**Gambling-Webseite**

**Lars, Tilo, Atal, Alexander, Carl, Mina, Silas, Florian**

# Die Idee

Wir wollen als Gruppe eine funktionsfähige Glücksspielanwendung programmieren. Dabei soll es möglich sein, dass ein Benutzer einen Account erstellen und verifizieren kann. So soll er in der Lage sein, persönliche Informationen zu seinem Konto hinzuzufügen, darunter Name, Vorname, Geburtsdatum und Adresse. Zusätzlich soll ein Status gespeichert werden, ob dieser Benutzer verifiziert wurde. Das Geburtsdatum wird überprüft, um festzustellen, ob der Benutzer an der Glücksspielanwendung teilnehmen darf. Nur in Deutschland wohnende Menschen ab einem Alter von 18 Jahren dürfen teilnehmen.

Die Anwendung soll außerdem in der Lage sein, die Gewinne bzw. die Verluste eines Benutzers zu verwalten. Dazu hat der Benutzer in der Datenbank einen gespeicherten Kontostand, der speichert, wieviel Geld ihm zur Verfügung steht. Die Teilnahme an den Spielen bzw. die dafür notwendigen Transaktionen werden ebenfalls in der Datenbank in einer anderen Tabelle gespeichert, sodass wir unsere Gewinne und Verluste genau überprüfen können. Über eine grafische Oberfläche sollen zwei verschiedene Spiele, Blackjack und eine Slotmaschine, spielbar sein.

# Backend/Frontend

Über ein sicheres Backend sollen die Daten ohne Einfluss des Clients gespeichert werden. Das bedeutet, dass der Client sich mit dem Server verbindet und sich über seinen Account-Token verifiziert. Daraufhin setzt das Backend über den http-Header einen Cookie mit dem Token des Benutzerkontos auf dem Gerät. Dadurch wird gewährleistet, dass der Benutzer auch nach dem Neuladen oder Wechseln der Seite immernoch angemeldet ist.

# Slots/Blackjack - Spiele

## Idee

Das Slots-Spiel (auch bekannt als Spielautomat oder „Einarmiger Bandit“) soll auf einem festen Einsatz basieren, der vom Konto des Benutzers beim Spielstart in der Datenbank abgezogen wird. Dazu soll in der Tabelle Transaktionen eine neue Transaktion angelegt werden. Vorher soll überprüft werden, ob der Benutzer genügend Geld auf dem Konto hat. Der aktuelle Kontostand des Benutzers kann durch die Summe aller Transaktionen ermittelt werden. Daher ist es nicht notwendig, den Kontostand redundant in der **Kunden**-Tabelle zu speichern. Stattdessen kann der Kontostand jedes Mal aus der Summe der Transaktionen des jeweiligen Benutzers in der **Transaktionen**-Tabelle ausgelesen werden. Ansonsten könnte es zu Anomalien kommen, wenn der Kunden-Kontostand nicht mit der Transaktion rechtzeitig aktualisiert werden kann.

Das Gleiche gilt für das Blackjackspiel. Hier ist allerdings der Einsatz variabel und muss, wie eben beschrieben, mit der Datenbank abgeglichen werden. Danach wird im Backend separat vom Client die gesamte Spiel-Logik berechnet. Dazu gehören unter anderem normale Spielfunktionen wie Karten ziehen, Split oder Double-Down. Dabei muss jeder Schritt im Spiel zwischen Backend und Frontend synchronisiert werden. Dazu kommt noch eine graphische Oberfläche, die es dem Benutzer einfach macht, die oben genannten Aktionen auszuführen.

Zu Beginn des Spiels öffnet der Benutzer die entsprechende Frontend-Seite. Über einen Cookie, der auf dem Client-Browser gesetzt wird, lässt sich erkennen, welchen Benutzer-Account der Benutzer verwendet. Es wird jedoch nicht direkt die Benutzer-ID im Cookie gespeichert, da diese manipuliert werden könnte. Stattdessen wird ein Token gespeichert, der als Referenz zum Nutzer-Account dient. Nach dem Spiel wird dieses Token durch die Benutzer-ID aufgelöst, sofern der Benutzer angemeldet ist. Anschließend wird eine neue Speicherungstransaktion (?) eingefügt. Die Kommunikation zwischen dem Spiel-Frontend und dem Skript im Backend erfolgt über einzelne HTTP-Anfragen. Wir haben uns jedoch gegen die Verwendung von WebSockets entschieden, da sie relativ viel Bandbreite und Serverleistung benötigen. Hier muss die TCP-Verbindung durchgehend aufrechterhalten werden. Bei den einzelnen HTTP-Anfragen wird die Authentifizierung durch den Benutzer-Token sichergestellt. Die Blackjack-Anwendung verwendet den Port 8080, während die Slots-Anwendung stattdessen auf den Port 8081 zurückgreift. Wir benötigen zwei verschiedene Endpunkt-URLs, über die die wesentliche Kommunikation mit dem Benutzer läuft:

/balance: Dieser Endpunkt wird vom Frontend genutzt, um den aktuellen Kontostand des Benutzers abzufragen. Hierbei erfolgt die Authentifizierung über das Benutzer-Token. Diese Abfrage erstellt einen zufälligen Wert für jedes benötigte Rad der Slot-Maschine.

/play: Dieser Endpunkt dient dazu, eine neue Transaktion zu erstellen, z. B. wenn der Benutzer seinen Einsatz für das Slot-Spiel tätigt.

Auf das Anfordern der TCP-Anfragen wird das Backend dann mit dem Ergebnis gespeichert. So wird gleichzeitig die entsprechende Transaktion in den Datenbank-Tabellen-Transaktionen. In den http Headern wird ein Cookie gesetzt, sodass die Benutzeranmeldung erhalten bleibt.