Meilenstein 1 WebE Gruppe Luca & Dennis

Spielregeln und -ziel

Unser Spiel findet im Cyberraum statt und thematisiert die dort lauernden Gefahren. Das Spiel wird im Multiplayer-Modus gespielt, indem private Sessions mit anderen Spielern erstellt werden können. Im Gegensatz zur Realität geht es in unserem Spiel explizit darum, so viele Viren wie möglich zu sammeln, indem man wiederholt auf einen Button klickt. Mit den gesammelten Viren können auf dem Markt PowerUps gekauft werden, um die eigene Virenproduktion zu erhöhen oder Waffen, um andere Spieler zu sabotieren. Je nach Bewertungsmethode wird am Ende der Runde eine Rangliste der Spieler erstellt, beispielsweise wer am meisten Viren pro Sekunde produziert hat.

Requirements Client

Funktional:

- **MUSS**
    - Spieler müssen private Sessions erstellen können, der andere Spieler beitreten können
    - Spielergruppen müssen sich für die Spieldauer vor Beginn einer Runde entscheiden können
    - Spieler müssen per Klick auf einen Button Viren generieren können
    - Die Spieler müssen Informationen über den aktuellen Spielstand durch ein Leaderboard sowie Angaben zur eigenen Virenproduktion erlangen können
    - Spieler müssen Debuffs erwerben und temporär anderen Spielern schaden können.
    - Spieler müssen Buffs erwerben und temporär ihre Virenproduktion verstärken können.
    - Spieler müssen PowerUps erwerben und damit ihre Viren-Produktion dauerhaft verstärken können.
    - Spieler müüssen permanente Waffen erwerben und damit ihren Gegnern schaden können.
    - Spieler müssen sich gegenseitig mit einer Chat-Funktion austauschen können.

- **SOLL**
    - Das Spiel soll durchsetzen, dass Spieler über den Spielverlauf hinweg immer wieder gewisse Mindestwerte der Produktionsrate vorweisen müssen, um nicht eliminiert zu werden
    - Die Bewertungsmethode soll eingestellt werden können, beispielsweise wer die meisten Viren gesammelt hat.

Nicht funktional:

- **MUSS**
    - Accessibility: Das GUI muss so gestaltet sein, dass das Spiel auch mit visuellen Einschränkungen problemlos gespielt werden kann
    - Der Code muss an den zentralen Stellen dokumentiert sein, Kommentare müssen den Best Practices folgen. Da das Spiel Open-Source ist, muss sichergestellt sein, dass der Code für andere Entwickler leicht verständlich ist.
    - Der Client muss mit dem Server lose gekoppelt sein, Anwendungslogik muss im Client auf ein Minimum beschränkt sein.
- **SOLL**
    - Flüssige Darstellung: Das GUI soll sich mit min. 60 FPS aktualisieren, sofern möglich aufgrund der Hardware des Client-Rechners

Requirements Server

Funktional:

- **MUSS**
    - Clients müssen in Echtzeit miteinander synchronisiert werden, sodass Spieler gegeneinander antreten können
    - Der Server muss alle nötigen REST-API Schnittstellen und die Verarbeitungslogik für sämtliche Client-Funktionen bereitstellen
    - Der Server muss alle Spielaktivitäten auf der Datenbank persistieren, sodass jede Aktion im Spiel auch nach Spielende nachvollziehbar bleibt
    - Der Server muss eine Registrierungsfunktion für neue Spieler bzw. eine Login-Funktion für bestehende Spieler bereitstellen.
    - Der Server muss mehrere unabhängige Spielsessions verwalten. Insbesondere muss der Server sicherstellen, dass ein Spieler nicht auf mehrere Spielsessions gleichzeitig zugreifen kann.

Nicht funktional:

- **MUSS**
    - Nach jeder Aktion eines Spielers müssen alle Clients innerhalb von einer halben Sekunde synchronisiert sein.
    - Sicherstellung von Partitionstoleranz: Der Ausfall eines Clients darf die anderen Clients nicht negativ beeinflussen
    - Sicherheit: Der Server muss die rechtlichen Bedingungen für Datenschutz umsetzen. Zudem muss der Server konsequent sicherstellen, dass nur authentifizierte User auf die Serverfunktionalität zugreifen können
    - Die Datenintegrität muss jederzeit sichergestellt sein, um ein reibunglsoses Spielerlebnis zu garantieren und die Nachvollziehbarkeit getätigter Aktionen zu erhalten.
    - Der Code muss an den zentralen Stellen dokumentiert sein, Kommentare müssen den Best Practices folgen. Da das Spiel Open-Source ist, muss sichergestellt sein, dass der Code für andere Entwickler leicht verständlich ist.
    - Verfügbarkeit: 99.9% der Zeit muss der Server erreichbar sein

- **SOLL**
    - Möglichkeiten des Cheatings sollen so weit wie möglich verhindert werden

Protokoll

- Typ: JSON
- Übertragungsweg: Daten werden grundsätzlich über HTTPS via Websocket übertragen, Ressourcen wie Bilder werden aber per AJAX geladen
- Custom Header
    - X-Session-Key: verifiziert, dass der Request vom Client kommt und aus der richtigen Session. Sorgt auch dafür, dass die Requests nicht manuell mit curl oder vergleichbaren Methoden ausgeführt werden können

- JSON Schema Registrierung eines Users
    - Username: STRING
    - Passwort: STRING
- JSON Schema Registrierung für Teilnahme an einer Session
    - Zu registrierender Spieler: INT
    - Session Key: UUID
- JSON Schema Spielverlauf
    - Ausführender Spieler: INT
    - Betroffene Spieler: INT
    - Typ: (powerup, buff, weapon)
    - Level: INT

Verwendete Technologien

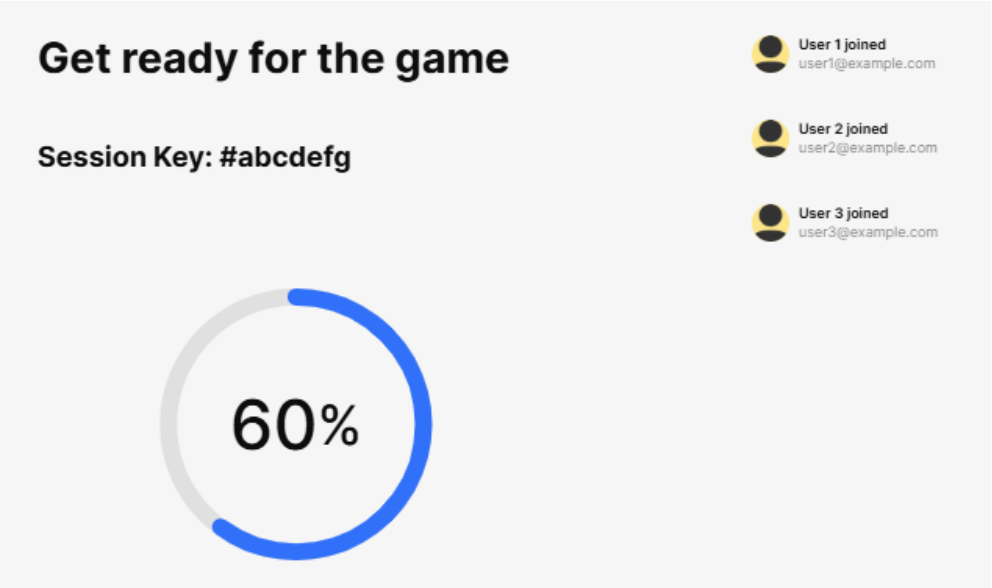
- Angular (Frontend)
- NestJS (Backend), Verwendung der Bibliotheken für Verschlüsselung und Authentifizierung
- PostgreSQL (Datenbank) mit PGAdmin als GUI-Tool
- Docker für das Deployment unserer Applikation

Arbeitsplan

- Bis PVA 2:
    - Anforderungen und Spielregeln definiert
    - Minimales NestJS- und Angular-Projekt aufgebaut
    - Erstes Konzept für Datenbankstruktur erstellt
    - Kick-Off durchgeführt
- Bis PVA 3:
    - Erste Version des Frontends umgesetzt
    - Alle Spielelemente final definiert (PowerUps, Waffen, …), jeweils mit einem Icon versehen. Es muss klargestellt sein, welche Elemente um wieviele Levels verbessert werden können und welchen Effekt sie haben.
    - Login-/Registrierungs-Funktionalität vollständig umgesetzt

- Bis PVA 4:
    - Registrierung für Spielsessions umgesetzt
    - Umsetzung sämtlicher Spielfunktionalitäten, die keine Synchronisierung zwischen Clients benötigen
    - Erste erfolgreiche Kommunikation zwischen Frontend und Backend über Websockets umgesetzt
    - Chat-Funktion umgesetzt
- Bis PVA 5:
    - Erste Funktionalitäten umgesetzt, die andere Spieler beeinflussen (Waffen etc.)
    - Leaderboard umgesetzt
    - Sicherstellung der nicht-funktionalen Anforderungen in der bisherigen Implementierung für Server und GUI
- Bis PVA 6:
    - Funktionalitäten, die andere Spieler beeinflussen, vollständig umgesetzt (Waffen etc.)
- Bis PVA 7:
    - Sicherstellung der Persistenz aller Spielschritte auf der Datenbank zur späteren Nachvollziehbarkeit
    - Überprüfung der Einhaltung aller funktionalen und nicht-funktionalen Anforderungen
    - In-Depth Testing und Beheben von Bugs
- Bis PVA 8:
    - Erfolgreiches Deployment des Projekts
    - In-Depth Testing und Beheben von Bugs
- Bis PVA 9:
    - In-Depth Testing und Beheben von Bugs
- Bis PVA 10:
    - Vorbereitung Präsentation und Demo

Mockup Session Joining



Mockup Spielscreen

Die folgenden Elemente sind hier dargestellt:

- Leaderboard (oben links): zeigt die ersten drei Plätze an
- Aktuelle Anzahl Viren (oben mitte)
- Chat (rechs)
- Marketplace (mitte): in Grün sind die Items dargestellt, für die das aktuelle Guthaben reicht.
- Button: Durch Klick auf den Button werden Viren generiert, dies ist anhand der grauen Plus-Zeichen visualisiert.