Modul 165

Leistungsbeurteilung LB2 – Dokumentation

Modul	IET-165 - Projektarbeit
Eingereicht von	INF2023F Naomi, Emil
Eingereicht bei	K. Abegglen
Datum	28. März 2025

Inhalt

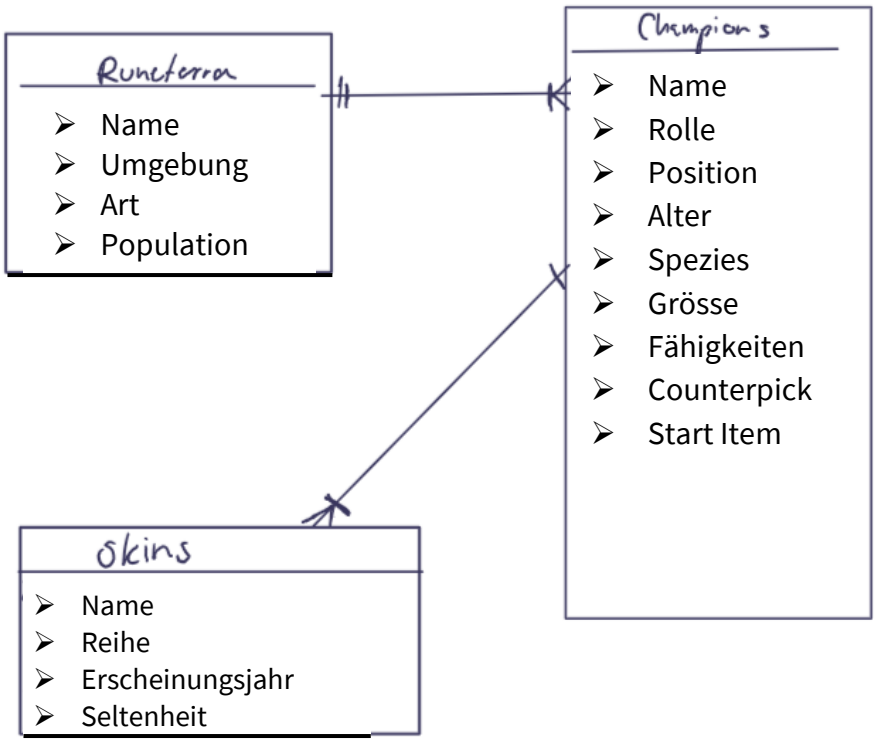
2 NoSQL-Datenbank

2.1 Projektidee und Begründung der Wahl der NoSQL-Datenbank

Wir kreieren eine App für neue Spieler von Leauge of Legends. Das Ziel ist, ihnen zu helfen sich in den rund 140 Champions besser zurecht zu finden. Und ihnen einen Vorteil zu verschaffen bevor sie in ihr erstes Spiel gehen. Vorteil darauf, wen sollen sie spielen, wenn ihr Gegner X spielt und mit welchen Items man starten soll.

Wir kreieren dies mit der Datenbank Anbindung von Neo4j. Wir haben uns auf Neo4j geeinigt, da Neo4j uns am besten hilft übereinstimmende Daten aufrecht zu erhalten und sie hilft uns die vielen ineinander verwickelten Beziehungen des Spiels simpel und verständlich darzustellen und zu speichern.

2.2 Konzeptionelles Datenmodell

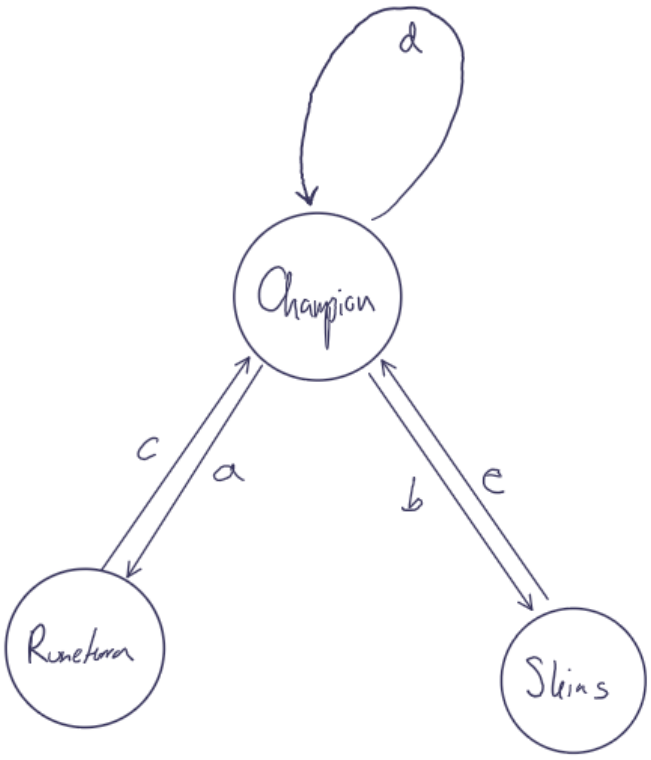


Champions -> Charaktere/ Helden im Spiel

Runeterra -> Ort /Welt von wo die Helden Kommen

Skins -> Kleider welche die Helden anziehen können (kein spiel vorteil)

2.3 Logisches Datenmodell



Nodes: Champion, Skins, Runeterra

Edges:

a) rules, dictates, fights_for, created, rebuilt, belongs_to, protects

b) owns

c) ruled_by, dictated_by, created_by, rebuilt_by, protected_by

d) loves, hates, likes, knows, sibling_of, parent_of, child_of, partner_of

e) belongs_to

2.4 Attribute der Hauptentitätsmengen

Unsere Attribute der Champions:

- Name
- Species
- Height
- Age
- Role
- Position
- Skills
- StartItems
- Counterpicks

2.5 Einfügen von Daten

Champion:

CREATE (:Champion {

name: "Ahri",

role: "Mage",

age: 300,

species: "Vastaya",

height: 1.68 m,

skills: ["Orb of Deception", "Fox-Fire", "Charm", "Spirit Rush"],

```
  counterpick: ["Kassadin", "Zed", "LeBlanc"],
  startItems: ["Doran's Ring", "Health Potion"]
});
```

Skin:

```
CREATE (:Skin {
  name: "K/DA Ahri",
  series: "K/DA",
  uploadyear: 2018,
  rarity: "Epic"
});
```

Runeterra:

```
CREATE (:Runeterra {
  name: "Ionia",
  ecosystem: "magic flatland ",
  politics: "Kingdome",
  population: 5000000
});
```

2.6 Ändern und Löschen von Daten

Ändern:

```
MATCH (c:Champion {name: "Ahri"})
SET c.rolle = "Assassine"
RETURN c;

MATCH (s:Skin {name: "K/DA Ahri"})
SET s.rarity = "Legendary"
RETURN s;

MATCH (r:Runeterra {name: "Ionia"})
SET r.population = 6000000
RETURN r;
```

Löschen:

```
MATCH (c:Champion {name: "Ahri"})
DETACH DELETE c;

MATCH (s:Skin {name: "K/DA Ahri"})
DETACH DELETE s;

MATCH (r:Runeterra {name: "Ionia"})
DETACH DELETE r;
```

2.7 Dynamische Daten

Yasuo ohne alter, position und rolle:

```
CREATE (:Champion {
  name: "Yasuo",
  spezies: "Mensch",
  grösse: 1.75 m,
  skills: ["Steel Tempest", "Wind Wall", "Sweeping Blade", "Last Breath"],
  startItems: ["Doran's Blade", "Health Potion"],
  counterpick: ["Teemo", "Malphite"]
});
```

Kennen ohne grösse und position:

```
CREATE (:Champion {
  name: "Kennen",
  rolle: "Magier",
  alter: 25,
  spezies: "Yordle",
  skills: ["Thundering Shuriken", "Lightning Rush", "Electrical Surge", "Slicing Maelstrom"],
  counterpick: ["Teemo", "Malphite", "Kennen"],
  startItems: ["Doran's Ring", "Health Potion"]
});
```

2.8 Anzeigen von Daten

Einzeln:

MATCH (c:Champion {name: "Ahri"})

RETURN c;

MATCH (s:Skin {name: "K/DA Ahri"})

RETURN s;

MATCH (r:Runeterra {name: "Ionia"})

RETURN r;

Alle:

MATCH (n)

RETURN n;

MATCH (c:Champion)

RETURN c;

MATCH (s:Skin)

RETURN s;

MATCH (r:Runeterra)

RETURN r;

Alle Champions mit ihrer Rolle anzeigen:

MATCH (c:Champion)

RETURN c.name, c.rolle;

Alle Skins eines bestimmten Champions anzeigen:

MATCH (c:Champion {name: "Ahri"})-[:HAS_SKIN]->(s:Skin)

RETURN c.name, s.name, s.reihe, s.erscheinungsjahr, s.rarity;

Alle Champions aus einer bestimmten Runeterra-Region anzeigen:

MATCH (c:Champion)-[:BELONGS_TO]->(r:Runeterra {name: "Ionia"})

RETURN c.name, r.name;

3 Datenbankoperationen und -architektur

3.1 Zugriffsberechtigungen: Konzept

Wir wollen folgende User:

- Admin — Alles: Zugriff auf alle DBs und auf alle CRUD Optionen
- AppUser — Zugriff auf 1 DB und auf read & create Optionen
- BackupUser — Zugriff auf 1 DB, Backup Tools, jedoch keine CRUD Optionen
- MainainUser — Zugriff auf 1 DB, Logs usw., jedoch keine CRUD Optionen

3.2 Zugriffsberechtigung: Umsetzung

create user admin set password "sml12345" change not required
create user app_user set password "password1" change not required
create user backup_user set password "password2" change not required
create user maintain_user set password "password3" change not required

create role admin
create role app
create role backup
create role maintain

grant role admin to admin
grant role app to app_user
grant role backup to backup_user
grant role maintain to maintain_user

```
GRANT ALL PRIVILEGES ON DBMS TO admin;
GRANT ALL DATABASE PRIVILEGES ON DATABASE * TO admin;
```

```
GRANT TRAVERSE ON GRAPH dataoflegends TO app;
GRANT READ {*} ON GRAPH dataoflegends TO app;
GRANT CREATE ON GRAPH dataoflegends TO app;
GRANT MATCH {*} ON GRAPH dataoflegends TO app;
```

```
GRANT TRAVERSE ON GRAPH dataoflegends TO backup;
GRANT READ {*} ON GRAPH dataoflegends TO backup;
GRANT ACCESS ON DATABASE dataoflegends TO backup;
```

```
GRANT EXECUTE PROCEDURE db.dump ON DBMS TO backup;
GRANT EXECUTE PROCEDURE db.backup ON DBMS TO backup;

GRANT TRAVERSE ON GRAPH dataoflegends TO maintain;
GRANT READ {*} ON GRAPH dataoflegends TO maintain;
GRANT ACCESS ON DATABASE dataoflegends TO maintain;
GRANT EXECUTE PROCEDURE apoc.log.* ON DBMS TO maintain;
```

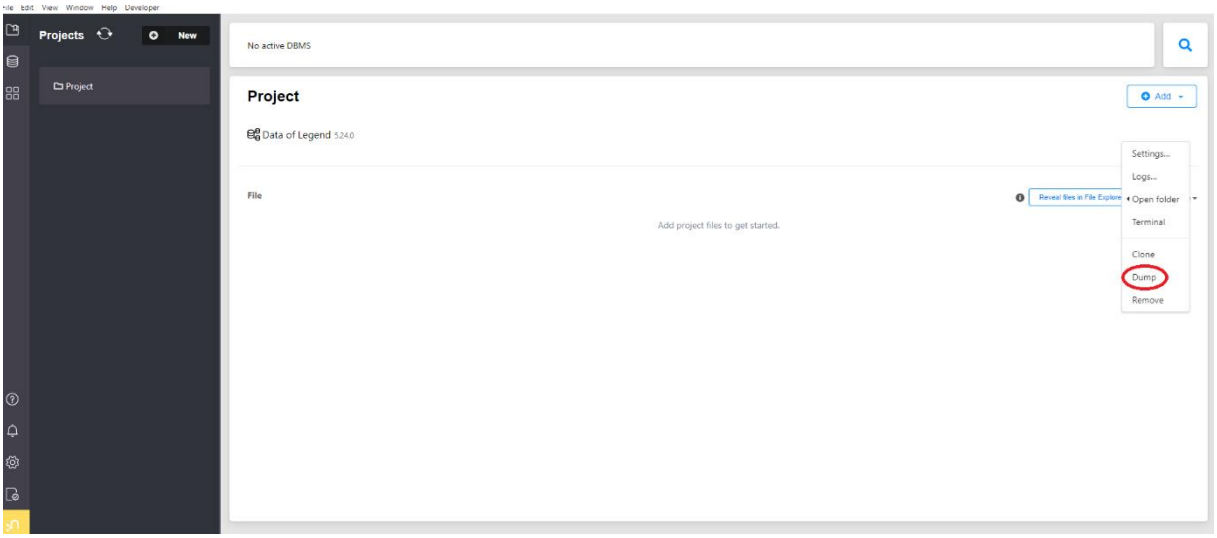
revoke role admin from admin
revoke role app from app_user
revoke role backup from backup_user
revoke role maintain from maintain_user

drop role admin
drop role app
drop role backup
drop role maintain

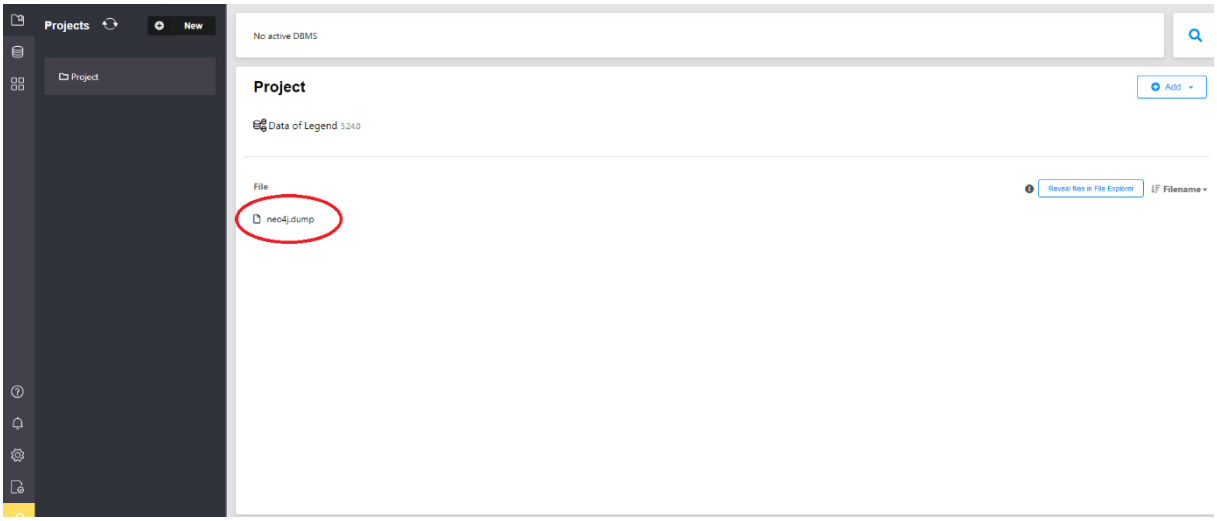
drop user admin
drop user app_user
drop user backup_user
drop user maintain_user

3.3 Backup der DB

1. Auf die drei Punkte bei Datenbank klicken und dann Dump auswählen.

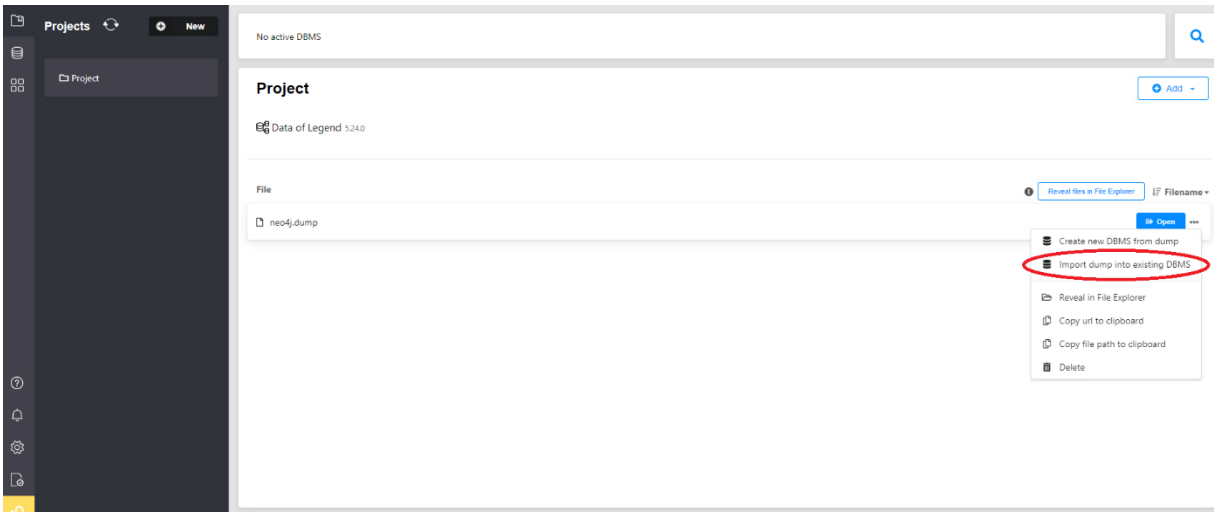


2. Der erstellte Dump.



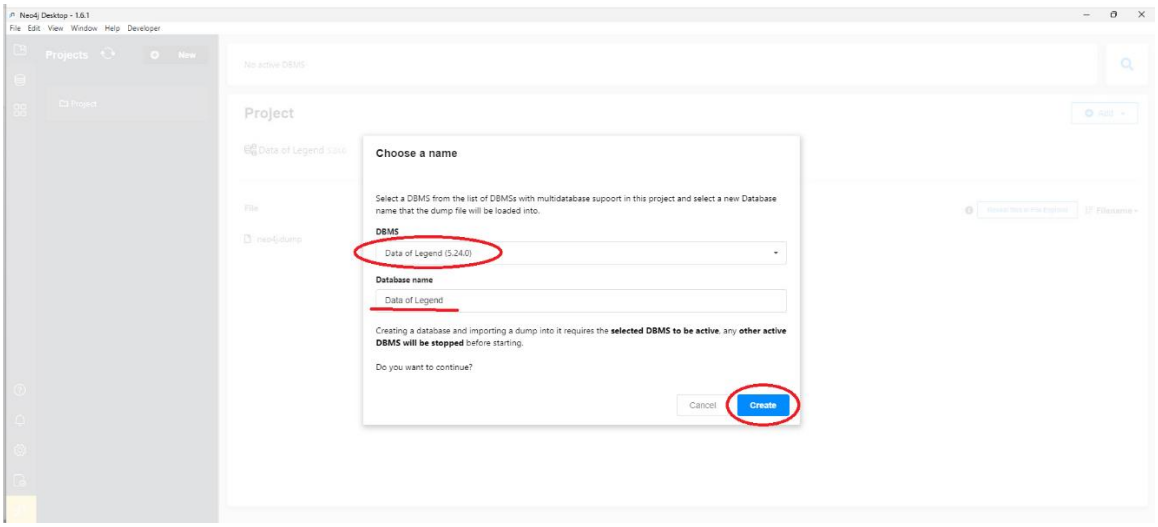
3.4 Restore eines DB-Backups

1. Auf den Dump klicken und Import into DBMS auswählen.

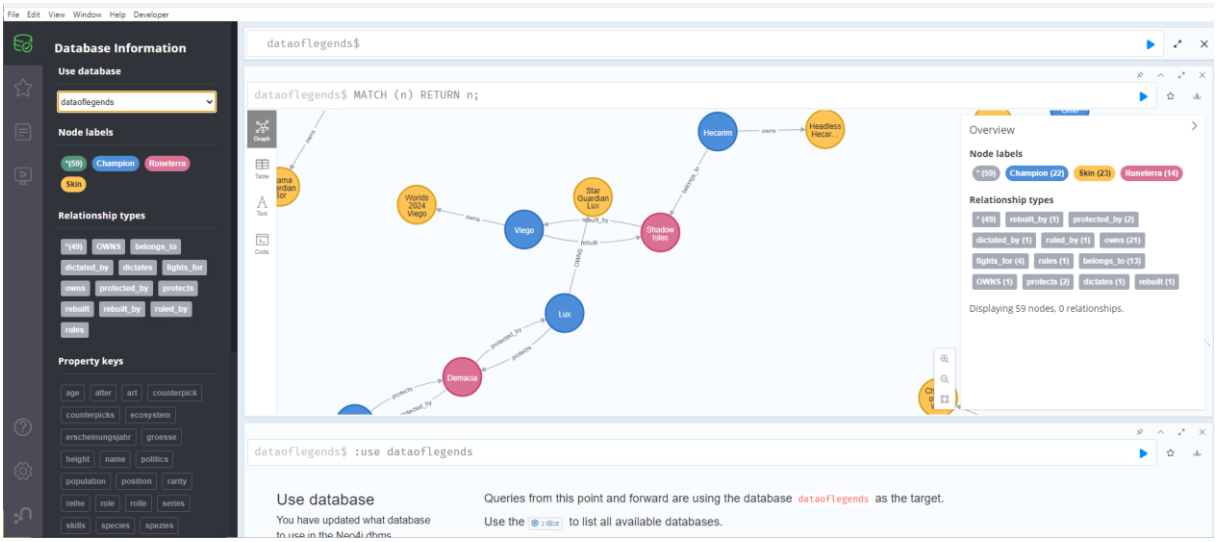


2. Die richtige DBMS auswählen, neuen Namen erstellen und

dann Create.



3. Die neue Datenbank auswählen und überprüfen ob alle Daten vorhanden.



3.5 Konzept für horizontale Skalierung

Geografisches Sharding basierend auf der Spielwelt

Die Datenbank würde horizontal skaliert werden, indem Daten auf Basis der natürlichen geografischen Einteilungen der Spielwelt - den Regionen von Runeterra - partitioniert werden. Dieser Ansatz bringt die technische Architektur in Einklang mit der narrativen Struktur des Spiels.

Implementierungsdetails:

- Jeder Shard enthält vollständige Daten einer oder mehrerer Runeterra-Regionen
- Beispiel für die Verteilung:
  - Shard 1: Demacia, Noxus (militärische Regionen)
  - Shard 2: Ionia, Freljord (mystische/elementare Regionen)
  - Shard 3: Piltover, Zaun (technologische Regionen)
  - Shard 4: Schatteninseln, Bilgewater (dunklere/gesetzlose Regionen)

- Shard 5: Shurima, Ixtal, Targon (antike/himmlische Regionen)

Logik der Datenverteilung:

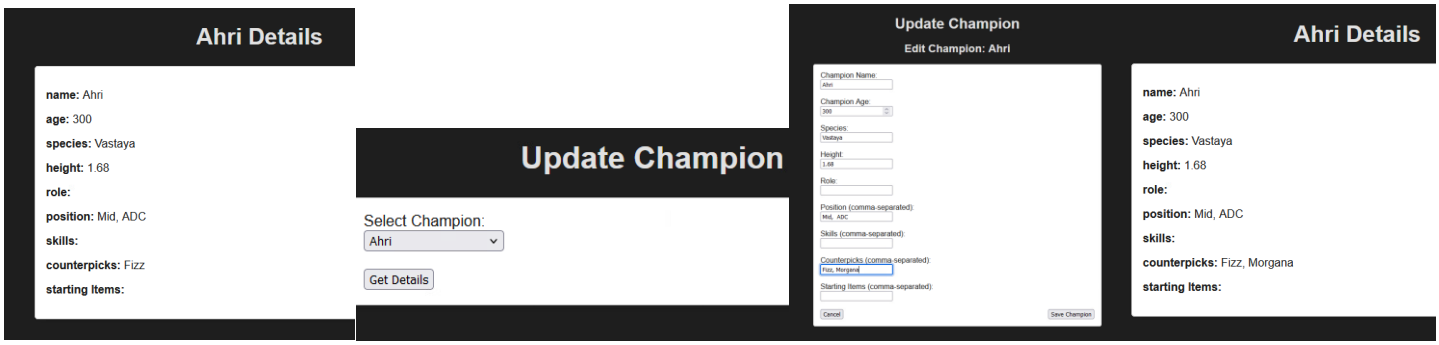
- Champions, Skins und Beziehungen werden auf dem Shard gespeichert, der ihrer Heimatregion entspricht
- Regionsübergreifende Beziehungen werden durch Referenzlinks verwaltet
- Jeder Shard unterhält seine eigene Neo4j-Instanz mit regionsspezifischen Daten

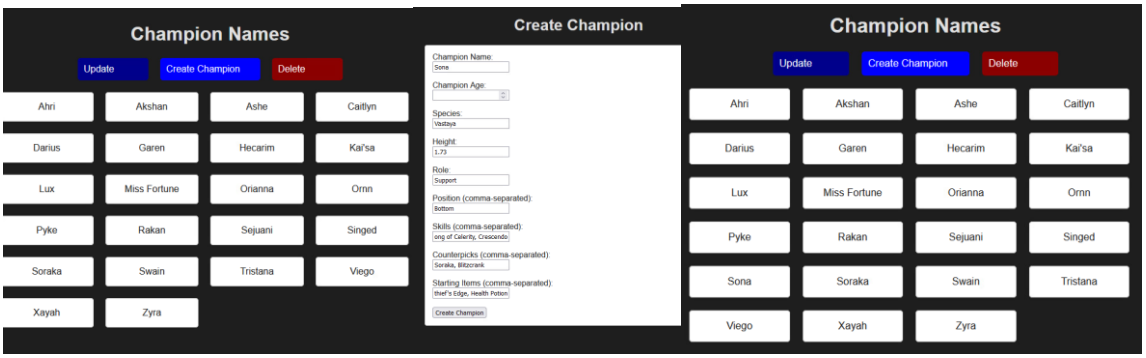
Vorteile:

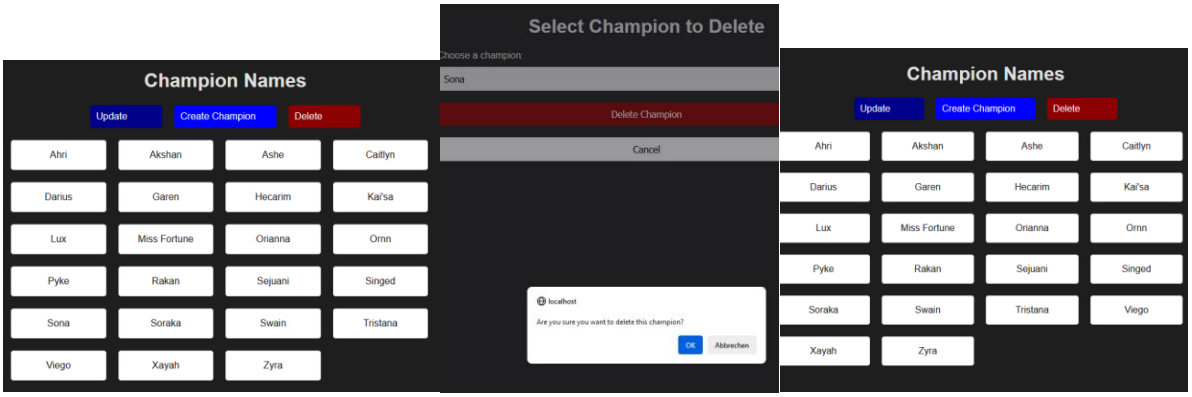
- Natürliche Datenaufteilung, die dem Spielkontext folgt
- Ermöglicht regionsspezifische Optimierungen (z.B. unterschiedliche Indexstrategien für kampflastige vs. magielastige Regionen)
- Reduziert Datenabhängigkeiten zwischen Shards
- Erlaubt gezielte Skalierung basierend auf der Beliebtheit von Regionen
- Verbessert die Abfrageleistung für regionsspezifische Suchen

4 Applikation

4.2 Daten einfügen, ändern, löschen







4.5 Technologie / Aufbau der Applikation

4.5.1 Eingesetzte Technologien

Unsere Applikation basiert auf folgenden Technologien:

1. **Neo4j** als Graphdatenbank:
   - Perfekt geeignet für die komplexen Beziehungen zwischen Champions, Skins und Runeterra-Regionen
   - Ermöglicht intuitive Abfragen der Verbindungen zwischen Champions (loves, hates, sibling_of, etc.)
   - Visualisierung der Graphen hilft neuen Spielern, Zusammenhänge besser zu verstehen
2. **PHP** für das Backend:
   - Einfache Integration mit Neo4j über HTTP-Requests (cURL)
   - Breite Unterstützung auf Webservern
   - Verarbeitung der Datenbankergebnisse und dynamische Seitengenerierung
3. **HTML/CSS** für das Frontend:
   - Responsive Design für verschiedene Bildschirmgrössen
   - Einheitliches Styling mit dunklem Grundthema (ähnlich wie League of Legends selbst)
   - Einfache Navigation durch die Champions
4. **JSON** für Datenaustausch:
   - Effiziente Kommunikation zwischen Neo4j und PHP-Backend
   - Strukturierte Datendarstellung

4.5.2 Begründung der Technologiewahl

Wir haben uns für diese Technologien entschieden, weil:

- **Neo4j**: Die Graphdatenbank ermöglicht es uns, komplexe Beziehungen zwischen Champions darzustellen, die in einer relationalen Datenbank viel schwieriger zu modellieren wären. Dies ist ideal für unser Anwendungsszenario, da wir verschiedene Verbindungstypen zwischen Champions (liebt, hasst, Geschwister), zwischen Champions und Regionen (gehört zu, beschützt) sowie zwischen Champions und Skins (besitzt) darstellen müssen.
- **PHP**: Als serverseitige Skriptsprache bietet PHP eine einfache Integration mit Datenbanken und ermöglicht dynamische Webseiten. Die cURL-Bibliothek macht die Kommunikation mit der Neo4j-API unkompliziert. Zudem ist PHP auf den meisten Webhostings verfügbar, was die spätere Veröffentlichung der Anwendung erleichtert.
- **HTML/CSS**: Diese Standardtechnologien für Webseiten ermöglichen uns, eine benutzerfreundliche Oberfläche zu erstellen, die auf verschiedenen Geräten funktioniert. Das dunkle Farbschema passt zur ästhetischen Gestaltung von League of Legends.

4.5.3 Struktur des Codes

Unsere Anwendung folgt einem vereinfachten MVC-Muster (Model-View-Controller):

1. **Model**: Die Neo4j-Datenbank dient als unser Datenmodell. Die Cypher-Queries in den PHP-Dateien interagieren mit diesem Modell.
2. **View**: Die HTML-Ausgabe in den PHP-Dateien bildet die View-Schicht, die für die Präsentation der Daten verantwortlich ist.
3. **Controller**: Die Logik in den PHP-Dateien fungiert als Controller, der Benutzeranfragen verarbeitet und die entsprechenden Daten aus dem Modell abfragt.

4.5.4 Dateien und ihre Funktionen:

- **Home.php**: Hauptseite der Anwendung, zeigt alle Champions alphabetisch sortiert an und bietet Navigationsoptionen für CRUD-Operationen
- **Champ.php**: Seite mit allen Champions
- **Rune.php**: Seite mit allen Regionen von Runeterra
- **Skin.php**: Seite mit allen Skins
- **…_display.php**: Detailansicht einer Identität mit allen Attributen
- **create.php**: Formular zum Erstellen neuer Champions und neuer Skins (2 Files: C&S)
- **update.php**: Formular zum Aktualisieren vorhandener Champions, Regionen und Skins (3Files: C,R&S)
- **delete.php**: Funktion zum Löschen von Champions und Skins (2 Files, C&S)

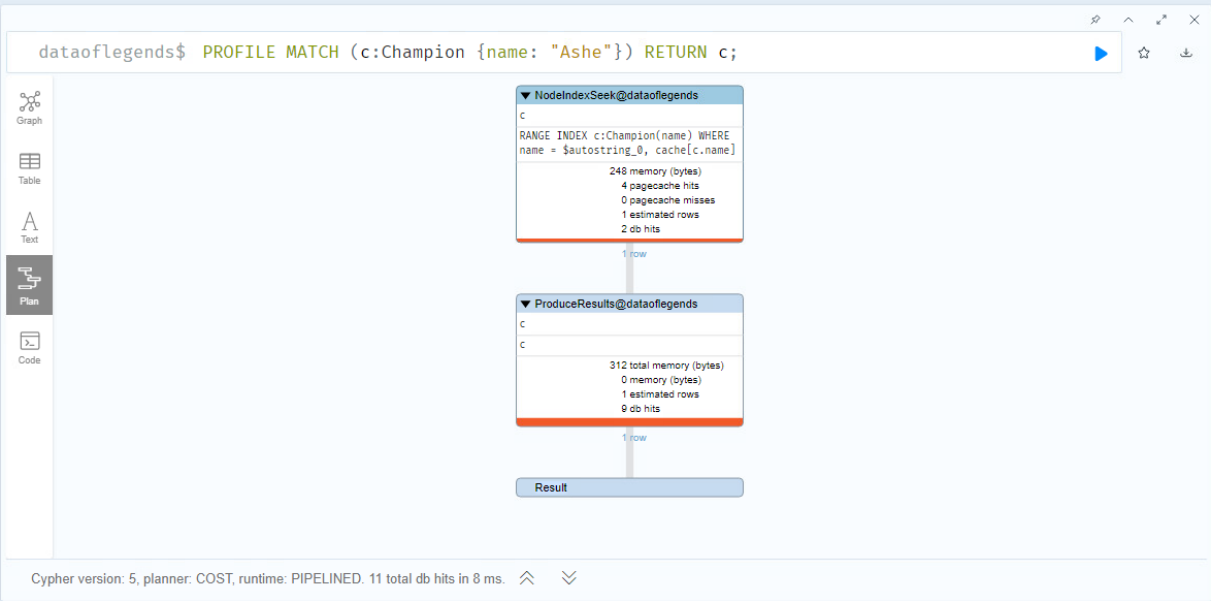
Der Datenaustausch zwischen Frontend und Backend erfolgt über HTTP-Requests, wobei die Ergebnisse der Neo4j-Abfragen als JSON zurückgegeben und von PHP verarbeitet werden.

Das Design ist modular aufgebaut, sodass neue Funktionen (wie z.B. neue Entitäten) einfach hinzugefügt werden können, ohne die bestehende Struktur zu beeinträchtigen.

5 Weitere Kompetenzen (optional)

5.1 Indizes

```
// Index für Champions nach Name (häufiges Suchkriterium)

CREATE INDEX champion_name_index FOR (c:Champion) ON (c.name);


// Index für Skins nach Name

CREATE INDEX skin_name_index FOR (s:Skin) ON (s.name);


// Index für Runeterra-Regionen

CREATE INDEX runeterra_name_index FOR (r:Runeterra) ON (r.name);


// Index für Champions nach Rolle

CREATE INDEX champion_role_index FOR (c:Champion) ON (c.role);


// Composite-Index für Champions nach Role und Position

CREATE INDEX champion_role_position_index FOR (c:Champion) ON (c.role, c.position);


SHOW INDEXES;


PROFILE MATCH (c:Champion {name: "Ashe"}) RETURN c;
```



Die Statistiken zeigen:

RANGE INDEX c:Champion(name) WHERE name = $autostring_0, cache[c.name] - Ein Bereichsindex auf Champion-Namen wird verwendet

NodeIndexSeek@dataoflegends:

- 248 memory (bytes) - Speicherverbrauch
- 4 pagecache hits - Anzahl der Cache-Treffer
- 0 pagecache misses - Keine Cache-Fehlschläge
- 1 estimated row - Es wird ein Ergebnis erwartet
- 2 db hits - Nur 2 Datenbankoperationen

ProduceResults@dataoflegends:

- 312 total memory (bytes) - Gesamtspeicherverbrauch
- 0 memory (bytes) - Kein zusätzlicher Speicher für diese Operation
- 1 estimated rows - Ein Ergebnis wird zurückgegeben
- 0 db hits - Keine weiteren Datenbankoperationen

5.6 Professionelles Design, intuitive Bedienung

Für die Einfachheit und die Standardisierung brauche ich Rottöne (dunkelrot) um Zurück-buttons und Delete-buttons zu definieren. Für Create-buttons und Update-buttons brauchte ich Blautöne da diese als Beruhigend und gelassen wirken. Als Hintergrundfarbe nahm ich schwarz da die Zielgruppe unserer Applikation Gamer sind und diese Dark-mode meist präferieren. Für sonstige Objekte brauchte ich weiss da es eine gewisse Reinigkeit und Richtigkeit ausstrahlt ,die die professionelle Wirkung unserer Applikation verstärken soll.

6 Arbeitsjournal, Reflexionen

6.1 Journal Naomi

Datum: 06.03.2025
Was habe ich erledigt? Wie bin ich vorgegangen?
Unterkapitel 2.1-2.4 erstellt und frisiert, mitgeholfen bei 2.4-2.8.
Was waren die Schwierigkeiten? Wie habe ich sie gelöst?
Bei Skins dir richtige Beziehung zu finden, ich habe jetzt 2 Pfeile mit je 1 Beziehung.
Was sind die Erkenntnisse? Was ist gut gelaufen, was weniger?
Es geht sehr schnell, sobald man eine Idee hat, eine DB aufzusetzen. Sehr produktiv gearbeitet.
Wie geht es weiter? Was will ich besser machen?
Nächstes Mal: Kap 3 – evtl. 4 erledigen & abgeben.

Datum: 13.03.2025
Was habe ich erledigt? Wie bin ich vorgegangen?
Abgabe 2 | Berechtigungen 3.1-3.2
Was waren die Schwierigkeiten? Wie habe ich sie gelöst?
Umsetzung der Rollen, recherche
Was sind die Erkenntnisse? Was ist gut gelaufen, was weniger?
Die Planung lief sehr gut und auch die Gruppen-kommunikation
Wie geht es weiter? Was will ich besser machen?
Abgabe am FR, Kapitel 4 machen

Datum: 20.03.2025
Was habe ich erledigt? Wie bin ich vorgegangen?
Kapitel 4 umsetzen (bis 4.3)
Was waren die Schwierigkeiten? Wie habe ich sie gelöst?
Ich hatte nicht wirklich probleme

Was sind die Erkenntnisse? Was ist gut gelaufen, was weniger?
Ich kam schnell voran und lernte neues dazu (php)
Wie geht es weiter? Was will ich besser machen?
Bis nächstes Mal 4.5 schaffen und nächstes Mal 5.5 umsetzen

Datum: 27.03.2025
Was habe ich erledigt? Wie bin ich vorgegangen?
Kapitel 5.5, 5.6, 4.2, 4.4 abgegeben
Was waren die Schwierigkeiten? Wie habe ich sie gelöst?
Ich hatte nicht wirklich Probleme aus Kopfschmerzen (php)
Was sind die Erkenntnisse? Was ist gut gelaufen, was weniger?
PHP ist simpel jedoch Kopfschmerzen erregend, jedoch habe ich sehr viel gelernt über PHP.

6.2 Journal Emil

Datum: 06.03.2025
Was habe ich erledigt? Wie bin ich vorgegangen?
Ich habe die Punkte 2.4-2.8 erstellt, mitgeholfen bei 2.1-2.4.
Was waren die Schwierigkeiten? Wie habe ich sie gelöst?
Ich hate keine Schwierigkeiten.
Was sind die Erkenntnisse? Was ist gut gelaufen, was weniger?
Ich konnte alles gut und produktiv Lössen.
Wie geht es weiter? Was will ich besser machen?
Nächstes Mal, Kapitel 3-4 weitermachen.

Datum: 13.03.2025
Was habe ich erledigt? Wie bin ich vorgegangen?
Ich habe die Punkte 3.3 -3.4 gemacht.
Was waren die Schwierigkeiten? Wie habe ich sie gelöst?
Ich hate keine Schwierigkeiten.
Was sind die Erkenntnisse? Was ist gut gelaufen, was weniger?
Ich konnte alles gut und produktiv Lössen.
Wie geht es weiter? Was will ich besser machen?
Kapitel 4 machen

Datum: 20.03.2025/27.03.2025
Was habe ich erledigt? Wie bin ich vorgegangen?
Ich habe die Punkte 3.5, 4,5 und 5.1 gemacht.
Was waren die Schwierigkeiten? Wie habe ich sie gelöst?
Ich habe 5.6 versucht habe es aber nicht geschaft
Was sind die Erkenntnisse? Was ist gut gelaufen, was weniger?
Ich habe vieles neues gelernt
Wie geht es weiter? Was will ich besser machen?
Abgeben :D