Ringstats Proposal

von
Yasin Ergün, Michelle Reichelt, Justin Rost

Zuletzt Bearbeitet am 13.05.2026

Einleitung

Unser Anwendungsbeispiel ist eine vollständige Elden Ring Datenbank mit Informationen zu Sämtlichen Daten wie Waffen, Gebiete, Monster, Bosse etc.. Es gibt eine Internetseite, auf der sich Spieler anmelden und sich im angemeldeten Zustand live mit der Desktop App verbinden können. Im verbundenen Zustand werden Informationen über den Spieler und seine Daten wie Weltkoordinaten, Leben, Geschwindigkeit und Ausrüstung in die Datenbank gestreamt und somit gleichzeitig über die Internet Seite abrufbar gemacht.

Dem Spieler wird mit einer Desktop Anwendung, die mit der Website verbunden wird, live auf dem Bildschirm angezeigt, welche Gegner, Bosse und Minibosse in seinem jetzigen Gebiet aufzufinden sind. Zudem soll auch eine Live Checkliste optional angezeigt werden, die mit einem Hotkey an und ausgeschaltet werden kann. Auf dieser können Händler, Items, Dungeons und Bosse gespeichert werden. Der Spieler kann Checklisten Elemente filtern und für diese aussuchen, ob sie automatisch mit einer Bedingung oder manuell abgehakt werden. Außerdem werden für Gegner und Bosse, die anvisiert sind, Informationen in einem Panel im Spiel angezeigt.

Anwendungsfälle

Ein Spieler startet einen Bosskampf und kann in einem kleinen Fenster sehen, was für Resistenzen dieser Boss hat und welche Fähigkeiten er am häufigsten benutzt.

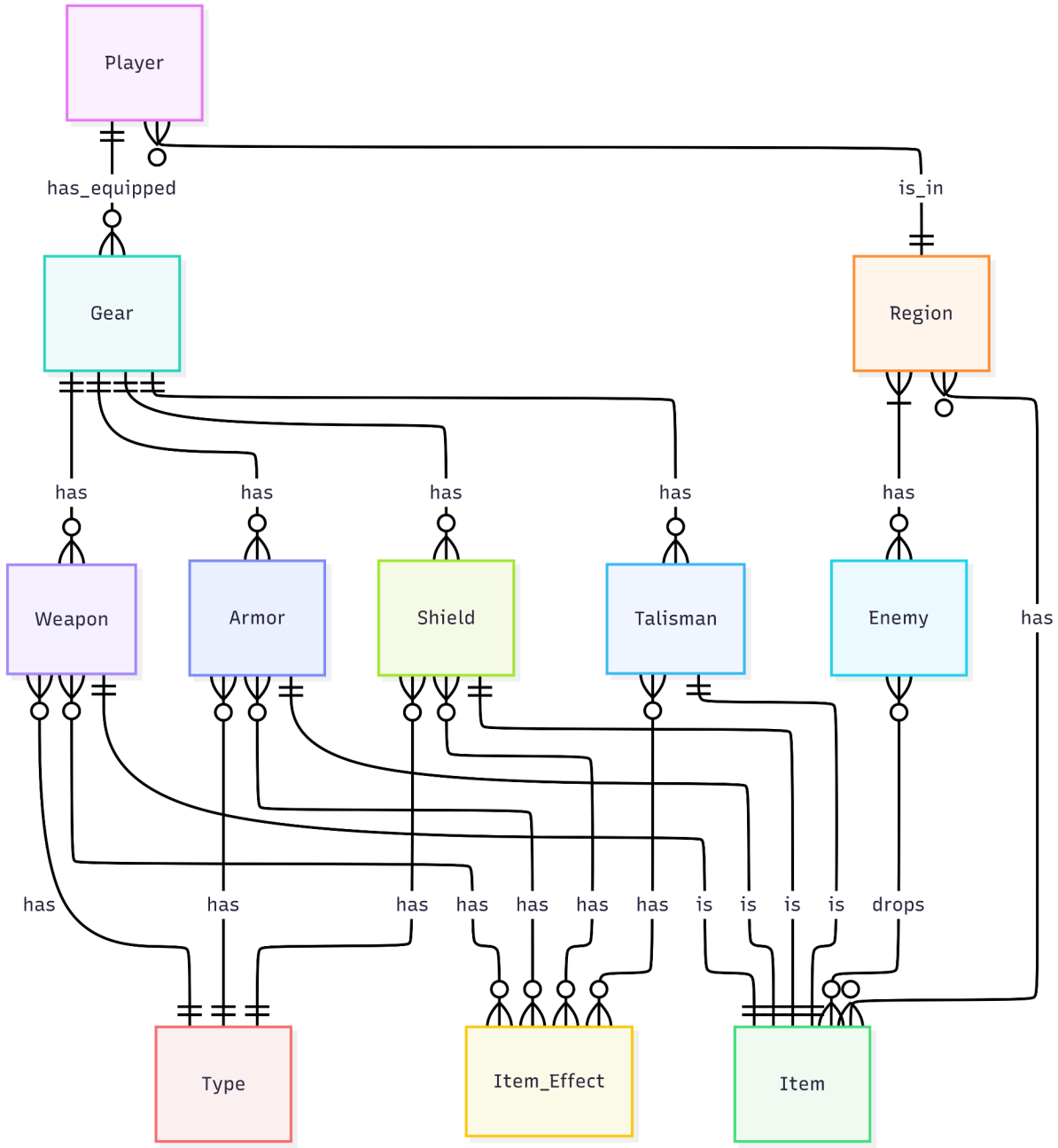
Spielerdaten werden live gespeichert und an den Server gesendet, damit der Spieler direkt sehen kann, welche Truhen und Sammelobjekte in der Nähe sind.

Die Gebiets-Checkliste zeigt bei einem Gebietswechsel an, welche Aufgaben der Spieler in dem neuen Gebiet hat.

Integritätsbedingungen

1. Jede Waffe hat immer genau einen Waffentyp.
2. Es werden keine Sonderzeichen benutzt.
3. Die Desktop-Anwendung sendet maximal 10 Anfragen pro Sekunde an die Datenbank.
4. Desktop Anwendungs Panels werden nicht über das GUI vom Spiel gerendert.
5. Es werden Timestamps für Daten benutzt.
6. Die Spielperformance sollte nicht stark von der Anwendung beeinflusst werden.

ER-Modell


Player
has_equipped
is_in
Gear
Region
has
has
has
has
has
Weapon
Armor
Shield
Talisman
Enemy
has
has
has
has
has
has
has
has
is
is
is
is
drops
Type
Item_Effect
Item

Entitäten

Player – Zu jeder echten Person gehört ein **Player**, der einen Charakter mit **Gear** beinhaltet. Konten von realen Menschen haben immer genau einen **Player**.
Gear – Durch diese speziellen **Items** bekommt der **Player** seine **Item-Effects** und sind alles, was momentan ausgerüstet ist. Dies wird live durch die Desktopanwendung mitverfolgt und wird bei Berechnungen jederzeit benutzt, um Statistiken zu realisieren.
Item-Effects – Die Acht Haupt-**Item-Effects** (Kraft, Geist, Kondition, Stärke, Geschick, Glaube und Arkane) helfen bei Rechnungen im Spiel, auf genaue Ergebnisse wie zum Beispiel einer Schadensberechnung zu kommen.
Type – Eine feste Menge von verschiedenen Arten wie Schwert, Axt oder Schild und beinhaltet Grundwerte (Schaden, Vorraussetzungen und Gewicht)
Weapon – Pro **Player** können bis zu zwei Einhandwaffen oder eine Zweihandwaffe ausgerüstet werden und bekommt seine Grund-**Item-Effects** aus den **Types** und modifiziert diese anhand von eigenen Modifikatoren. Darunter fällt zum Beispiel ein Schadensmodifikator von ×1.2, wenn der Träger der Waffe eine bestimmte Anzahl an Stärke hat.
Armor – Ist ähnlich wie die **Weapon**, doch können pro **Player** bis zu vier Rüstungsteile gleichzeitig angezogen werden.
Shield – Ist ähnlich wie die **Weapon**, doch kann pro **Player** maximal ein **Shield** ausgerüstet werden.
Talisman – Ist ähnlich wie die **Weapon**, doch können pro **Player** maximal vier **Talismans** ausgerüstet werden.
Item– Diese werden von **Enemies** fallen gelassen und vom **Player** ausgerüstet**.** Sie stärken den **Player** und geben diesem einzigartige Effekte und zugleich **Item-Effects**, wodurch sich der Spielstil verändert.
Enemy– Sind Entitäten, die in der Spielwelt herumlaufen und **Item-Effects** haben. Sie lassen **Items** fallen und werden je nach **Region**, in der sie sind, schwerer oder leichter.
Region – Diese sind mit Koordinaten abgegrenzt und werden benutzt, um zu bestimmen, in welchem Ort sich ein **Player** aktuell befindet und welche **Enemies** und **Items** hier aufzufinden sind.

Datenbankauswahl

Auswahlkriterien

Ergeben aus dem, was wir wirklich für das Projekt benötigen sieht unsere Gewichtung mit den Auswahlkriterien wie folgt aus:

Auswahlkriterien	Muss-Kriterium	Gewicht
Hohe Performanz	Ja	5
Daten-Streaming	Ja	3
Niedrige Latenz	Ja	3
Skalierbarkeit	Nein	1
Keine Lizenzkosten	Ja	5
Indexing	Nein	2
Vektorrechnungen	Nein	1
Präferenz	Nein	1

Vergleiche

Auswahlkriterien	MariaDB	Redis	MongoDB
Hohe Performanz	0	2	1
Daten-Streaming	0	2	0
Niedrige Latenz	1	2	0
Skalierbarkeit	1	1	1
Keine Lizenzkosten	1	1	1
Indexing	1	1	1
Vektorrechnungen	0	1	1
Präferenz	0	1	1
Zwischensumme	11	30	15

Da unsere App absolut latenzfrei und Performant sein muss, haben wir uns bei den Testergebnissen auf Redis geeinigt. Außerdem ist Redis besonders schnell, wenn es wenige Einträge gibt, was bei unserem Projekt mit ~100 Regionen, ~3,300 Gegenständen, ~500 Gegnern perfekt zusammen passt.

Begründung der Auswahlkriterien

Da unsere App stark auf Live-Daten aufbaut, haben wir die Kriterien entsprechend gewichtet. Hohe Performanz und keine Lizenzkosten sind für uns am wichtigsten. Datenstreaming und niedrige Latenz sind auch essentiell, damit die Infos aus dem Spiel ohne spürbare Verzögerung auf der Webseite und im Desktop Overlay ankommen. Kriterien wie Vektorrechnungen, Indizierung oder extreme Skalierbarkeit sind für unser Projekt momentan nicht relevant, da wir keine KI-Funktionen nutzen und die Datenbankgröße bei ~100 Regionen, ~500 Gegnern und ~3,300 Gegenständen überschaubar bleibt.

Organisatorisches

Meilensteine und Terminplan

1. Vorlesungswoche (20.04)
    a. Erste Projektidee anhand der Elden Ring Gruppe (Yasin)
    b. Brainstorming eines Projektes in Bezug auf Datenbanken (Alle)
2. Vorlesungswoche (27.04)
    a. Datenbank Auswahlkriterien festgelegt (Alle)
    b. Auswahl der in Betracht kommenden Datenbanken (Alle)
    c. Performance-Tests der Datenbanken mit Dummy Daten (Justin)
3. Vorlesungswoche (04.05)
    a. Einigung der zu benutzenden Datenbank (Alle)
    b. Anwendungsfälle definiert (Michelle)
    c. Er-modell definiert und Entitäten beschrieben (Yasin)
    d. Evaluation und finale Datenbankauswahl (Justin)
4. Vorlesungswoche (11.05)
    a. Proposal überarbeitet (Alle)
5. Vorlesungswoche (01.06)
    a. Zwischenpräsentation planen (Alle)
    b. Datenbanken auf Server aufsetzen (Justin)
    c. Datenbankschema planen (Michelle)
6. Vorlesungswoche (08.06)
    a. Zwischenpräsentation halten (Alle)
    b. Sehr kleine Mini Demo des projektes (Alle)
7. Vorlesungswoche (15.06)
    a. Demo Server erstellen (Justin)
    b. Website Mockups erstellen (Michelle)
8. Vorlesungswoche (22.06)
    a. Demo Client (Yasin)
    b. Demo Website (Michelle)
9. Vorlesungswoche (29.06)
    a. Stresstests (Yasin)
10. Vorlesungswoche (06.07)
    a. Endpräsentation vorbereiten (Alle)
11. Vorlesungswoche (13.07)
    a. Endpräsentation halten (Alle)