Ringstats Dokumentation

Justin Rost 7208260
Michelle Reichelt 7216628
Yasin Ergün 7215184

31. Mai 2026

Inhaltsverzeichnis

1 Anwendungsszenario

In diesem Kapitel zeigen wir, worum es bei Ringstats eigentlich geht und in welchen Situationen die Anwendung dem Spieler im Spiel weiterhilft.

1.1 Projektidee und Zielsetzung

Die Idee von Ringstats ist die Bereitstellung einer Schnittstelle zwischen dem aktiven Spiel und einer Datenbank mit hilfreichen Spielinformationen. Während des Spiels erfasst die Desktop Anwendung in Echtzeit Parameter des Spielers und schickt diese an die Datenbank. Zu diesen Parametern zählen z.B. die aktuelle Position oder Ausrüstung. Auf Grundlage dieser Live Daten stellt unsere Anwendung dann hilfreiche Informationen über Bosse oder Gegner in der Region des Spielers bereit. Zudem kann auch eine regionenbasierte Checkliste angezeigt werden, auf der sich weitere benutzerdefinierte Aufgaben befinden. All diese Informationen werden auf In-Game Panels eingeblendet.

1.2 Kernfunktionen und Anwendungsfälle

Um den Nutzen von Ringstats zu verstehen, haben wir folgende Kernfunktionen definiert: Die wichtigste Funktion ist das Erfassen der Spielerdaten in Echtzeit. Während der Spieler spielt, liest die Desktop Anwendung laufend aus, wo der Spieler gerade steht, wie viel Leben er hat, wie schnell er ist und was er ausgerüstet hat. Diese Daten werden direkt an die Datenbank geschickt, damit immer der aktuelle Stand vorliegt. Sobald der Spieler einen Gegner oder Boss anvisiert, blendet die Anwendung ein kleines Panel im Spiel ein. Darin sieht der Spieler die Resistenzen des Gegners und welche Fähigkeiten dieser am häufigsten benutzt. So weiß man direkt, worauf man sich einstellen muss. Zusätzlich gibt es eine Checkliste, die sich nach der Region richtet, in der sich der Spieler gerade befindet. Diese lässt sich mit einem Hotkey ein und ausblenden. Auf der Liste stehen zum Beispiel Items, Dungeons und Bosse. Der Spieler kann die Einträge filtern und selbst festlegen, ob sie automatisch abgehakt werden, sobald eine Bedingung erfüllt ist, oder ob er das lieber von Hand macht. Alle diese Daten landen nicht nur im Spiel, sondern sind gleichzeitig über die Internetseite abrufbar. Der Spieler meldet sich auf der Seite an und verbindet sich mit der Desktop Anwendung. Dann kann er seine Informationen auch im Browser einsehen.

Damit das Ganze einen Sinn ergibt, hier ein paar typische Situationen aus dem Spiel:
Der Spieler startet einen Bosskampf. In einem kleinen Fenster sieht er sofort, welche Resistenzen der Boss hat und welche Fähigkeiten am häufigsten kommen.
Während der Spieler durch ein Gebiet läuft, werden seine Daten an den Server geschickt. Dadurch sieht er direkt, welche Truhen und Sammelobjekte in seiner Nähe sind.
Wechselt der Spieler in ein neues Gebiet, zeigt die Checkliste an, welche Aufgaben in diesem Gebiet noch offen sind.

2 Dokumentation der Datenmodellierung

In diesem Kapitel schauen wir uns an, wie wir die Daten von Ringstats aufbauen und wie das Ganze in den verschiedenen Datenbanken aussieht.

2.1 Das konzeptionelle ER-Modell

Die komplexe Struktur der Spielwelt bilden wir mit einem Klassischen ER-Modell ab. Das Modell veranschaulicht, wie der Spieler (Player) über seine Ausrüstung (Gear) mit Waffen (Weapon), Rüstungen (Armor) oder Talismanen (Talisman) etc. verbunden ist. Fast alle dieser Gegenstände haben einen Typen (Type) und Effekte (Item_Effect). Auch ist der Spieler (Player) immer in einer Region (Region), in der Gegenstände (Item) und Gegner (Enemy) aufzufinden sind.

Das Modell dient in den folgenden Kapiteln als Grundlage für die Modellierung und den Vergleich.

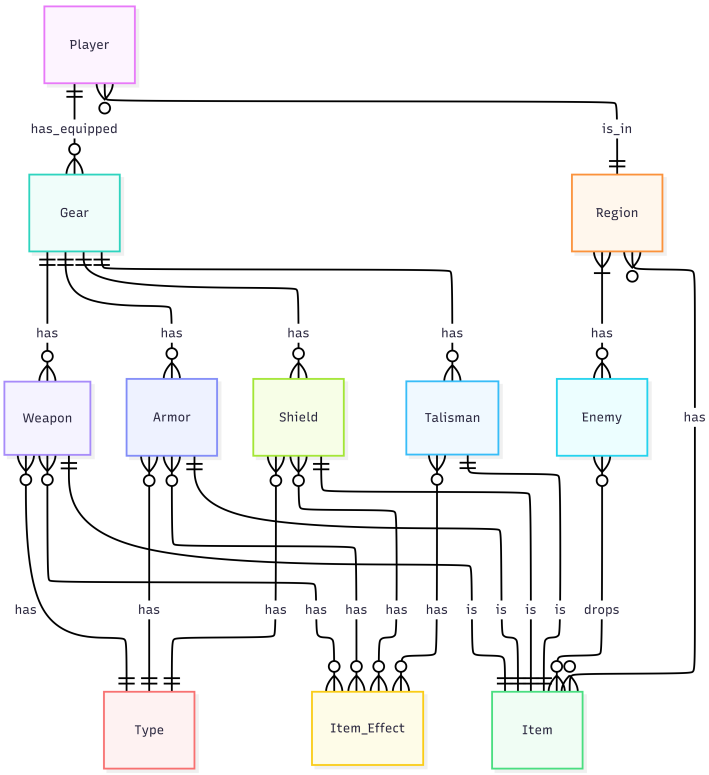

Abbildung 1: Konzeptionelles ER-Modell

2.2 Key-Value-Modellierung

Key-Value Datenbanken sind eine einfache und schnelle Lösung für die Speicherung von unstrukturierten Daten. Wir verwenden in diesem Kapitel Redis.

Schlüsselaufbau

Keys sind Strings, die eindeutig zum identifizieren von Objekten verwendet werden. Diese starten immer mit dem Namen der Entität, gefolgt von einem Doppelpunkt und der ID des Objekts. Dabei werden alle Entitäten als Redis-Datentyp Hash Map gespeichert um die Daten effizient zu verwalten.

Anhand dieses Systems bilden wir die Schlüssel wie folgt ab:

Datentyp	**Key-Struktur**	**Attribute**
HMAP	item_effect:{id}	id, description
HMAP	item_type:{id}	id, item_id, name
HMAP	weapon:{id}	id, item_effect_id, item_type_id, weight, damage
HMAP	armor:{id}	id, item_effect_id, item_type_id, weight, defense
HMAP	shield:{id}	id, item_effect_id, item_type_id, weight, damage, defense
HMAP	talisman:{id}	id, item_effect_id
HMAP	item:{id}	id, name, gear_type, gear_id, region_id
HMAP	player:{id}	id, name, health, stamina, level, region_id
STRING	player:{id}:weapon	item_id
STRING	player:{id}:shield	item_id
SET	player:{id}:armor	item_id
SET	player:{id}:talisman	item_id
SET	region:{id}:enemies	enemy_id
HMAP	region:{id}	id, name
SET	region:{id}:items	item_id
SET	region:{id}:enemies	enemy_id
HMAP	enemy:{id}	id, region_id, name, health, level, damage
SET	enemy:{id}:items	item_id

Tabelle 1: Übersicht der Schlüsselstrukturen in Redis

2.3 Dokumentenorientierte Modellierung

2.4 Graph-Datenbank-Modellierung

3 Vergleich der Datenmodellierung

3.1 Vor- und Nachteile des Key-Value-Modells

3.2 Vor- und Nachteile des dokumentenorientierten Modells

3.3 Vor- und Nachteile des Graph-Modells

4 Fazit