Meilenstein 3

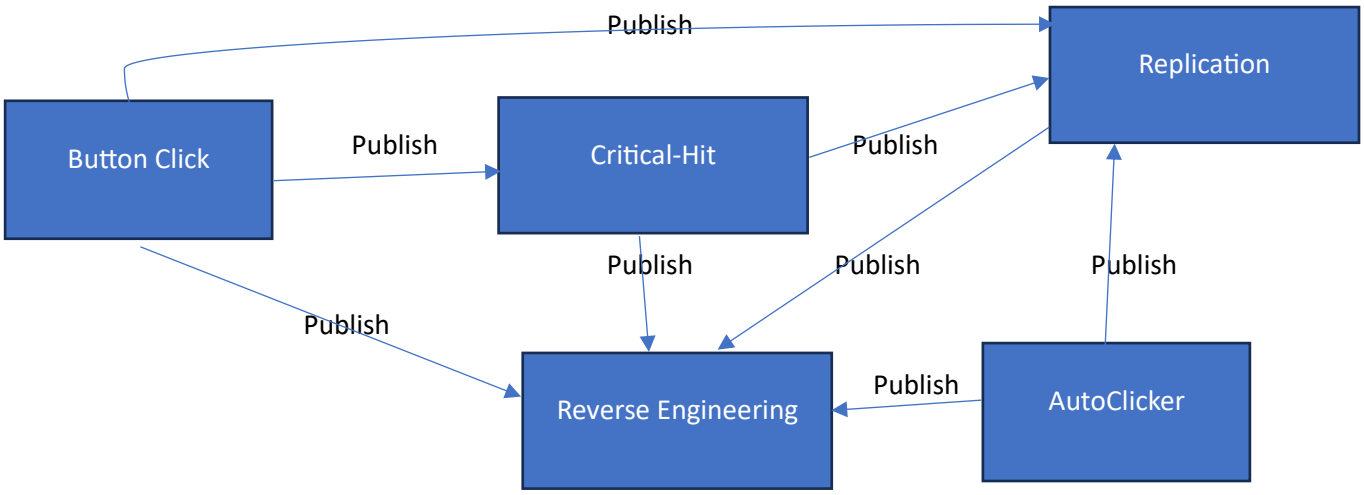
Zweiter Entwurf Server

Implementierung Effekte

Der Server enthält nun die Backend-Logik für die ersten Effekte. Dazu kam das Package ***effects*** hinzu:

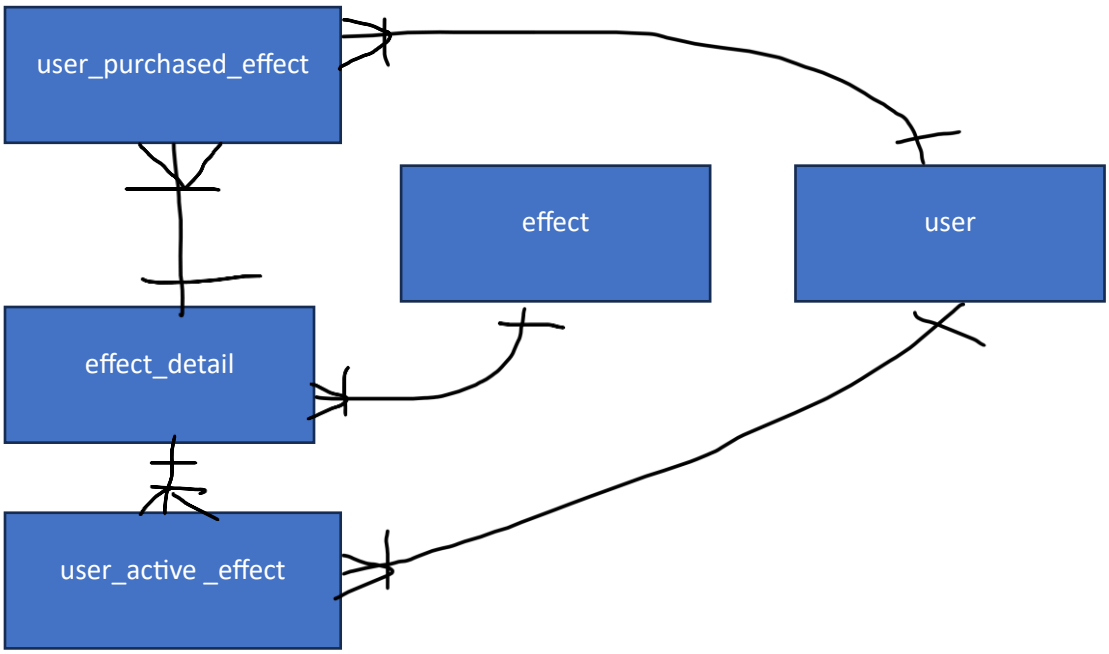
- Single-effects: enthält individuelle Logik für die einzelnen Effekte
    - Autoclicker-effect.ts: Im Sekundentakt wird eine gewisse Anzahl Viren generiert
    - Button-click-effect.ts: Klick auf den Haupt-Button
    - Critical-hit-effect.ts: Mit einer gewissen Wahrscheinlich ergibt ein Button-Klick mehr als einen Virus
    - replication-effect.ts: Während fünf Sekunden werden alle generierten Viren gezählt und anschliessend ein Vielfaches davon gutgeschrieben
    - reverse-engineering-effect.ts: Befällt einen anderen Spieler, während einer gewissen Zeitdauer zählen sämtliche seiner generierten Viren negativ
- Abstract-effect.ts: Marker-Klasse für einzelne Effekte
- Effect.gateway.ts: enthält WebSocket-Routen, um die verfügbaren Effekte auszulesen
- Effect.module.ts: Einstiegspunkt für das Effekt-Modul
- Effect.service.ts: Datenbankabfragen für Effekte
- Effect.util.ts: generische Logik für einzelne Effekte
- Effect-manager.ts: entdeckt alle Klassen, die von AbstractEffect erben und injected sie

Da die einzelnen Effekte aufeinander reagieren können müssen, wurde das Publish-Subscribe-Pattern gewählt. Die Abhängigkeiten der vier bisher umgesetzten Effekte sehen folgendermassen aus:



Auf der Datenbank gibt es folgende, neue Tabellen:

- effect: enthält alle verfügbaren Effekte
- effect_detail: ordnet jedem Effekt-Level den Preis und die Effizienz zu
- user_purchased_effect: hält fest, welcher User welchen Effekt und mit welchem Level gekauft hat
- user_active_effect: hält fest, welcher Effekt aktuell auf welchen User wirkt



Weitere Optimierungen

- Das Updaten der Punktzahlen im Frontend wurde im Server automatisiert. Zuvor wurden diese nach jeder Änderung manuell neu geschickt, dies führt nun ein Actor in regelmässigen Intervallen aus.
- Der Server schickt nun bei Anforderung der Rundendauer immer die aktuell noch verbleibende Zeit. Wenn der Client die Seite neu lädt, ist der Timer nun weiterhin mit dem Server synchronisiert, vorher hatte der Timer auf Seite Client wieder von vorn begonnen.
- Viel Logik für das Erstellen und Updaten von Entitäten wurde in die service-Klassen verschoben und dort mit QueryBuilder umgesetzt. Zuvor wurden oft Entity-Objekte erstellt, die Instanzvariablen befüllt und danach persistiert.

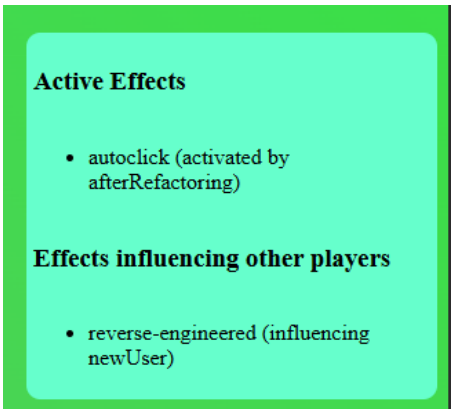
Zweiter Entwurf Frontend

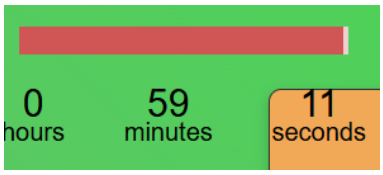
- Für das Frontend wurde ein einheitliches Design-Konzept erstellt
    - “Altmodisches” Design, welches an verdächtige Webseiten erinnern soll

- Das Frontend leitet bei Zugriffen ohne Login auf geschützte Seiten direkt auf den Login-Bildschirm weiter

- Das Frontend zeigt nun die verfügbaren Effekte in einem Shop-Bereich an und informiert den Server, wenn ein Effekt gekauft wurde

- Es gibt einen animierten Rundentimer, ein Leaderboard und einen Punkte-Counter und ein Effekt-Log.





- Gewisse Effekte haben eine Abklingzeit, diese werden mit einem Spinner markiert