Större inlämningsuppgift-Whack-a-mole

(FEU 22, Mana Pecar)

Github: https://github.com/Manpec/Whack-a-mole

Sammanfattning:
Syftet med uppgiften är att skapa ett digitalt spel, Whack-a-mole med Angular. Applikationen innehåller en modul, två komponenter (Board-komponenten, LeaderBoard-komponenten), en router och en service. Applikationen kopplas till Firebase som är en app-utvecklingsplattform. Detta möjliggör att spara data till databasen och hämta datan från databasen och rendera i mallen.

Arkitektur:
Board-componenten motsvarar huvudsidan i whack-a-mole-spelet. I html:n visas först en input ruta där användaren skriver sitt namn som sparas senare för att skickas till databasen. När man trycker "submit"-knappen renderas spelbrädan, timern och poängräknaren. Tid som kvarstår och poäng visas på skärmen under spelets gång. Och efter spelet slutar visas poängen och snabbaste mole slaget (reactionTime) man fick samt knappar för att spela igen eller visa topplistan. Det som också händer är att ett objekt med namn, score och reactionTime skickas till Firebase.
För att rendera olika utseendet av sidan implementerade jag <ng-container><ng-template> och villkorlig rendering med *ngIf directive.

I ngOninit()-metoden i Board-komponenten skapas en observer för poängräkning och sedan skapas 25 instanser av Mole-klassen med en for-loop och stoppar in varje instans i “holes”-arrayen. Observer objektet i form av en subscriber till vår skapade observer skickas som ett argument till Mole-konstruktorn. Detta gör att Mole-instanserna kan meddela observern i board-komponenten när de blir träffade om if-satsen är true i smacked()-metoden i mole-klassen. Om if-satsen är true meddelar subscribern till observablen för att uppdatera poängen när molen träffas.

startGame()-metoden startar spelet genom att ställa in ett intervall för att visa och dölja moles slumpmässigt samt minskar den återstående tiden. stopGame()-metoden stoppar spelet genom att rensa intervallet och dölja alla moles samt kör fastestReactionTime()-metoden som beräknar den snabbaste reaktionstiden för spelaren. I saveScore()-metoden sparas användarens score med hjälp av Leaderboard-Servicen som är ansvarig för att kommunicera med databasen.

smack()-metoden är länkad till ett click event i html templaten och den anropar smacked()-metoden för Mole-klassen för att ändra tillståndet till "smacked".
Metoden moleStateChangeCssClass() är länkad till [ngClass]-direktivet i html:n, och den ändrar klassen för ett div-element till ett av tre olika status baserat på status parametern som skickas till

den. Metoden använder en switch-sats för att bestämma vilken klass som ska returneras baserat på värdet på statusen. Klasserna definieras i CSS-filen och används för att kontrollera stilen för div-elementet.

Leaderboard-komponenten visar de 10 spelarna med högst poäng samt spelaren med den snabbaste reaktionstiden. Komponenten använder Router och LeaderBoard-Service med dependency-injection. Router används för att navigera till huvudsidan och Service används för att hämta spelardata från en databas. Metoderna getPlayers() och getFastestReaction() anropas och deras returnerade observables prenumereras med ngOnInit() för att få top 10 spelardata och den snabbaste reaktionstiden.

Leaderboard-servicen är ansvarig för kommunikation med databasen och det finns tre metoder i den. postPlayerScore()-metoden används för att lägga till en ny spelares poäng till collection. getFastestReaction()-metoden används för att hämta spelaren med den snabbaste reaktionstiden från collection genom att använda en query mot databasen. Den sorterar collection efter "fastestReaction"-fältet i stigande ordning och begränsar resultatet till den första i den sorterade listan. getPlayers()-metoden används för att hämta de 10 bästa spelarna från collection genom att sortera collection efter "score"-fältet i fallande ordning och begränsar resultatet till de 10 bästa spelarna.