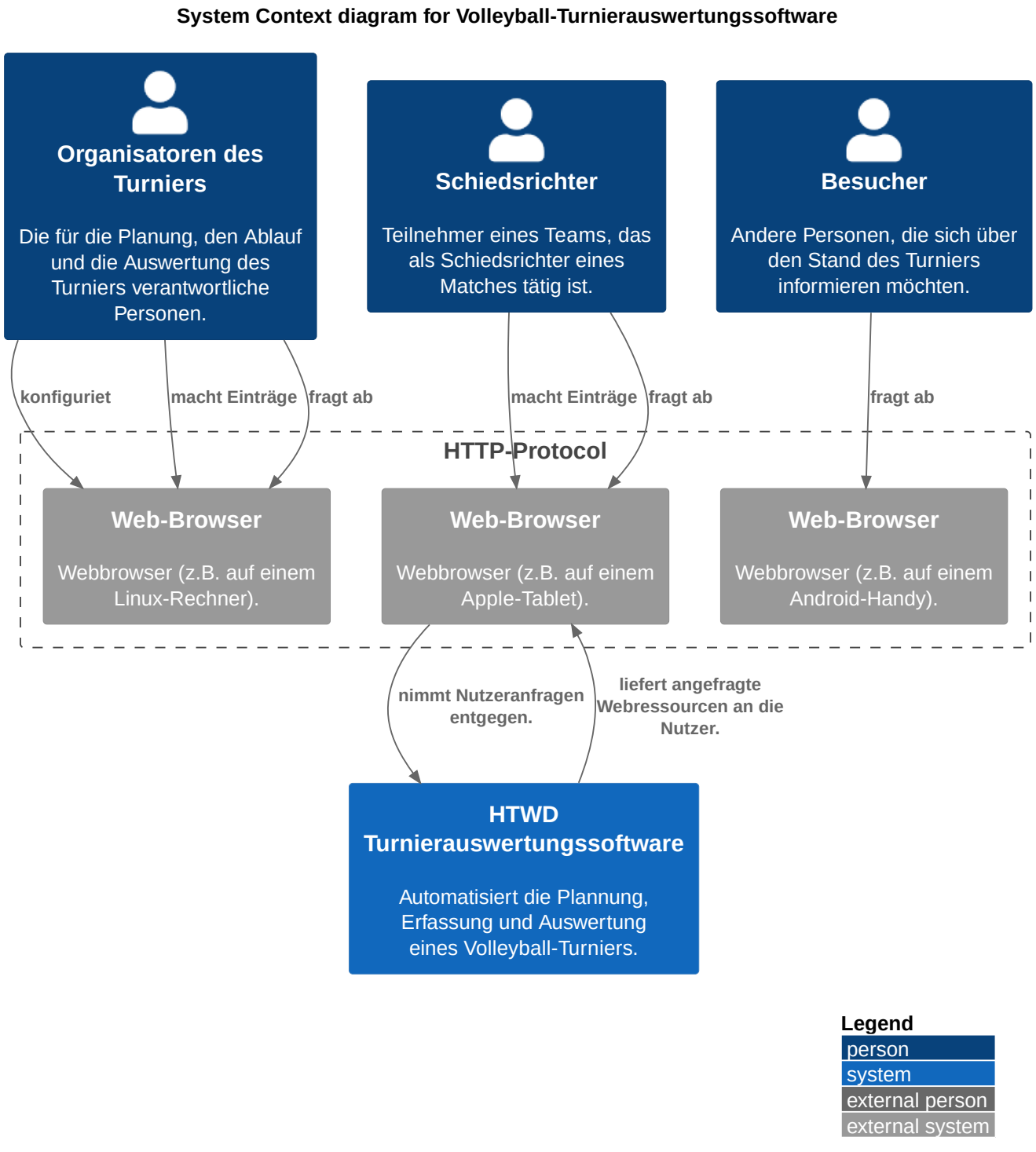
9. Schichten oder Architektur-Framework

10. Architektursichten (Views)

Folgende Sichten werden empfohlen:

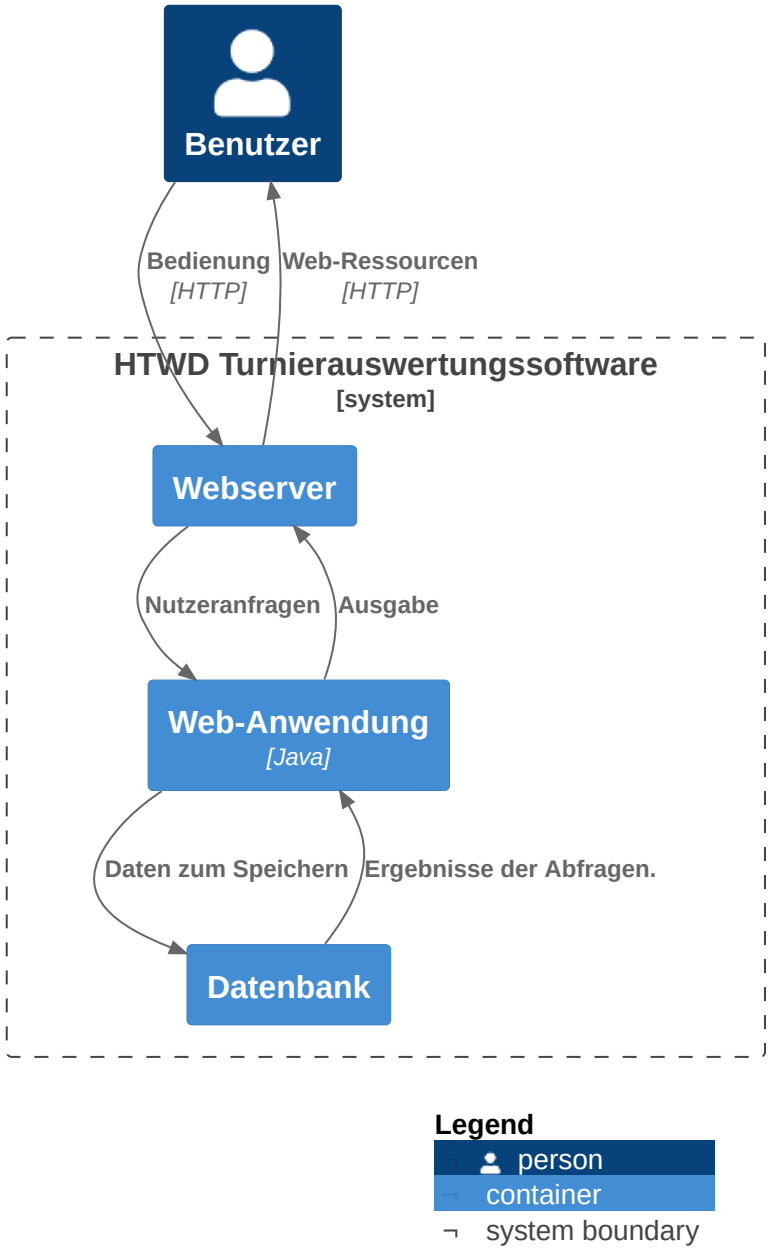
10.1. Logische Sicht

10.1.1. C4 Context



10.1.2. C4 Container

System Container diagram for Volleyball-Turnierauswertungssoftware



10.2. Physische Sicht (Betriebssicht)

10.3. Use cases