**Gambling App**

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

# Die Idee

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

Die App soll außerdem in der Lage sein die Gewinne Bzw. Die Verluste eines Benutzers verwalten zu können. Dazu hat der Benutzer in der Datenbank einen gespeicherten Kontostand, der anzeigt wie viel Geld ihm zu Verfügung steht. Die Teilnahme an den Spielen soll als Transaktionen gespeichert werden, sodass wir unsere Gewinne und Verluste genau sehen können. Über eine grafische Oberfläche sollen zwei verschiedene spiele visualisiert werden und spielbar sein.

# Backend/Frontend

Über ein sicheres Backend sollen die Daten vom Client unabhängig gespeichert werden. Das heißt der Client verbindet sich mit dem Server und authentifiziert sich über seinen Account Token. Darauf setzt das Backend über den http Header einen Cookie mit dem Token des Benutzerkontos auf dem Gerät selbst nach dem Navigieren durch die Seite, einem erneuten Öffnen oder einem Laden.

# Slots/Blackjack - Spiele

## Idee

Das Slot Spiel 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 natürlich genau wie eben beschrieben mit der Datenbank abgeglichen werden. Danach wird damit im Backend separat von Client die gesamte Spiel-Logik berechnet, mit einigen den normalen Funktionen wie weitere Karten zu nehmen oder auch den Stapel zu splitten und einen Double-Down zu machen. Dabei muss jeder Schritt im Spiel zwischen Backend und Frontend synchronisiert sein. Dazu kommt noch eine graphische Oberfläche, die es dem Benutzer einfach macht die eben genannten doch komplizierten Aktionen auszführen.

Zu Beginn des Spiels öffnet der Benutzer die Frontend-Seite des Spiels. Über einen Cookie, das auf dem Client-Browser gesetzt wird, lässt sich erkennen, welchem Benutzer-Account der Benutzer zugeordnet ist. 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 der Spiel-Frontend und dem Skript im Backend erfolgt über einzelne HTTP-Anfragen. Wie 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. 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 Request 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.