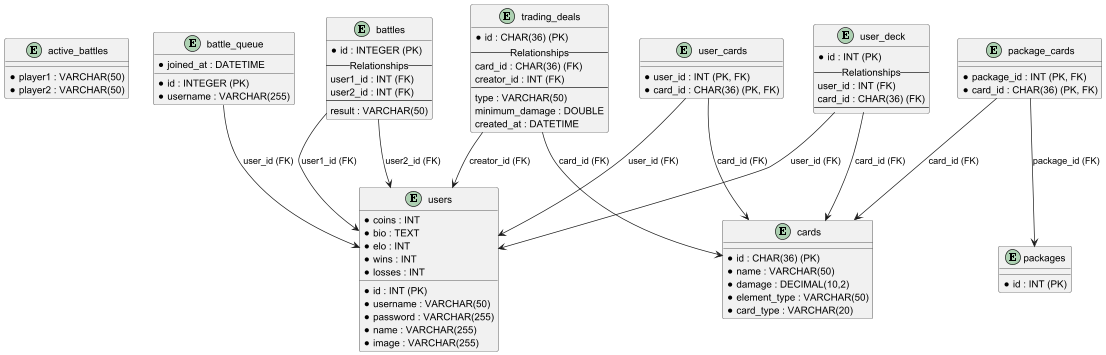
**Projektprotokoll**

**Git Repository:** <https://github.com/MrPlotTwist/MTCG>

UML DIAGRAMM:



**1. Technische Schritte**

**Designs**

* **Datenbankschema:**
  + **Die Datenbank wurde mit den folgenden Tabellen entworfen: users, cards, battles, trading\_deals, user\_deck, user\_cards, packages und package\_cards.**
  + **Beziehungen wurden mit Primär- und Fremdschlüsseln definiert. Zum Beispiel referenziert user\_id in user\_cards die Tabelle users.**
  + **Das Schema wurde mit PlantUML visualisiert, um Korrektheit und ein klares Verständnis der Beziehungen sicherzustellen.**
* **Unit-Tests:**
  + **Eine Reihe von Unit-Tests wurde entwickelt, um die kritischen Funktionen des Systems zu validieren, wie z. B. die Benutzerregistrierung, den Kartentausch, die Deckkonfiguration und die Kampflogik.**
  + **Die Tests wurden geschrieben, um die ordnungsgemäße Funktion von PostgreSQL-spezifischen Abfragen sicherzustellen.**
* **Code-Integration:**
  + **Es wurden modulare Funktionen zur Verwaltung von Datenbankoperationen implementiert (z. B. Benutzerregistrierung, Handelsgeschäfte).**

**2. Unit-Tests: Auswahl und Bedeutung**

**Ausgewählte Unit-Tests**

1. **Benutzerregistrierung:** 
   * **Tests zur Registrierung neuer Benutzer und zum Umgang mit Duplikaten.**
2. **Benutzeranmeldung:** 
   * **Validiert korrekte Anmeldedaten und Fehlerbehandlung bei falschen Passwörtern.**
3. **Kartentausch:** 
   * **Stellt sicher, dass Handelsgeschäfte korrekt erstellt, verarbeitet und validiert werden.**
4. **Deckkonfiguration:** 
   * **Überprüft, ob Benutzer gültige Decks mit ihren Karten konfigurieren können.**
5. **Kampfsystem:** 
   * **Testet die Kampflogik, einschließlich Schadensberechnung, Elementmultiplikatoren und ELO-Updates.**
6. **Datenbankintegrität:** 
   * **Stellt sicher, dass Fremdschlüsseleinschränkungen und kaskadierende Löschvorgänge wie erwartet funktionieren.**

**Warum diese Tests wichtig sind**

* **Kernfunktionalität:** 
  + **Die Tests decken die kritischen Funktionen des Systems ab, wie z. B. Benutzerverwaltung, Kartentausch und Kämpfe, die für die Funktion der Anwendung unerlässlich sind.**
* **Validierung der Geschäftslogik:** 
  + **Stellt sicher, dass Regeln wie ELO-Berechnung, Kartenbesitz und Deckvalidierung eingehalten werden.**
* **Datenintegrität:** 
  + **Überprüft, ob die Datenbank konsistent bleibt, selbst bei komplexen Vorgängen wie Tauschgeschäften und kaskadierenden Löschvorgängen.**
* **Skalierbarkeit:** 
  + **Die Tests stellen sicher, dass das System mehrere Benutzer und Transaktionen ohne Probleme verarbeiten kann.**

**3. Zeitaufwand**

**Tätigkeiten und Zeitplan**

* **29. Dezember 2024:**
  + **Beginn des Projekts mit der Erstellung neuer Klassen.**
  + **Erstellung des initialen Datenbankschemas und der Beziehungen. (7 Stunden)**
* **1. Januar 2025:**
  + **Überarbeitung der README-Datei und Hinzufügen weiterer Klassen und Funktionen. (4 Stunden)**
* **2. Januar 2025:**
  + **Erstellung des ersten Entwurfs der Datenbank mit PostgreSQL und grundlegende Funktionen. (7 Stunden)**
* **3. Januar 2025:**
  + **Datenbankimplementierung abgeschlossen und erste Tests vorbereitet. (8 Stunden)**
* **4. Januar 2025:**
  + **Entwicklung der zentralen Unit-Tests für Benutzerregistrierung, Anmeldung und Kartenverwaltung. (6 Stunden)**
* **5. Januar 2025:**
  + **Implementierung der Kampflogik und des Handelssystems abgeschlossen. (6 Stunden)**
* **8. Januar 2025:**
  + **Visualisierung des Datenbankschemas mit PlantUML.**
  + **Debugging und Verfeinerung (3 Stunden)**
  + **Abschluss der Unit-Tests für alle kritischen Komponenten. (4 Stunden)**
  + **Dokumentation des Prozesses und Erstellung des Protokolls. (2 Stunden)**

**Geschätzte Stunden: 48 Stunden**

**4. Schwierigkeiten**

**Datenbanklogik erstellen**

* **Herausforderung:** 
  + **Die korrekte Definition und Verknüpfung von Tabellen und Beziehungen war komplex, insbesondere bei der Einhaltung von Datenintegrität und Performance.**
* **Lösung:** 
  + **Verwendung von Primär- und Fremdschlüsseln zur Sicherstellung der Datenkonsistenz.**
  + **Debugging und Verfeinerung**

**Unit-Tests korrekt erstellen**

* **Herausforderung:** 
  + **Sicherzustellen, dass die Tests nicht nur die Funktionalität, sondern auch Grenzfälle und Fehlerzustände abdecken.**
* **Lösung:** 
  + **Entwicklung einer umfassenden Testabdeckung für Benutzerregistrierung, Kartentausch, Deckkonfiguration und Kampflogik.**

**Kampflogik erstellen**

* **Herausforderung:** 
  + **Die Implementierung einer dynamischen und korrekten Kampflogik, die Schadensberechnungen und Elementmultiplikatoren berücksichtigt.**
* **Lösung:** 
  + **Erstellen eines flexiblen Kampfsystems mit klar definierten Regeln und Multiplikatoren.**
  + **Umfassende Testszenarien, um die Funktionalität sicherzustellen.**

**Handelssystem erstellen**

* **Herausforderung:** 
  + **Sicherzustellen, dass Handelsgeschäfte zwischen Spielern korrekt validiert und verarbeitet werden.**
* **Lösung:** 
  + **Implementierung von Logiken zur Validierung von Karten, Benutzerrechten und Handelsbedingungen.**
  + **Integration von Tests zur Simulation realistischer Handelsszenarien.**

**5. Git-Historie als Dokumentation**

* **Die Git-Commit-Historie verfolgt alle Änderungen, die am Projekt vorgenommen wurden, einschließlich Schema-Anpassungen, Testentwicklung und Fehlerbehebungen.**
* **Jeder Commit ist korrekt beschriftet, um die durchgeführte Arbeit widerzuspiegeln und eine transparente Zeitleiste für den Fortschritt des Projekts zu bieten.**

**6. Lessons Learned**

* **Bessere Zeitplanung:**
  + **Während des Projekts wurde deutlich, dass eine effizientere Einteilung der Arbeitszeit notwendig ist, insbesondere bei der Verteilung von Entwicklungs- und Debugging-Zeit.**
* **Mehr Zeit für Planung:**
  + **Die Planung der Datenbankstruktur und die Konzeption der Logik erforderten mehr Zeit als erwartet. Ein ausführlicher Plan hätte potenzielle Probleme früher identifizieren können.**

**7. Unique Feature**

* **Damage Booster:**
  + **Ein einzigartiges Feature des Kampfsystems ist der "Damage Booster", der es jedem Spieler erlaubt, den Damage einer Karte zu boosten. Jede 2. Runde kann mit einer 15% Wahrscheinlichkeit die Karte des Spielers um 50% geboostet werden (Damage + 50%).**
  + **Durch dieses Feature wird das Battle mehr Glück abhängig und kann schnell aus einer verlierenden Position eine gewinnende erschaffen, wodurch das Spiel spannend bleibt.**