

Euristrike

Realisatiedocument

Wout Jacobs
Student Bachelor in de Toegepaste Informatica – Applicatieontwikkeling

Inhoudsopgave

1. INLEIDING	5
2. ANALYSE	6
2.1. Project information	6
2.2. Probleemstelling tijdens eerste editie van het spel	6
2.2.1. Doel van de stageopdracht	6
2.3. Game Logica	7
2.3.1. Game mode 1	7
2.3.2. Game Mode 2	7
2.3.3. Free for All	7
2.4. Front-end	7
2.4.1. Vue	7
2.4.2. React	8
2.4.3. Eind Conclusie	8
2.5. Back-end	8
2.6. .NET	8
2.6.1. Conclusie:	9
2.7. Modellen	9
2.7.1. Data model	9
2.7.2. Use Case diagram	10
2.8. PWA	11
3. REALISATIE	13
3.1. Pipelines	13
3.2. Basisstijl	14
3.3. Navigatie	15
3.4. Header	15
3.5. Login	17
3.5.1. Loginpagina en authenticatie	17
3.5.2. Gebruik van MSAL in Login.tsx	17
3.5.3. Gebruik van React Context in AuthProvider	17
3.6. Spel aanmaken	18
3.6.1. Navigatie naar het spel aanmaken	18
3.6.2. Het formulier voor spelconfiguratie	18
3.6.3. Validatie	18
3.6.4. Aanmaak en navigatie	19
3.6.5. Backend: Wat er achter de schermen gebeurt	19
3.6.6. Target bepalen	20
3.7. Game selecteren	20
3.7.1. Component en relevante games	20
3.8. Game info	21
3.8.1. Algemene weergave	21
3.8.2. Dode gebruiker	22

3.8.3. Afhandeling van game die beëindigd is	22
3.8.4. Foutafhandeling en statusbeheer	22
3.9. LeaderBoard	22
3.9.1. Front-end	23
3.9.2. backend	23
3.9.3. Game Statistieken: Gedetailleerde Informatie van Spelers	23
3.9.4. Samenvatting	24
3.10. User stats	24
3.10.1. Functionaliteit en dataverwerking	25
3.10.2. UI-componenten en opbouw	25
3.10.3. Backendstructuur: User Entity en API	25
3.10.4. Ontwerp en synchronisatie	26
3.11. Qr code laten scannen	26
3.11.1. Generatie van de QR-code	26
3.11.2. Scan-knop en interactie	27
3.12. Qr code scannen	27
3.12.1. Scan proces	27
3.12.2. Weergave en styling van de camera-feed	28
3.12.3. Backendverwerking en spelmechaniek	28
3.13. Game owner management	29
3.13.1. Functionaliteit in de Front-end	29
3.13.2. Admin Home Pagina	29
3.13.3. Backend Verwerking	30
3.13.4. Samenvatting	30
3.14. Rules	31
3.15. Manage Admin	31
3.15.1. Functionaliteit van de toggle-switch	31
3.15.2. Ontwerp en synchronisatie	32
3.15.3. Backend: Wijziging van de Adminstatus	32
3.16. Ranked game	33
3.16.1. Spelverloop en Toewijzing van Targets	33
3.16.2. Automatische Elo-berekening	33
3.16.3. Backend: Verwerking van Eliminaties	33
3.16.4. Gebruikersinterface	34
3.16.5. Elo-berekening in Ranked Mode	34
3.16.6. Unit Test elo	35
3.17. Free for all	35
3.17.1. Spelverloop zonder toegewezen targets	35
3.17.2. Eliminatieregels en scoreverwerking	35
3.17.3. Backendverwerking	36
3.17.4. Gebruikerservaring	36
3.18. Qr-code pdf	37
3.18.1. Front-end-integratie: een centrale downloadknop	37
3.18.2. Backendlogica: QR-code generatie en PDF-samenstelling	37
3.18.3. API-endpoint: veilige en dynamische bestandsnaam	37
3.18.4. Onderhoudbaarheid en uitbreidbaarheid	38

4. BESLUIT	39
LITERATUURLIJST	40
LIJST VAN FIGUREN	42
AI PROMPTS	43

1. Inleiding

Dit document biedt een uitgebreide beschrijving van de realisatie van de Gotcha Applicatie, die nu de nieuwe naam **EuriStrike** heeft voor een aantrekkelijkere naam. Het project is ontwikkeld voor Euricom's jaarlijkse DEV-Cruise, een teambuilding evenement van vier dagen in het buitenland waar medewerkers elkaar beter leren kennen en lezingen geven over interessante onderwerpen. Het richt zich op het verbeteren van de teamdynamiek en samenwerking tijdens dit evenement. In de volgende secties wordt gedetailleerd ingegaan op de verschillende aspecten van de applicatie, van de initiële analyse en probleemstelling tot de uiteindelijke implementatie en evaluatie. We bespreken de gebruikte technologieën, de samenwerking tussen de verschillende teams, en de uitdagingen die we zijn tegengekomen en hebben overwonnen. Dit document dient als een volledige gids voor de ontwikkeling en realisatie van de EuriStrike Applicatie, en biedt inzicht in zowel de technische als organisatorische aspecten van het project.

Euricom levert technologische diensten zoals cloudoplossingen, softwareontwikkeling en consultancy aan een breed scala van sectoren. Ze combineren technische expertise met praktische bedrijfskennis en staan bekend om hun open bedrijfscultuur waarin samenwerking en kennisdeling centraal staan. Met dit project wilden ze op een speelse manier hun waarden rond innovatie, community en samenwerking versterken.

De ontwikkeling van dit project gebeurde niet alleen, maar in samenwerking met een student van een andere hogeschool, wat bijdroeg aan een waardevolle kruisbestuiving van kennis, inzichten en vaardigheden. Dit onderstreept het multidisciplinaire en samenwerkende karakter van het project.

De EuriStrike Applicatie is ontworpen met het oog op gebruiksvriendelijkheid en efficiëntie, waarbij de nadruk ligt op het bevorderen van interactie en betrokkenheid onder de deelnemers. Door middel van innovatieve functies en een intuïtieve interface, stelt de applicatie gebruikers in staat om op een speelse en competitieve manier met elkaar te communiceren. Dit draagt niet alleen bij aan een betere teamgeest, maar ook aan een verhoogde productiviteit en tevredenheid onder de deelnemers.

Daarnaast wordt in dit document aandacht besteed aan de feedback van de gebruikers en hoe deze input heeft bijgedragen aan de verdere verfijning en verbetering van de applicatie. We zullen ook enkele succesverhalen en casestudies presenteren die de impact van de Applicatie op de teamdynamiek en samenwerking illustreren. Tot slot biedt dit document aanbevelingen voor toekomstige ontwikkelingen en mogelijke uitbreidingen van de applicatie, met als doel om de gebruikerservaring nog verder te optimaliseren.

Betrokken hoofdstukken:

- Analyse: In dit hoofdstuk wordt de initiële analyse van de Gotcha Applicatie besproken, inclusief de belangrijkste problemen en uitdagingen die tijdens de eerste editie van het spel naar voren kwamen
- Realisatie: Hier wordt de volledige realisatie van de applicatie beschreven, van de initiële analyse en probleemstelling tot de uiteindelijke implementatie en evaluatie.
- Feedback en Verbeteringen: Dit hoofdstuk richt zich op de feedback van de gebruikers en hoe deze input heeft bijgedragen aan de verdere verfijning en verbetering van de applicatie.
- Succesverhalen en Casestudies: Hier worden enkele succesverhalen en casestudies gepresenteerd die de impact van de Gotcha Applicatie op de teamdynamiek en samenwerking illustreren.
- Aanbevelingen voor Toekomstige Ontwikkelingen: Tot slot biedt dit hoofdstuk aanbevelingen voor toekomstige ontwikkelingen en mogelijke uitbreidingen van de applicatie, met als doel om de gebruikerservaring nog verder te optimaliseren.

2. Analyse

De analyse van de Gotcha Applicatie richt zich op het identificeren van de belangrijkste problemen en uitdagingen die tijdens de eerste editie van het spel naar voren kwamen. Deze omvatten zelftargeting en een kleinere targetpool, evenals vroege eliminaties die de betrokkenheid van sommige deelnemers verminderten. Het doel van deze analyse is om oplossingen te vinden die het spel eerlijker, spannender en inclusiever maken.

Een diepgaande analyse van het project werd eerder uitgevoerd door het volledige team. Dit document bevat onder meer de probleemstelling, minimum requirements, QR-code implementatie en risicoanalyse. De volledige analyse is hier te raadplegen:

 <https://portfoliowoutjacobs.netlify.app/pdf-viewer-analyse.html>

2.1. Project information

Deze stageopdracht is gebaseerd op een spel dat tijdens de jaarlijkse DEV-Cruise van 2024 bij Euricom werd gespeeld. Het spel, dat veel enthousiasme oproep onder de deelnemers, draaide om het elimineren van collega's met een Nerf-geweertje. Elke deelnemer ontving een mysterieuze enveloppe met de naam van een target die ze gedurende het weekend moesten uitschakelen door hem of haar te raken met een Nerf-pijltje. Om het spel eerlijk en overzichtelijk te houden, waren er enkele basisregels:

- Er mocht geen andere Euricommer binnen een straal van 3 meter zijn.
- Eliminaties waren niet toegestaan tijdens de sessies.
- Werd je uitgeschakeld, dan gaf je jouw target door aan degene die je had uitgeschakeld.

Hoewel het spel een groot succes was, kwamen er enkele verbeterpunten naar voren. Deze vormen de basis voor deze stageopdracht, waarin het doel is om een verbeterde versie van het spel te ontwikkelen.

2.2. Probleemstelling tijdens eerste editie van het spel

Tijdens de eerste editie van het spel kwamen twee belangrijke knelpunten naar voren:

1. Zelftargeting en beperkte targetpool
Doordat de targets willekeurig werden toegewezen, werd de pool van beschikbare spelers steeds kleiner.
Dit leidde ertoe dat sommige deelnemers uiteindelijk zichzelf als target kregen, ondanks dat er nog andere actieve spelers waren.
2. Vroege eliminatie
Sommige deelnemers werden al in een zeer vroeg stadium uitgeschakeld – soms zelfs voordat ze wisten wie hun target was. Hierdoor konden zij nauwelijks deelnemen aan het spel en misten ze de volledige spelervaring.

2.2.1. Doel van de stageopdracht

Het doel van deze stageopdracht is om deze en andere verbeterpunten aan te pakken door een vernieuwde en verbeterde versie van het spel te ontwikkelen. Gedurende de stageperiode zal ik, in samenwerking met mijn begeleiders, werken aan oplossingen die het spel eerlijker, spannender en inclusiever maken.

2.3. Game Logica

In deze sectie zal ik de verschillende spelmodi beschrijven en uitleggen wat deze precies inhouden:

2.3.1. Game mode 1

Spelers worden in een cirkelvorm aan een target gekoppeld met een maximale tijdslimiet voor het spel. Bij het aanmaken van een game wordt een begin- en eindtijd/datum ingevoerd, wat vooraf plannen mogelijk maakt. De admin kan de game op elk moment stoppen. Targets worden willekeurig toegewezen en bij eliminatie krijgt een speler de target van de geëlimineerde persoon. Om te voorkomen dat een speler zichzelf als target krijgt, worden alle gebruikers in een unieke array opgenomen en willekeurig verdeeld met de Fisher-Yates-Shuffle-methode

2.3.2. Game Mode 2

Het Elo-systeem bepaalt de sterkte van elke speler en past zich dynamisch aan op basis van hun prestaties. Het Elo-systeem is een methode om de relatieve sterkte van spelers in competitieve spellen te bepalen.

De Elo-score is een dynamisch beoordelingssysteem dat binnen de Euristrike Ranked Mode wordt gebruikt om spelers te rangschikken op basis van hun prestaties. Het systeem werd oorspronkelijk ontwikkeld voor schaken, maar is hier aangepast voor gebruik in een real-time eliminatiemode.

Door gebruik te maken van een dynamische array worden spelers met een hogere rating vaker als target gekozen, wat zorgt voor een extra uitdaging. Om te voorkomen dat het spel eindeloos doorgaat, biedt deze game mode een flexibele tijdslimiet en heeft de admin de mogelijkheid om de game op elk moment te stoppen. Het scoresysteem beloont spelers voor eliminaties en straf verliezen, waarbij de aanwezigheid in de array wordt aangepast bij elk veelvoud van zes punten. Uiteindelijk wordt de winnaar bepaald op basis van de Elo-rating en de behaalde prestaties aan het einde van het spel. Deze geavanceerde, Elo-gedreven Gotcha mode biedt een spannende, eerlijke en strategische spelervaring die zich voortdurend aanpast aan de vaardigheden van de spelers.

2.3.3. Free for All

In de "Free for All" spelmodus speelt elke deelnemer voor zichzelf, zonder teams of bondgenoten. Het is pure chaos, waarbij iedereen iedereen mag afschieten. Het doel is om zoveel mogelijk tegenstanders uit te schakelen binnen de gestelde tijdslimiet. Deze modus bevordert een intense en competitieve sfeer, waarbij strategisch denken en snelle reflexen essentieel zijn voor succes. De speler met de meeste eliminaties aan het einde van de ronde wordt uitgeroepen tot winnaar.

2.4. Front-end

We vergelijken Vue en React, twee populaire front-end frameworks, om de meest geschikte keuze te maken voor onze gebruikersinterface. We bekijken hun kenmerken, voordelen en nadelen in het kader van ons project.

2.4.1. Vue

Vue.js is een progressief JavaScript-framework met een modulaire, component-gedreven opzet. Het is eenvoudig te leren en maakt gebruik van declaratieve HTML-sjablonen en een reactief datamodel. Officiële tools zoals Vue Router en Vuex ondersteunen de ontwikkeling van gestructureerde applicaties. Dankzij het lichte gewicht is Vue geschikt voor zowel kleine als middelgrote projecten.

Voordelen:

- Eenvoudig te leren met duidelijke documentatie.
- Geschikt voor zowel kleine integraties als SPA's.
- Lichtgewicht en snel.

Nadelen:

- Minder ondersteuning voor grote enterprise-applicaties.
- Kleiner ecosysteem dan React

2.4.2. React

React, ontwikkeld door Meta, is een JavaScript-bibliotheek gericht op het bouwen van dynamische interfaces via componenten en het Virtual DOM. Dankzij JSX kunnen HTML-structuren in JavaScript worden geschreven. State management is flexibel via hooks of externe bibliotheken zoals Redux. React Router ondersteunt client-side routing.

Voordelen:

- Grote community en veel tooling.
- Flexibel en schaalbaar, ideaal voor complexe applicaties.
- Uitbreidbaar met frameworks zoals Next.js.

Nadelen:

- Steilere leercurve door advanced concepten.

2.4.3. Eind Conclusie

Voor dit project kiezen we React vanwege de schaalbaarheid, flexibiliteit en het brede ecosysteem. React wordt ook vaker gebruikt in de industrie, wat zorgt voor meer beschikbare kennis, tools en ondersteuning. Dit maakt het uitermate geschikt voor de complexe eisen van onze applicatie, zoals live updates, meerdere spelmodi en toekomstige uitbreidingen zoals notificaties en custom games.

2.5. Back-end

Voor de backend van deze applicatie hebben we gekozen voor **.NET**. .NET is een krachtig en veelzijdig framework ontwikkeld door Microsoft, ideaal voor het bouwen van schaalbare en veilige webapplicaties. Er zijn verschillende redenen waarom we voor .NET hebben gekozen voor dit project:

2.6. .NET

.NET integreert naadloos met Microsoft-tools zoals Entra ID en Azure-diensten, wat binnen de Euricom-omgeving essentieel is voor zaken als Single Sign-On en infrastructuurkoppelingen. Het biedt hoge prestaties en betrouwbaarheid, wat belangrijk is voor live updates van bijvoorbeeld het leaderboard en game-statistieken.

Beveiliging is een kernpunt: .NET ondersteunt moderne standaarden voor veilige verbindingen, authenticatie en autorisatie, wat essentieel is gezien de verwerking van gevoelige bedrijfsdata. Dankzij C# kunnen we de game-logica en administratie efficiënt en gestructureerd ontwikkelen.

Tot slot biedt .NET uitgebreide documentatie, sterke community-ondersteuning en een rijke set aan tools, waardoor we snel en toekomstbestendig kunnen bouwen.

2.6.1. Conclusie:

.NET vormt een stabiele, veilige en schaalbare basis die perfect aansluit bij de technische omgeving van Euricom en de eisen van onze applicatie

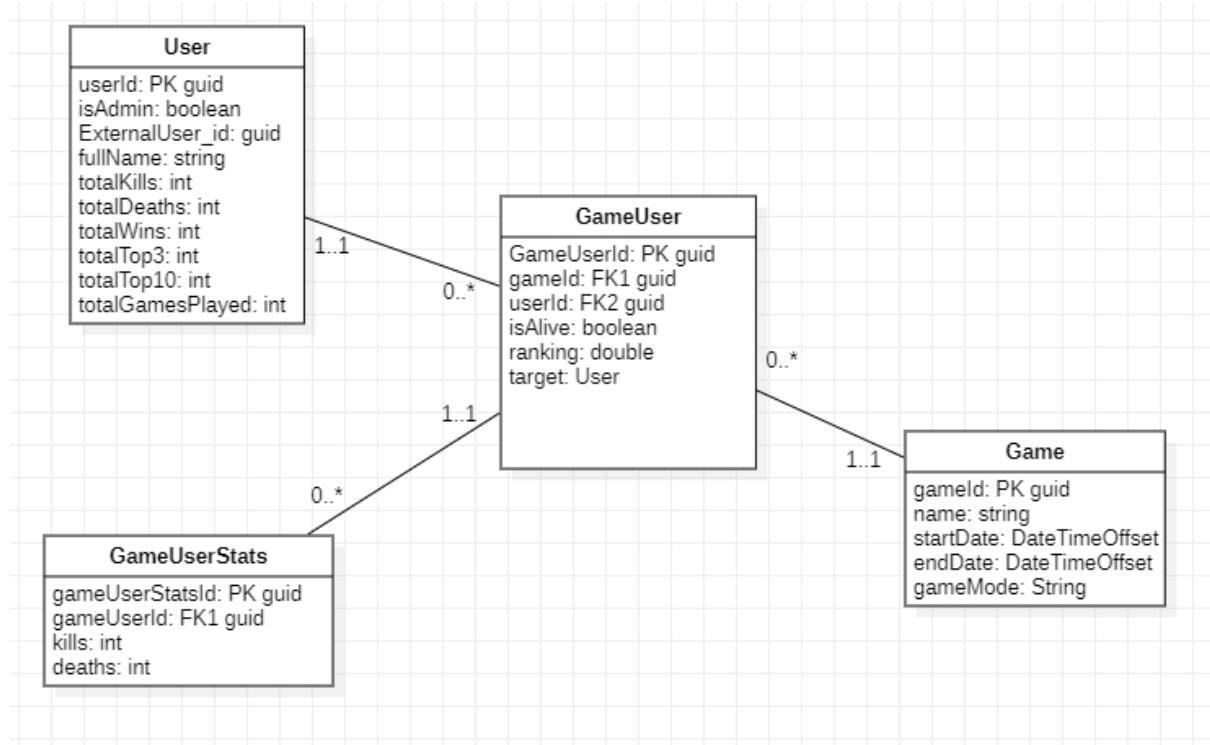
2.7. Modellen

Hier beschrijf ik de modellen die ik gemaakt heb.

2.7.1. Data model

Relaties tussen entiteiten

- User ↔ GameUser
Eén gebruiker kan deelnemen aan meerdere games, maar elke GameUser hoort bij exact één gebruiker.
(Relatie: $1 \leftrightarrow 0..*$)
- Game ↔ GameUser
Eén game bevat meerdere deelnames, waarbij elke deelname bij één specifieke game hoort.
(Relatie: $1 \leftrightarrow 0..*$)
- GameUser ↔ GameUserStats
Elke deelname kan één of meerdere bijbehorende statistiekrecords hebben, die elk uniek gekoppeld zijn aan één deelname.
(Relatie: $1 \leftrightarrow 0..*$)
- GameUser.target ↔ User
Elke GameUser kan een andere gebruiker als doelwit toegewezen krijgen. Dit maakt het mogelijk om het spelmechanisme van bijvoorbeeld Gotcha te ondersteunen.



Figuur 1: data model

2.7.2. Use Case diagram

Dit use case diagram zie *Figuur 2* toont de interacties tussen verschillende typen gebruikers (actoren) en de functionaliteiten van de Gotcha-app. De app wordt gebruikt voor het organiseren en spelen van het spel "Gotcha", waarbij spelers elkaar proberen te elimineren door QR-codes te scannen.

Hieronder ziet me de actoren

1. User (Gebruiker)

De standaardspeler van het spel. Deze gebruiker heeft toegang tot basisfunctionaliteiten zoals inloggen, QR-codes scannen en het aanmaken van spellen.

2. GameOwner

Een gevorderde gebruiker met extra rechten. Deze gebruiker kan eigen spellen beheren en spelers handmatig elimineren.

3. Admin

De gebruiker met de hoogste autoriteit in de applicatie. Deze rol heeft de bevoegdheid om gebruikers te beheren.

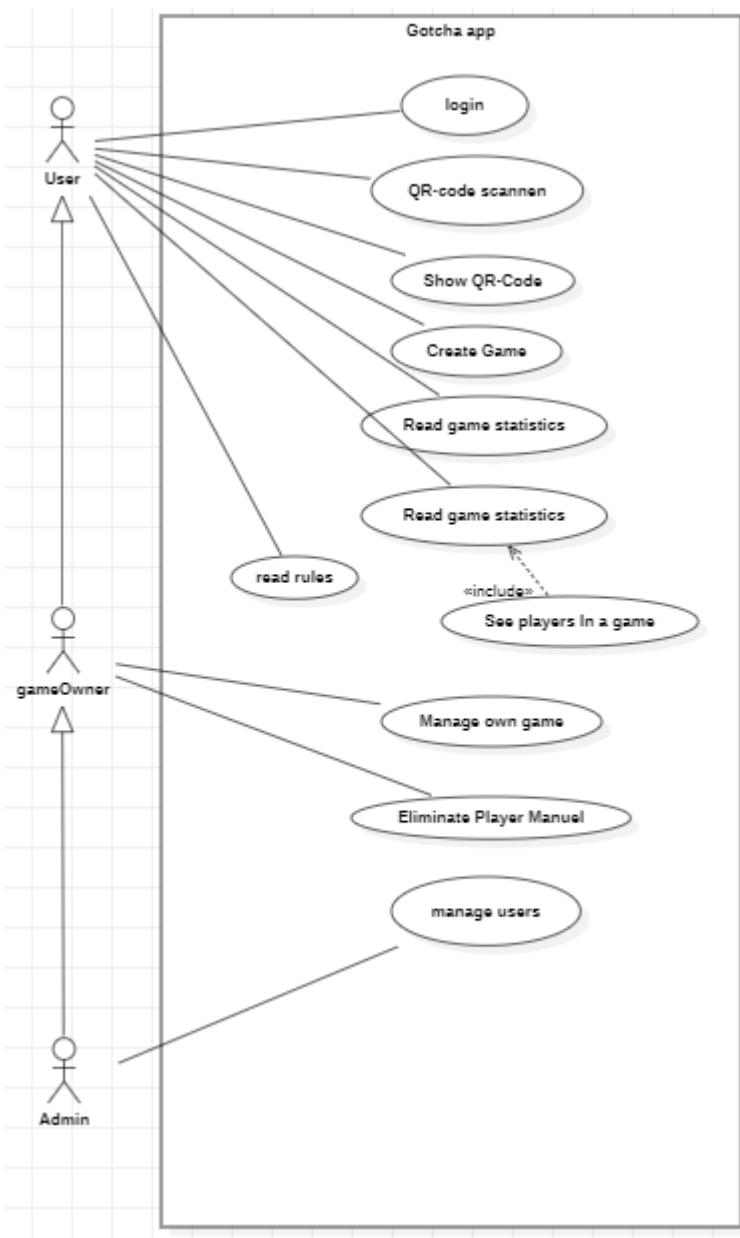
Let op: de hiërarchie tussen de actoren wordt weergegeven met pijlen. Een admin is ook een user, en een superAdmin is ook een admin.

Hieronder worden de use cases toegelicht:

- **Login:** Elke gebruiker moet kunnen inloggen op de app.
- **QR-code scannen:** Gebruikers kunnen QR-codes scannen van andere spelers om hen te elimineren.
- **Show QR-code:** Gebruikers kunnen hun eigen QR-code tonen zodat anderen hen kunnen scannen.
- **Create Game:** Gebruikers kunnen zelf een nieuw spel aanmaken.
- **Read game statistics:** Gebruikers kunnen statistieken bekijken van het spel, zoals aantal overgebleven spelers.
- **See players in a game:** Deze functionaliteit is onderdeel van het lezen van statistieken (aangegeven met «include»). Gebruikers kunnen zien wie er allemaal meedoelen aan een spel.
- **Read rules:** Alle gebruikers kunnen de spelregels raadplegen.

Extra functionaliteiten voor hogere rollen:

- **Manage own game (Admin):** Admins kunnen spellen die zij hebben aangemaakt beheren.
- **Eliminate Player Manual (Admin):** Admins kunnen handmatig een speler elimineren uit een spel.
- **Manage users (SuperAdmin):** SuperAdmins hebben toegang tot het gebruikersbeheer binnen de app.



Figuur 2: use case diagram

2.8. PWA

Een PWA is een webapplicatie die werkt als een native app, maar toegankelijk is via een browser. Gebruikers kunnen de app installeren op hun startscherm, offline gebruiken en notificaties ontvangen, zonder dat een download via de App Store nodig is.

- Direct toegankelijk via een URL, geen installatie via appstores nodig.
- Offline functionaliteit via caching en service workers.
- Snelle updates zonder handmatige installatie.
- Eén codebase voor alle platforms (iOS, Android, desktop).
- Tijd- en kostenbesparend.

Met tools zoals Create React App is PWA-ondersteuning eenvoudig te integreren. React gebruikt een service worker om resources te cachen, zodat de app offline blijft werken.

Spelers hebben snel toegang nodig tot hun QR-code en notificaties. Een PWA biedt hiervoor de juiste snelheid, toegankelijkheid en gebruiksgemak, ook bij tijdelijke verbindingsproblemen.

3. Realisatie

De volledige applicatie is gemaakt met de bedoeling dat alles eenvoudig aanpasbaar is en dat we zaken kunnen aanpassen en toevoegen met zo weinig mogelijk code. Dit gecombineerd met verschillende schermformaten, een dynamische en responsieve layout te garanderen.

We zijn gestart met het maken van een basisstijl samen met een loginfunctie. Aan de hand van deze start hebben we verder gebouwd richting het eindresultaat zodat er één rode lijn door het project loopt.

3.1. Pipelines

De **Build en Deploy pipeline** is opgezet met het oog op betrouwbaarheid, duidelijkheid en een vlotte uitrol naar de ontwikkelomgeving. De volledige CI/CD-structuur is ondergebracht in één pipelinebestand, waarbij een duidelijke scheiding gemaakt wordt tussen de verschillende fases: het bouwen van de front-end en back-end, gevolgd door het deployen ervan. Deze structuur maakt het mogelijk om op een gecontroleerde en uitbreidbare manier wijzigingen automatisch door te voeren naar Azure zodra deze in de `main` branch terechtkomen.

- Een front-end buildfase waarin de React-app wordt opgebouwd,
- Een backend buildfase waarin de .NET-oplossing gecompileerd en voorbereid wordt,
- Een deployfase voor de front-end naar Azure Static Web Apps,
- Een deployfase voor de backend naar Azure App Service.

In de front-end buildfase wordt gebruikgemaakt van Node.js en npm. De build start met het installeren en cachen van de node dependencies, wat zorgt voor snellere builds bij volgende runs. Daarna wordt met `npm run build` een productieklare versie van de React-app gegenereerd. Deze build wordt als artifact opgeslagen zodat die later opnieuw kan worden opgehaald tijdens de deployfase. Dankzij deze aanpak blijft de buildfase efficiënt en voorspelbaar.

In de backend buildfase wordt de .NET SDK geïnstalleerd en worden NuGet packages hersteld en gecachet. De solution wordt opgebouwd in Release-configuratie, en verpakt in een `.zip`-bestand dat klaar is voor deployment. Eventuele Entity Framework Core migraties worden automatisch uitgevoerd zodat de databank steeds gesynchroniseerd blijft met de meest recente applicatiestructuur. Ook dit backend-artifact wordt bewaard voor gebruik in de deployfase.

De front-end deployfase gebruikt het eerder gegenereerde artifact om de React-app te uploaden naar Azure Static Web Apps. Aangezien de build reeds in een eerdere fase is uitgevoerd, hoeft deze stap enkel het artifact te downloaden en te publiceren. Dit versnelt de pipeline en zorgt voor een consistente deployment.

De backend deployfase zorgt voor het uitrollen van de .NET backend naar een App Service omgeving op Azure. Dit gebeurt met behulp van een vooraf gedefinieerde service connection. Door gebruik te maken van een apart deployment slot wordt het risico bij live-updates beperkt. Het volledige `.zip`-pakket van de backend wordt automatisch gedeployed naar de juiste omgeving.

Dankzij de duidelijke opbouw en het gebruik van artifacts is de pipeline goed schaalbaar. Extra validaties, teststappen of deploy-omgevingen kunnen eenvoudig worden toegevoegd zonder de bestaande structuur te doorbreken. Deze aanpak maakt de pipeline onderhoudsvriendelijk en klaar voor toekomstige uitbreidingen zoals staging-omgevingen of automatische rollbacks.

3.2. Basissstijl

De applicatie maakt gebruik van een centrale layoutcomponent die als basis dient voor alle schermen zoals u kan zien in *Figuur 3*. Deze BaseLayout zorgt voor een uniforme structuur en gebruikservaring, en omvat drie hoofdonderdelen: de header, de contentzone, en – indien nodig – een onderste navigatiebalk.

Het doel van deze layout is om een consistente visuele opbouw te garanderen over alle pagina's heen, ongeacht de inhoud. De layout is bovendien zo opgebouwd dat nieuwe pagina's eenvoudig toegevoegd kunnen worden, zonder telkens terugkerende elementen opnieuw te moeten implementeren. Dit draagt bij aan zowel de onderhoudbaarheid als uitbreidbaarheid van de applicatie.

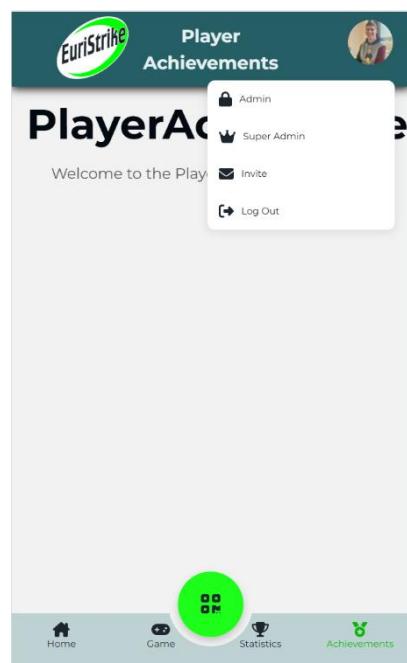
De header is steeds zichtbaar bovenaan het scherm. Op pagina's die niet rechtstreeks via de navigatie bereikbaar zijn, verschijnt een terugknop naast het logo. Hiermee kan de gebruiker intuïtief terugnavigeren naar de vorige pagina. Centraal in de header wordt de titel van de huidige pagina weergegeven. Helemaal rechts bevindt zich de profielfoto van de ingelogde gebruiker. Bij het klikken op deze foto opent een menu met gebruikersspecifieke opties.

Het logo zelf is aanklikbaar en brengt de gebruiker terug naar de homepage. Afhankelijk van de breedte van het scherm past het logo zich automatisch aan in formaat en positie, zodat de layout ook op mobiele toestellen correct weergegeven wordt.

De layout is voorzien van volgende stijlkenmerken:

- Sticky header: blijft altijd bovenaan zichtbaar, met een vaste hoogte en een subtile slagschaduw.
- Responsief ontwerp: elementen zoals het logo en de headercomponenten passen zich aan volgens de schermgrootte.
- Centrale paginatitel: zorgt voor duidelijke context op elk scherm.
- Profielmenu met submenu: biedt snel toegang tot gebruikersacties.
- Centrale contentzone: alle pagina-inhoud wordt binnen dit gebied geladen.
- Optionele navigatiebalk onderaan: enkel zichtbaar op schermen waar dit relevant is.

Deze opbouw zorgt ervoor dat gebruikers op een herkenbare en intuïtieve manier door de applicatie kunnen navigeren. De layout is daarnaast flexibel opgebouwd zodat aanpassingen aan de header of footer niet manueel per pagina moeten gebeuren.



Figuur 3: basissstijl

3.3. Navigatie

De navigatie in de applicatie is opgebouwd met het oog op gebruiksgemak, uitbreidbaarheid en visuele duidelijkheid. Deze ziet u in *Figuur 4* en *Figuur 5*. Ze is ondergebracht in één centrale component, waarbij de volledige navigatiestructuur afkomstig is uit één configuratiebestand. Dit maakt het mogelijk om op een snelle en efficiënte manier nieuwe navigatie-items toe te voegen of bestaande te wijzigen zonder de logica van de component zelf te moeten aanpassen. Deze aanpak verhoogt de onderhoudbaarheid van de code en beperkt foutgevoeligheid bij toekomstige uitbreidingen.

De navigatiebalk is onderaan het scherm gepositioneerd en bestaat uit drie zones:

- Een linkerzijde met een reeks menu-items,
- Een centrale, visueel opvallende knop,
- Een rechterzijde met nog een reeks menu-items.

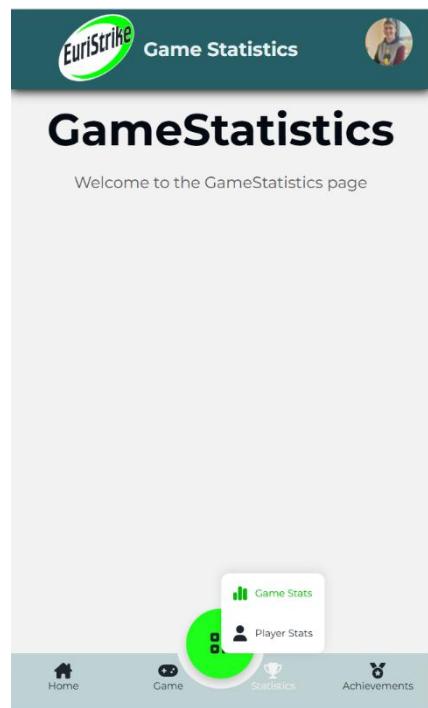
Elk item in de navigatie is voorzien van een icoon en een korte tekstuele label. Indien van toepassing kunnen menu-items ook een submenu bevatten dat opent wanneer het hoofditem geselecteerd wordt.

De actieve pagina wordt duidelijk aangeduid: het bijhorende menu-item in de navigatiebalk kleurt groen. Dit geldt ook voor submenu-items; wanneer een gebruiker zich op een subpagina bevindt, wordt zowel het hoofdmenu-item als het submenu-item visueel gemarkeerd. De actieve status helpt gebruikers te oriënteren en geeft direct aan waar ze zich bevinden binnen de applicatie.

Wanneer een submenu geopend is, worden het hoofditem en het icoon in een witte kleur weergegeven. Dit contrast helpt om geopende menu's visueel te onderscheiden van gesloten menu's en draagt bij aan een overzichtelijke gebruikersinterface.

Opvallend is de zwevende knop die centraal boven de navigatiebalk gepositioneerd is. Deze knop leidt naar een belangrijke functie in de applicatie, namelijk het starten van een spel via een QR-code. Door de plaatsing en vormgeving – een ronde knop met een opvallende groene kleur – krijgt deze actie extra visuele nadruk. Dit zorgt ervoor dat gebruikers deze functie snel kunnen terugvinden, wat essentieel is voor de gebruikservaring.

Dankzij de modulaire opbouw en het gebruik van een configuratiebestand, kan de navigatie eenvoudig aangepast worden aan toekomstige behoeften. Nieuwe submenu's, iconen of routes kunnen via het centrale bestand toegevoegd worden zonder de navigatiecomponent zelf te moeten herschrijven. Dit principe van scheiding tussen configuratie en logica bevordert de schaalbaarheid van de applicatie.



Figuur 4: navigatie

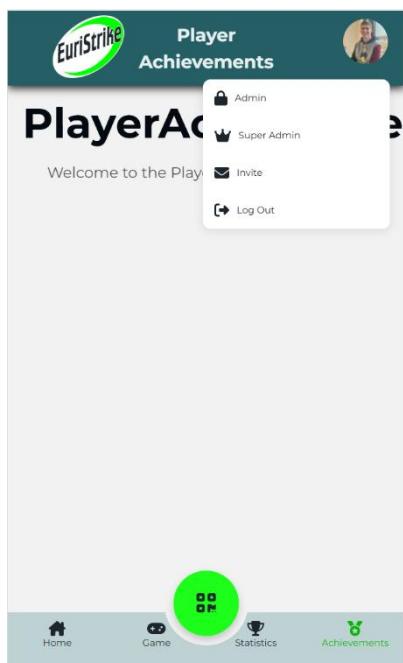
3.4. Header

Naast de onderste navigatiebalk bevat de applicatie ook een compact hoofdmenu, dat toegankelijk is via een icoon rechtsboven in het scherm. Dit menu biedt gebruikers toegang tot diverse beheertaken, afhankelijk van hun rol binnen het systeem. Het menu verschijnt als een dropdown wanneer op het gebruikersicoon wordt geklikt, waarbij het icoon ook vervangen kan worden door een profielfoto indien beschikbaar.

Het hoofdmenu bevat de volgende opties:

- Admin: Geeft toegang tot de beheerpagina's van het geselecteerde spel. Indien er een spel geselecteerd is, wordt dit spel automatisch meegegeven in de URL zodat de beheeromgeving direct in de juiste context wordt geopend.
- Super Admin: Leidt naar een overzicht van de beheermogelijkheden voor gebruikers met verhoogde rechten.
- Invite: Biedt toegang tot een uitnodigfunctie waarmee gebruikers nieuwe spelers of beheerders kunnen toevoegen.
- Log Out: Meldt de gebruiker af en keert terug naar het inlogscherm.

Deze structuur is flexibel opgebouwd, zodat het menu dynamisch aangepast kan worden op basis van de rol van de gebruiker of de context (zoals een geselecteerd spel). Ook hier wordt gebruikgemaakt van een configuratie-array, wat toekomstige aanpassingen of uitbreidingen vereenvoudigt.



Figuur 5: header

3.5. Login

De Gotcha-applicatie maakt gebruik van Microsoft Entra ID voor veilige toegang. Dit betekent dat gebruikers zich kunnen inloggen met hun bestaande Euricom Microsoft-account, zonder dat ze een aparte set inloggegevens hoeven te onthouden. De authenticatie wordt afgehandeld via de Microsoft Authentication Library (MSAL), die zorgt voor een vlotte en veilige loginervaring.

3.5.1. Loginpagina en authenticatie

Bij het openen van de loginpagina ziet de gebruiker een knop met het label "Inloggen met Microsoft" zoals in *Figuur 6*. Wanneer de gebruiker op deze knop klikt, wordt hij doorgestuurd naar de officiële Microsoft-loginpagina. Nadat de gebruiker succesvol is ingelogd, wordt hij automatisch teruggeleid naar de applicatie.



Figuur 6: login pagina

3.5.2. Gebruik van MSAL in Login.tsx

In de Login.tsx-component wordt de MSAL hook `useMsal` gebruikt om de authenticatie te beheren. Na een succesvolle login controleert de applicatie of er een actieve gebruiker is. Als dat het geval is, wordt de gebruiker doorgestuurd naar de homepage van de applicatie.

De configuratie van de Microsoft Authentication Library bevindt zich in AuthConfig.ts. Dit bestand zorgt ervoor dat de applicatie zowel in de lokale ontwikkelomgeving als in de productieomgeving goed functioneert. De configuratie bepaalt welke rechten de applicatie nodig heeft, zodat deze toegang krijgt tot de juiste bronnen.

De belangrijkste instellingen in deze configuratie zijn:

- clientId: de unieke identificatie van de applicatie in Azure Active Directory,
- authority: de URL die de applicatie naar Microsoft's login-systeem leidt,
- redirectUri: de locatie waar de gebruiker na succesvolle login terugkomt.

3.5.3. Gebruik van React Context in AuthProvider

De AuthProvider maakt gebruik van React Context om gebruikersinformatie beschikbaar te stellen aan de hele applicatie. Dit zorgt ervoor dat andere delen van de app snel kunnen controleren of de gebruiker ingelogd is en de benodigde toegang heeft. De volgende gegevens worden gedeeld:

- externalUserId: de unieke ID van de gebruiker,
- isAuthenticated: een indicator of de gebruiker ingelogd is,
- acquireToken: een functie waarmee de applicatie een toegangstoken kan verkrijgen voor toegang tot beschermd gegevens.

Het toegangstoken wordt normaal gesproken opgehaald met de functie `acquireTokenSilent()`. Als dat niet lukt, wordt de gebruiker automatisch doorgestuurd om opnieuw in te loggen.

Aan de serverzijde wordt de authenticatie beheerd door het AuthController-endpoint. Dit endpoint is beveiligd met autorisatie, wat betekent dat alleen ingelogde gebruikers toegang hebben. Wanneer een gebruiker zich aanmeldt:

- Het toegangstoken wordt gecontroleerd om te bevestigen dat het geldig is.
- De server haalt de gebruikersinformatie (bijvoorbeeld het externalUserId) uit het token.

- De server controleert of de gebruiker al bestaat in de database. Als dat niet het geval is, wordt een nieuw gebruikersaccount aangemaakt.
- De server stuurt een bericht terug naar de front-end met de gebruikers-ID, zodat de app weet wie is ingelogd.

Deze opzet zorgt ervoor dat de backend perfect geïntegreerd is met Microsoft Entra ID en dat gebruikers veilig en eenvoudig toegang hebben tot de applicatie, zonder dat ze hun wachtwoorden of andere inloggegevens hoeven in te voeren.

3.6. Spel aanmaken

Het proces om een nieuw spel aan te maken in de applicatie is ontworpen om intuïtief, gebruiksvriendelijk en efficiënt te verlopen zie *Figuur 7*. Zowel de gebruikersinterface als de achterliggende logica in de backend werken samen om ervoor te zorgen dat het spel correct geconfigureerd en opgeslagen wordt.

3.6.1. Navigatie naar het spel aanmaken

Op de homepage van de applicatie bevindt zich rechtsonder een opvallende zwevende actieknop met een plusteken ('+') vormgegeven in het groen. Deze knop is bedoeld om direct de aandacht te trekken van de gebruiker en dient als toegangspoort tot het aanmaken van een nieuw spel.

3.6.2. Het formulier voor spelconfiguratie

Wanneer de gebruiker op de knop klikt, wordt hij doorgestuurd naar een formulierpagina. Hier vult hij stap voor stap de noodzakelijke gegevens in om het spel te configureren:

1. Naam van het spel – vrije tekstinvoer.
2. Speltype – keuze uit beschikbare opties (zoals ‘Standaard’ of ‘Ranked’).
3. Datum van het spel – via een datumkiezer, eventueel met een datumbereik.
4. Start- en eindtijd – alleen indien het spel niet de hele dag duurt. Anders worden automatisch standaardtijden ingesteld: start om 00:00 en einde om 23:59. Zie *Figuur 8*
5. Spelers uitnodigen – selectie van gebruikers via een multiselect component. Enkel geauthenticeerde gebruikers kunnen als deelnemer toegevoegd worden.

De spelerselectie gebeurt meestal uit een lijst van eerder geregistreerde externe gebruikers (bijvoorbeeld medewerkers of stagiairs), die worden opgehaald van de backend.

3.6.3. Validatie

Het formulier maakt gebruik van Yup-validatieschema's die tijdens het invullen live de ingevoerde gegevens controleren. Indien er fouten zijn, worden die onmiddellijk weergegeven bij het bijbehorende veld. Bij het indienen van het formulier worden alle velden opnieuw gecontroleerd om ervoor te zorgen dat het formulier volledig en correct is.

Figuur 7: spel aanmaken

3.6.4. Aanmaak en navigatie

Wanneer alle gegevens correct zijn ingevuld en de gebruiker op de 'Create Game' knop drukt:

- Er wordt een API-call verstuurd naar de backend via een POST /api/Game verzoek.
- De front-end bepaalt na een succesvolle response of de gebruiker een deelnemer of enkel de spelbeheerder is:
 - Als de gebruiker ook een deelnemer is, wordt hij doorverwezen naar zijn persoonlijke spelpagina met informatie over zijn target.
 - Als de gebruiker enkel beheerder is, wordt hij doorgestuurd naar een beheerpagina van het spel waar hij een overzicht heeft van alle deelnemers, voortgang en instellingen.

3.6.5. Backend: Wat er achter de schermen gebeurt

De backend handelt het verzoek tot het aanmaken van een spel af via een CreateGame handler, die een reeks stappen doorloopt om alles correct op te zetten:

De eerste stap is het aanmaken van een nieuw Game object met de naam, speltype en start- en eindtijd. Dit object wordt opgeslagen in de database.

Voor elke speler die werd uitgenodigd:

- Wordt gecontroleerd of deze gebruiker al bestaat.
- Als dat niet het geval is, wordt hij aangemaakt via een aparte GetOrCreateUser handler.
- Voor elke gebruiker wordt een GameUser aangemaakt en gekoppeld aan het spel.

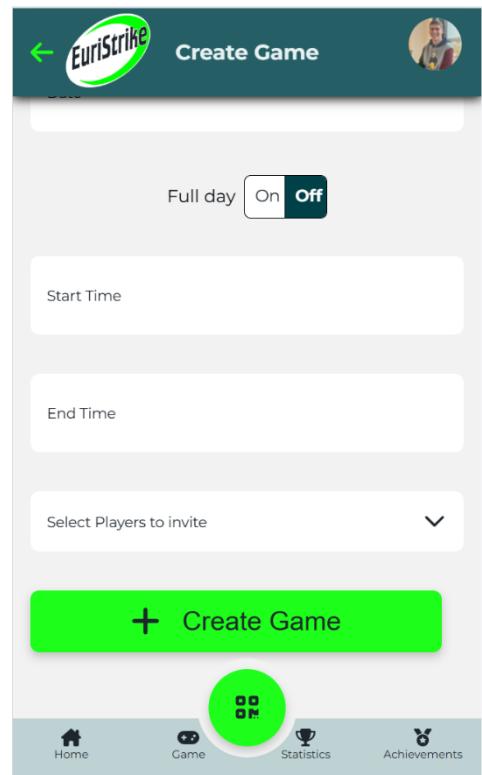
Indien de spelbeheerder zelf geen deelnemer is, wordt hij alsnog aan het spel gekoppeld als beheerder, zonder actieve deelname (bijvoorbeeld met IsAlive = false). Dit zorgt ervoor dat hij toch het spel kan beheren via de beheerinterface.

Een belangrijk onderdeel van het spel is het toewijzen van een target aan elke speler. Hiervoor wordt een Fisher-Yates shuffle toegepast op de lijst van GameUsers:

- De lijst van spelers wordt eerst in willekeurige volgorde geshuffeld.
- Daarna krijgt elke speler als target de volgende speler in de lijst (de laatste speler krijgt de eerste als target).
- Dit garandeert dat niemand zichzelf als target krijgt en dat het spel eerlijk start.

Als het speltype Ranked is, wordt voor elke speler een standaard rankingwaarde (bijv. 100) ingesteld.

Tot slot wordt voor elke deelnemende gebruiker het totaal aantal gespeelde spellen met 1 verhoogd.



Figuur 8: spel aanmaken met tijd

3.6.6. Target bepalen

Zoals in de analyse besproken gaan we aan de hand van de fisher yates shuffle bepalen welke speler het target is van de andere speler. Deze shuffle krijgt een lijst binnen van alle deelnemers aan het spel gooit deze is een virtuele hoed en geeft een nieuwe lijst terug waar alle deelnemers bepaald zijn en zo een random target krijgen toegewezen. Als men een target elimineert krijgt hij het target van zijn slachtoffer. *Figuur 9* is de shuffle

```
3 references
public static void FisherYatesShuffle<T>(List<T> list)
{
    int n = list.Count;
    for (int i = n - 1; i > 0; i--)
    {
        int j = _random.Next(0, i + 1);
        (list[i], list[j]) = (list[j], list[i]);
    }
}
```

Figuur 9: fisher yates shuffle

3.7. Game selecteren

In de Euristrike-applicatie heeft elke game een specifieke status die wordt bepaald aan de hand van de start- en einddatum van het spel. De status kan één van de volgende waarden zijn:

- Upcoming: Het spel wordt in de toekomst gespeeld.
- Active: Het spel wordt momenteel gespeeld.
- Inactive: Het spel is afgesloten.

De status van een game wordt dynamisch bepaald door het vergelijken van de huidige tijd met de start en einddatum van het spel. De GameDetailCard-component bevat een logica die ervoor zorgt dat de juiste status wordt weergegeven afhankelijk van de huidige datum en tijd. De status van de game kan als volgt worden berekend:

1. Upcoming: Wanneer de huidige tijd vóór de starttijd van het spel ligt.
2. Active: Wanneer de huidige tijd zich binnen het tijdsbestek van de start- en eindtijd bevindt.
3. Inactive: Wanneer het spel al voorbij is of het is handmatig afgesloten (bijvoorbeeld via de isGameEnded prop).

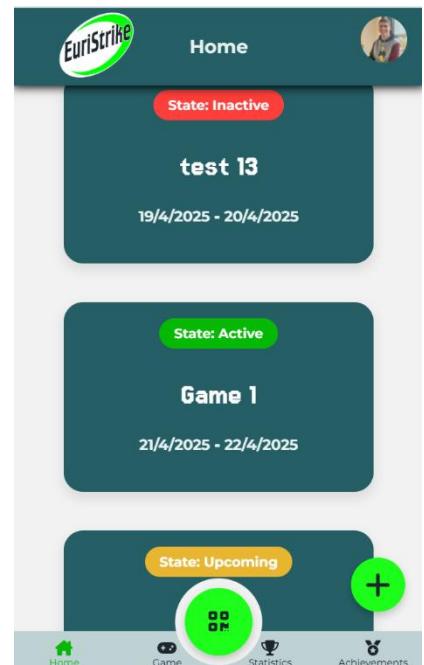
3.7.1. Component en relevante games

Alle games die aan de gebruiker worden getoond, zijn specifiek voor de gebruiker en worden opgehaald via een gebruikerscontext. Dit betekent dat elke gebruiker alleen de games ziet die voor hen relevant zijn, gebaseerd op hun profiel en de gegevens die aan hun account zijn gekoppeld zoals in *Figuur 10* al mijn persoonlijke games kan zien.

Wanneer een gebruiker op een game klikt, worden de volgende gegevens getoond:

- Titel van het spel: Dit helpt de gebruiker te weten welk spel hij of zij bekijkt.
- Target: Het doel van het spel, zoals de te doden vijanden of te voltooien opdrachten.
- Rules: De regels van het spel worden weergegeven.
- Leaderboard: Het huidige leaderboard, zodat de gebruiker de voortgang van de deelnemers kan zien.

De GameDetailCard-component is verantwoordelijk voor het weergeven van de game-informatie en het verwerken van gebruikersinteracties. Het is opgebouwd uit verschillende belangrijke onderdelen:



Figuur 10: home pagina

- Navigatie: Wanneer de gebruiker op de card klikt, wordt de gebruiker doorgestuurd naar een gedetailleerdere pagina van de game.
- Statusweergave: De status van de game wordt weergegeven met behulp van de classNames-functie, die de juiste CSS-stijl toepast op basis van de status (bijv. actieve, inactieve, of aankomende games).
- Datumformatting: De start- en einddatum van de game worden geformateerd om een overzichtelijke weergave te bieden van wanneer de game plaatsvindt.
- Game Mode: Er wordt aangegeven of het spel in ranked of standaard modus wordt gespeeld, met een bijbehorend icoon om dit te verduidelijken.

De handleClick-functie stuurt de gebruiker naar de specifieke gamepagina zodra er op de card wordt geklikt. Dit gebeurt door de game-ID op te slaan in de context via setSelectedGame en de gebruiker vervolgens naar de gedetailleerde gamepagina te navigeren met de navigate functie.

3.8. Game info

In dit onderdeel van de applicatie krijgt de gebruiker een gedetailleerd overzicht van de geselecteerde game zoals in *Figuur 11*. Bij het laden van het component worden meerdere gegevens opgehaald en weergegeven die afhankelijk zijn van de status van het spel en de rol van de gebruiker binnen deze game.

3.8.1. Algemene weergave

Bovenaan het scherm wordt steeds de naam van het spel getoond, gevolgd door het type game modus waarin de gebruiker zich bevindt. Er wordt hier onderscheid gemaakt tussen een Standard game en een Ranked game. Indien het om een Ranked game gaat, wordt er ook een icoon (RankingIcon) naast de spelmodus weergegeven om visuele duidelijkheid te bieden.

Na het ophalen van de gamegegevens en de bijhorende spelinformatie van de ingelogde gebruiker (via zijn of haar externalUserId), wordt ook de bijhorende target opgehaald indien deze beschikbaar is. Deze target wordt visueel weergegeven in een apart TargetComponent. Daarin zien we:

- De naam van de target
- Een eventuele profielfoto van de target
- De titel "Target:" bovenaan het component

Deze informatie is enkel zichtbaar indien de gebruiker nog in leven is binnen het spel (isAlive = true) en de game nog niet beëindigd is.

Onder de target-weergave bevinden zich verschillende knoppen waarmee de gebruiker kan navigeren naar andere delen van de applicatie:

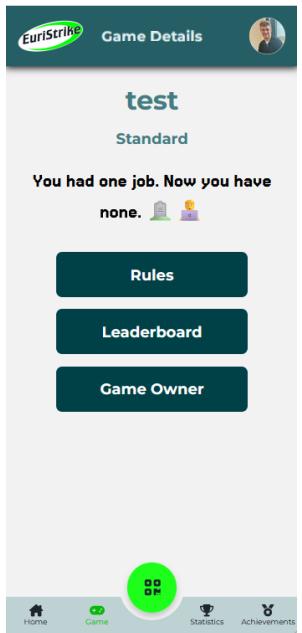
- Rules: leidt de gebruiker naar een pagina met de spelregels.
- Leaderboard: toont het algemene klassement van de game.
- Game Owner: is enkel zichtbaar voor gebruikers die als eigenaar van het spel aangeduid zijn (isGameOwner = true). Deze knop geeft toegang tot de beheerdersinterface van de game.



Figuur 11: geselecteerd spel

3.8.2. Dode gebruiker

Indien de speler niet meer in leven is (bijvoorbeeld door uitgeschakeld te worden door een andere speler), wordt de target niet meer getoond. In plaats daarvan verschijnt er een willekeurige ‘dead message’ die getrokken wordt uit een vooropgestelde lijst van teksten (`DeadMessages`) zoals te zien in *Figuur 12*. De gebruiker krijgt dan nog enkel de optie om het leaderboard te bekijken, aangezien verdere interactie in het spel voor deze gebruiker niet meer mogelijk is.



3.8.3. Afhandeling van game die beëindigd is

Wanneer het spel is afgelopen — wat automatisch gedetecteerd wordt op basis van de einddatum (`endDate`) of het feit dat een speler al een eerste plaats in het klassement heeft (`ranking === 1`) — dan verandert het component in een samenvattend eindscherm. Hierin wordt het volgende getoond:

- Een overzicht van de top 3 spelers in het spel, gepresenteerd via het `Top3Players` component.
- Belangrijke statistieken van de game, afhankelijk van het type game (Ranked of Standard). Enkele mogelijke statistieken zijn:
 - Most Kills: de speler die de meeste uitschakelingen heeft behaald
 - Most Deaths: enkel voor Ranked games – de speler met de meeste keren dat hij/zij is uitgeschakeld
 - Best KD (Kill/Death-ratio): enkel voor Ranked games – de speler met de hoogste verhouding kills tegenover deaths

Voor elke statistiek wordt er getoond:

- De naam van de statistiek
- De profielfoto van de bijhorende speler (indien beschikbaar)
- De bijhorende waarde (bijvoorbeeld aantal kills of KD-ratio)

Deze gegevens worden dynamisch opgehaald via een aparte API call naar `getAllGameUsersStats()` van de game. De component verwerkt deze data tot een aparte `SelectGameUserStat` structuur die apart in de interface weergegeven wordt.

Figuur 12: doods bericht

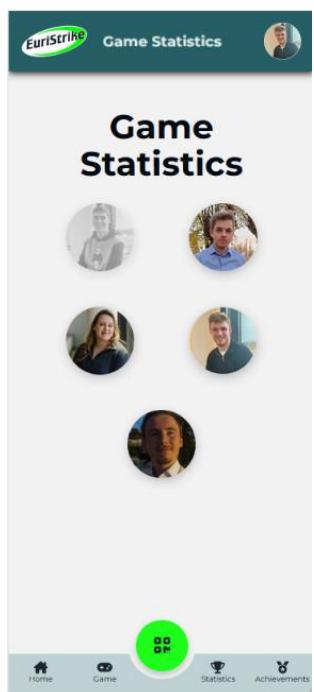
3.8.4. Foutafhandeling en statusbeheer

Tijdens de verschillende stappen van dataverwerking wordt de laadstatus (`loading`) en eventuele fouten (`error`) zorgvuldig bijgehouden:

- Tijdens het laden van de gegevens wordt een korte boodschap weergegeven: “`data loading...`”
- Bij fouten wordt er een `toast` gegenereerd via de `showToast()` hook, waarin de foutmelding zichtbaar wordt voor de gebruiker. De meldingen zijn niet automatisch sluitbaar en worden bovenaan het scherm weergegeven voor maximale zichtbaarheid.

3.9. LeaderBoard

De LeaderBoard-pagina biedt een visueel overzicht van alle spelers die deelnemen aan het geselecteerde spel, ongeacht hun status (levend of dood) zoals in *Figuur 13*. Deze pagina laat niet alleen het verloop van het spel zien, maar creëert ook een competitieve sfeer door statistieken op een toegankelijke manier zichtbaar te maken.



Figuur 13: spelers overzicht

3.9.1. Front-end

Alle spelers worden weergegeven als kaarten in een overzichtelijk grid-layout (.game-users-grid). Elke spelerkaart toont de profielfoto van de externe gebruiker, en de status wordt visueel aangeduid:

- Levende spelers worden in kleur getoond.
- Uitgeschakelde spelers krijgen automatisch een grayscale effect en een verlaagde opacity dankzij dynamische CSS-klassen (grayscale opacity-50), waardoor snel te zien is wie nog actief is.

De weergave van elke speler wordt opgebouwd op basis van twee objecten:

- gameUserStats: bevat statistieken zoals aantal kills per speler.
- externalUsers: bevat visuele en persoonlijke informatie zoals naam en profielfoto.

Wanneer een speler op een kaart klikt (onClick), wordt de bijbehorende speler geselecteerd en verschijnt er een popup-venster (modal) dankzij de Dialog-component van PrimeReact zoals te zien in *Figuur 14*. Deze modal toont:

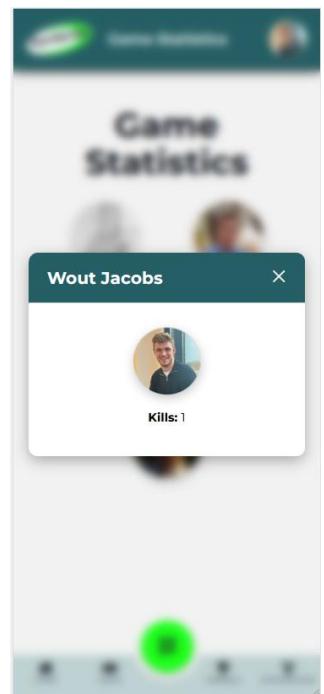
- De volledige naam van de geselecteerde speler als header.
- De profielfoto in een gestileerde avatarweergave.
- Het aantal kills (eliminaties) van de speler.

Dankzij useState en useRef wordt intern bijgehouden welke speler is geselecteerd (selectedStats) en of het modaal getoond moet worden (showModal). Door gebruik te maken van cardRefs zou in de toekomst zelfs smooth scrollen of focus mogelijk zijn bij interactie.

De opbouw van componenten volgt het principe van herbruikbaarheid:

- PlayerOverView.tsx: verwerkt data en toont de grid + modal.
- GameUserCard.tsx: verzorgt de individuele spelerkaart.

Deze modulaire aanpak zorgt ervoor dat het leaderboard eenvoudig uit te breiden is met bijvoorbeeld ranking-indicatoren, filteropties, of sortering.



Figuur 14: overzicht van een speler

De data van de spelers wordt beheerd en opgehaald via een API. Deze API zorgt ervoor dat de laatste gegevens van de spelers worden bijgewerkt, zoals het aantal kills, deaths en of de speler levend is.

Wanneer de LeaderBoard-pagina wordt geladen, wordt er een verzoek gestuurd naar de backend om alle statistieken van de spelers op te halen voor het geselecteerde spel. De backend haalt dan alle gegevens van de spelers uit de database en stuurt deze terug naar de pagina.

Deze informatie wordt vervolgens gebruikt om het LeaderBoard op te bouwen en te tonen, met inbegrip van:

- Het aantal kills van elke speler.
- Het aantal deaths van elke speler.
- Of de speler levend is of niet.

3.9.3. Game Statistieken: Gedetailleerde Informatie van Spelers

Op de **GameStatistieken** pagina kunnen spelers gedetailleerde informatie zien over hun prestaties in het spel. Dit kan bijvoorbeeld het aantal **kills** zijn (hoeveel andere spelers ze hebben geëlimineerd) of het aantal **deaths** (hoe vaak ze zelf zijn geëlimineerd).

De statistieken worden ook weergegeven in een overzichtelijke **tabel**. Dit helpt spelers om te zien hoe ze zich verhouden tot anderen in het spel, door een soort **ranglijst** te tonen van de beste spelers op basis van hun prestaties.

Deze statistieken worden continu bijgewerkt om de voortgang van de game in real-time te reflecteren.

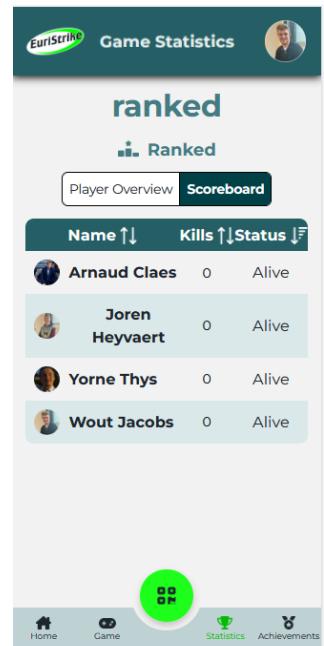
3.9.4. Samenvatting

1. Front-end (wat de speler ziet):
 - Het Leaderboard toont alle spelers als kaarten met hun profielfoto en status (levend of dood).
 - Spelers kunnen op een kaart klikken om meer gedetailleerde statistieken te zien, zoals hun kills.
 - Er is een Scoreboard dat een ranglijst toont van de beste spelers op basis van hun prestaties zoals in *Figuur 15*.
2. Backend (de serverzijde):
 - De backend zorgt ervoor dat de laatste statistieken van elke speler worden opgehaald en naar de LeaderBoard-pagina worden gestuurd.
 - De backend berekent de kills en deaths van elke speler en zorgt ervoor dat de juiste gegevens worden weergegeven op het Leaderboard.
3. Game Statistieken:
 - Spelers kunnen gedetailleerde statistieken zien over hun prestaties, zoals het aantal kills en deaths.
 - De statistieken worden in real-time bijgewerkt en zorgen voor een dynamisch overzicht van de voortgang in het spel.

Op deze manier krijgt elke speler een duidelijk, visueel overzicht van zijn of haar prestaties en kan iedereen zien wie de beste prestaties heeft in het spel!

3.10. User stats

Op de Player Statistics-pagina worden gedetailleerde spelstatistieken van een externe gebruiker weergegeven. Zie *Figuur 16*. Deze gegevens worden opgehaald op basis van het externe gebruikers-ID, dat beschikbaar is via de authenticatiecontext (`useAuth`). De statistieken geven inzicht in de prestaties van de speler, zoals kills, deaths, overwinningen en algemene plaatsingen in wedstrijden. De pagina biedt visuele ondersteuning via een interactieve staafdiagram (BarChart) en iconografie voor extra duidelijkheid.

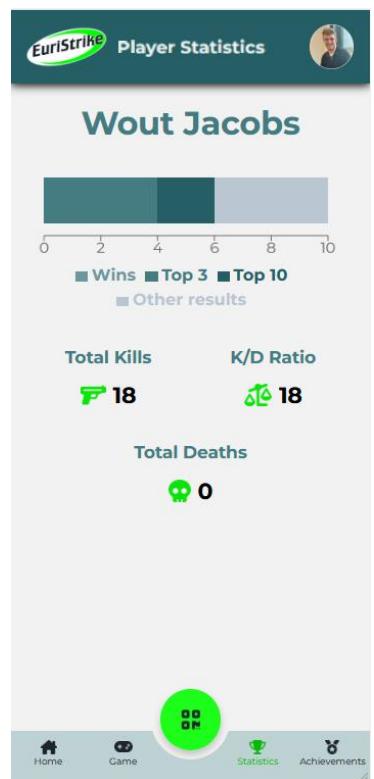


Figuur 15: statistieken van een game

3.10.1. Functionaliteit en dataverwerking

Bij het laden van de pagina worden de statistieken van de gebruiker opgehaald via de getUserStats-functie. Deze functie haalt zowel het externe gebruikersprofiel als de bijbehorende spelstatistieken op via een API-call. Zodra de gegevens succesvol zijn opgehaald, worden ze als volgt verwerkt:

1. Filtering van relevante data
De statistieken worden gefilterd op specifieke waarden die betrekking hebben op positie en prestaties (zoals wins, top 3, top 10 en totaal aantal gespeelde games).
2. Berekening van extra statistiekvelden
Op basis van de opgehaalde data worden aanvullende berekeningen uitgevoerd:
 - o otherResults wordt berekend door het aantal games buiten de top 10 af te leiden uit het totaal.
 - o totalTop3 en totalTop10 worden gecorrigeerd door eerdere posities eruit te filteren (bijv. top 3 zonder wins, top 10 zonder top 3).
3. Voorbereiding voor grafische weergave
De data worden omgevormd naar een formaat geschikt voor de BarChart component van recharts. De grafiek toont de spreiding van prestaties in categorieën: Wins, Top 3, Top 10 en overige resultaten.
4. Foutafhandeling en fallback-waarden
Indien de gebruiker geen statistieken heeft of het totaal aantal games gelijk is aan nul, wordt automatisch een fallback-waarde van 1 gebruikt om verdelingen mogelijk te maken.



Figuur 16: gebruikers statistieken

De pagina is opgebouwd uit de volgende onderdelen:

- Spelersnaam: De volledige naam van de externe gebruiker wordt prominent weergegeven.
- Statistiekenkaart: De data worden in twee vormen gepresenteerd:
 1. Staafdiagram: Een horizontale grafiek die het aantal wedstrijden per resultaatcategorie weergeeft.
 2. Statistiekenrij: Een rij met drie aparte items:
 - Totale kills (met GunIcon)
 - K/D-ratio (met balans-icon afhankelijk van verhouding)
 - Totale deaths (met SkullIcon)
- Iconografie voor K/D-ratio
Op basis van de verhouding tussen kills en deaths wordt automatisch een van de drie iconen getoond:
 - o Gelijk: ScaleBalanceIcon
 - o Voordeel: ScaleUnBalanceIcon
 - o Nadeel: ScaleUnBalanceFlipIcon
- Tooltip-logica
In de grafiek worden cumulatieve waarden getoond via aangepaste tooltips. Zo wordt voor Top 3 ook het aantal wins meegeteld, en voor Top 10 zowel Top 3 als Wins.

3.10.3. Backendstructuur: User Entity en API

De backend ondersteunt deze functionaliteit via de User entity in het domeinmodel. Deze bevat alle noodzakelijke velden voor het opslaan van spelerstatistieken, zoals:

- TotalKills
- TotalDeaths
- TotalWins

- TotalTop3
- TotalTop10
- TotalGamesPlayed

De User entiteit biedt daarnaast methodes om de games gespeeld dynamisch bij te werken, en een lijst van gekoppelde wedstrijden via GameUsers.

De interactie met de backend wordt gefaciliteerd via de userApi module. Deze biedt methodes om gebruikers op te halen, te creëren of bij te werken:

- getUserByExternalId: Haalt een gebruiker op aan de hand van het externe ID.
- getOrCreateUser: Zorgt voor automatische aanmaak indien de gebruiker nog niet bestaat.
- updateUser: Werkt bestaande gebruikersdata bij.

Deze architectuur maakt het mogelijk om zowel geregistreerde als nieuwe spelers naadloos te ondersteunen.

3.10.4. Ontwerp en synchronisatie

De pagina is ontworpen met een sterke focus op overzicht, gebruiksgemak en realtime feedback. Door gebruik te maken van duidelijke iconen, interactieve grafieken en directe foutafhandeling via toasts, ontstaat een intuïtieve en informatieve gebruikerservaring.

Bij het laden of falen van de data wordt automatisch feedback getoond via een toastnotificatie. Indien er geen statistieken beschikbaar zijn, toont de pagina een fallbackbericht.

Dankzij de modulaire opbouw met hooks (useAuth, useLayoutConfig, useToast) blijft de logica gescheiden en herbruikbaar. De grafische weergave via Recharts zorgt voor dynamische visualisatie, terwijl de componentstructuur (zoals PlayerStatistics) overzichtelijk en uitbreidbaar is.

3.11. Qr code laten scannen

Elke speler beschikt over een unieke QR-code zoals in *Figuur 17* te zien is die gegenereerd wordt op basis van zijn of haar externalUserId. Deze code wordt op het scherm weergegeven en kan worden gescand door de target van de speler om een kill te registreren of interactie te starten. De QR-code bevat een versleutelde waarde, zodat misbruik of manipulatie wordt voorkomen.

3.11.1. Generatie van de QR-code

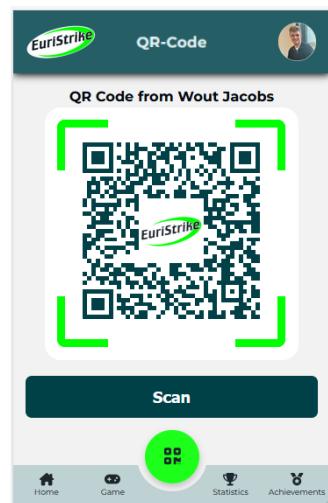
De QR-code wordt gegenereerd met behulp van de externe NPM package qrcode.react, die een SVG-renderbare QR-code teruggeeft. De gegenereerde waarde is een Base64-gecodeerde string waarin een specifiek prefix (euri:) gecombineerd wordt met de externalUserId van de speler. Deze versleuteling gebeurt in de functie encryptText

Deze QR-code wordt dynamisch weergegeven in het midden van het scherm, centraal binnen een visueel aantrekkelijke kaart.

De QR-code wordt weergegeven binnen een qr-card container, die met CSS dynamisch wordt gestyled en afhankelijk is van de grootte van het scherm. Via useRef wordt de breedte van het element gemeten, en op basis daarvan wordt de grootte (qrCodeSize) van de QR aangepast, samen met de padding. Deze aanpak garandeert responsieve schaalbaarheid, zelfs op mobiele toestellen.

Rondom de QR-code worden visuele markeringen geplaatst in de vorm van frame-lijnen (.frame-line). Deze zijn gepositioneerd op de vier hoeken en zijdes van het kader en hebben een herkenbare groene kleur, die de focus op de QR versterken en tegelijkertijd een gamified uitstraling geven.

Om de QR-code te genereren, moet eerst de huidige ingelogde gebruiker opgehaald worden via een API-call naar getAuthenticatedUser(). Deze call maakt gebruik van de accounts-waarde die uit de Azure MSAL-authenticatie komt. De toestand (loading, error, currentUser) wordt bijgehouden via useState.



Figuur 17: qr code

- Tijdens het laden wordt het bericht "Data is loading..." weergegeven.
- Indien er een fout optreedt, wordt dit getoond als tekstbericht op het scherm.
- Als er geen geldige gebruiker wordt gevonden, verschijnt de melding: "No user was found to display the QR code".

3.11.2. Scan-knop en interactie

Naast het tonen van de eigen QR-code bevat de pagina een duidelijke "Scan" knop onderaan. Deze knop leidt de gebruiker naar de scan-interface (Routes.QrScan) waar zij de QR-code van hun target kunnen inscannen. Deze interactie wordt afgehandeld via useNavigate() en een eenvoudige handleClick() functie. De QR-code zelf heeft de volgende kenmerken:

- Dynamisch formaat: Past zich aan op basis van containerbreedte.
- Logo in QR-code: Centraal in de QR wordt het EuriStrike-logo ingesloten, met excavate: true om plaats vrij te maken binnen het patroon.
- Foutcorrectieniveau: ingesteld op "H" (hoog) voor maximale scanbaarheid ondanks het centrale logo.

De bijhorende .css-file bevat aangepaste stijlen voor positionering, kleur en responsiveness:

- qr-code-wrapper: afgeronde hoeken, witte achtergrond, maximale breedte van 400px.
- .frame-line: geanimeerde hoeken en zijlijnen in felgroen.
- Responsieve aanpassingen voor schermen onder 768px en 480px met aangepaste padding en schaal.

3.12. Qr code scannen

Via deze pagina kan de speler de QR-code van zijn/haar target scannen zoals in *Figuur 18*. Wanneer een scan succesvol is, wordt de eliminatie doorgegeven aan de backend en wordt de speler doorgestuurd naar de juiste gamepagina. Deze feature is een essentieel onderdeel van de gameplay, aangezien het scannen het officiële bewijs vormt van een succesvolle eliminatie.

Wanneer de pagina wordt geladen:

- De naam van de pagina wordt ingesteld via updatePageName('QR Scan').
- De camera wordt geactiveerd via de @yudiel/react-qr-scanner NPM package, die QR-codes detecteert binnen een video feed.
- Zodra een QR-code succesvol gedetecteerd wordt, wordt de handleScan() functie uitgevoerd.

3.12.1. Scan proces

Belangrijkste stappen in het scanproces

Authenticatie controleren: De functie verifieert eerst of de speler geauthenticeerd is en een externalUserId heeft. Zoniet, gebeurt er niets.

QR-code decoderen: De gedetecteerde waarde wordt eerst Base64 gedecodeerd met `atob()`, en vervolgens geverifieerd op het correcte prefix `euri:`. Deze prefix zorgt ervoor dat alleen QR-codes van het juiste formaat aanvaard worden.

Dubbele scans vermijden: Er wordt een `useRef()` gebruikt om te onthouden welke QR-code het laatst gescand werd, zodat dubbele scans vermeden worden.

Eliminatie verwerken: De target (`scannedExternalUserId`) wordt opnieuw geënkodeerd met `btoa()` en samen met het ID van de huidige speler doorgestuurd naar de backend via de `gameUserApi.eliminateTarget()` functie.

Resultaat tonen:

1. Bij succes: Een toastbericht geeft aan dat de eliminatie gelukt is en de gebruiker wordt doorgestuurd naar de juiste gamepagina.
 2. Bij fout: Een error-toast verschijnt met de juiste foutmelding.
- Gebruikte NPM package
 - Voor het scannen van de QR-code maken we gebruik van: `@yudiel/react-qr-scanner`

Deze React-component maakt het mogelijk om op een performante en responsieve manier QR-codes te detecteren in real-time via de camera van het toestel.

Belangrijke opties die geconfigureerd zijn:

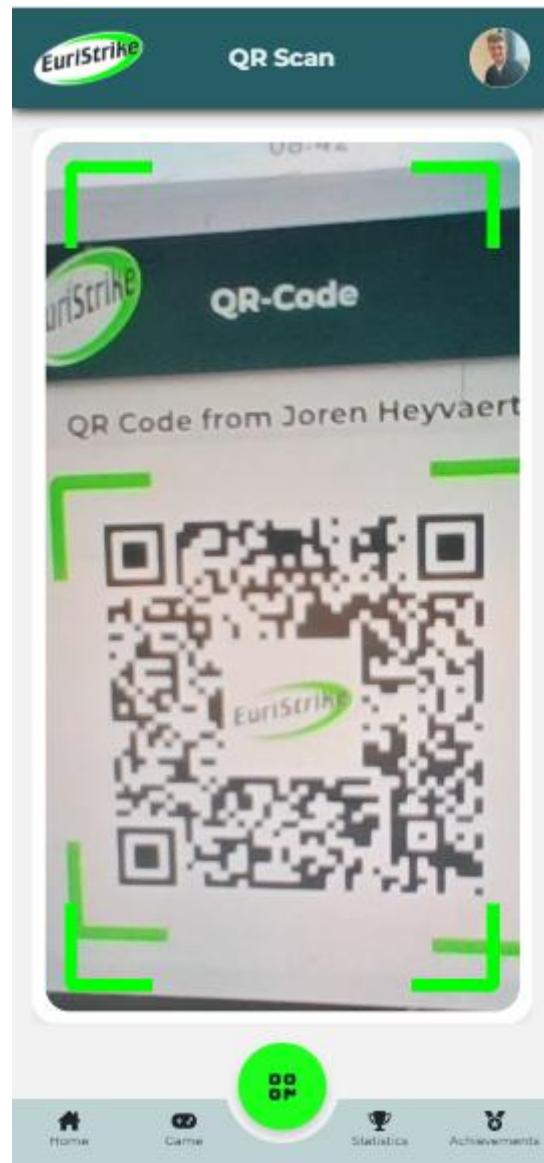
- formats: ['qr_code']: enkel QR-codes worden herkend.
- scanDelay: 500: scanfrequentie is 1 scan per halve seconde.
- finder: false: de visuele zoekindicator is uitgeschakeld voor een zuivere look.
- audio: false: geen geluid bij detectie.

3.12.2. Weergave en styling van de camera-feed

De camera-feed wordt weergegeven binnen een gestileerde kaart (camera-card) met afgeronde hoeken en schaduw. Om het scangebied visueel af te bakenen, zijn er frame-lijnen toegevoegd op de vier hoeken en zijdes in een opvallende groene kleur. Deze elementen zijn identiek aan die van de QR-codepagina voor een uniforme UI/UX.

De bijhorende CSS bevat responsive styling zodat de interface zowel op desktop als mobiele toestellen duidelijk en gebruiksvriendelijk blijft.

- Maximale breedte camera: 400px, aangepast naar 90-95vw op kleinere schermen.
- Hoogte camera-feed: varieert tussen 300px en 200px afhankelijk van de schermgrootte.
- Frame-lijnen: groene accenten op de randen om het scanvak te kaderen.



Figuur 18: scan Qr-Code

3.12.3. Backendverwerking en spelmechaniek

Wanneer de eliminatie wordt doorgestuurd naar de backend, wordt de request afgehandeld door de `EliminateTargetByUserId` use case binnen de .NET backend, die gebruikmaakt van MediatR en Entity Framework Core. Hierin wordt gecontroleerd of:

- De gescande gebruiker overeenkomt met de huidige target van de speler.
- De speler en target zich in een actieve game bevinden.

Bij een geldige eliminatie wordt de status van de target aangepast naar "uitgeschakeld". Vervolgens:

- In standaardgames krijgt de laatst overblijvende speler automatisch de hoogste ranking.
- In Elo-gebaseerde games wordt op basis van het Elo-algoritme een nieuwe target toegewezen, waarbij rekening wordt gehouden met gelijkaardige sterktes tussen spelers.
- Na succesvolle verwerking ontvangt de speler via de front-end visuele feedback en wordt hij/zij omgeleid naar de juiste gamepagina. Eventuele fouten zoals een ongeldig target, een reeds uitgeschakelde speler of een verkeerde QR-code worden correct afgehandeld via error-notificaties.

3.13. Game owner management

Binnen de applicatie is er een aparte beheerinterface ontworpen, waarmee de huidige game owner (de speler die het spel heeft aangemaakt) actief kan bepalen wie er mee het spel mee kan doen en wie als beheerder fungert. Dit wordt gerealiseerd via een combinatie van React-componenten in de front-end en specifieke command-handlers in de back-end.

3.13.1. Functionaliteit in de Front-end

De GameOwnerCard-component geeft voor iedere deelnemer in een spel een overzichtelijke kaart weer met de volgende elementen:

- Gebruikersinformatie: De kaart toont de profielfoto en de naam van de speler. Indien een speler al uitgeschakeld is, wordt dit visueel aangegeven met een grijze filter en een kleine “†” als doodssindicatie.
- Toggle voor Game Owner Status: Met een aanpasbare toggle-switch kan de huidige game owner bepalen of een speler extra beheersrechten krijgt binnen het spel. Wanneer de switch wordt bediend, wordt de nieuwe status direct via een API-call naar de backend doorgegeven. De status wordt vervolgens up-to-date gehouden in de context, zodat de wijziging in de interface wordt gereflecteerd.
- Eliminatieknop: Indien nodig kan de game owner een speler elimineren. Dit gebeurt via een knop waarop wordt bevestigd of de speler echt uitgeschakeld moet worden. Bij bevestiging wordt een eliminatieverzoek verstuurd naar de backend, waarna de betrokken speler wordt gemarkeerd als niet-levend.

3.13.2. Admin Home Pagina

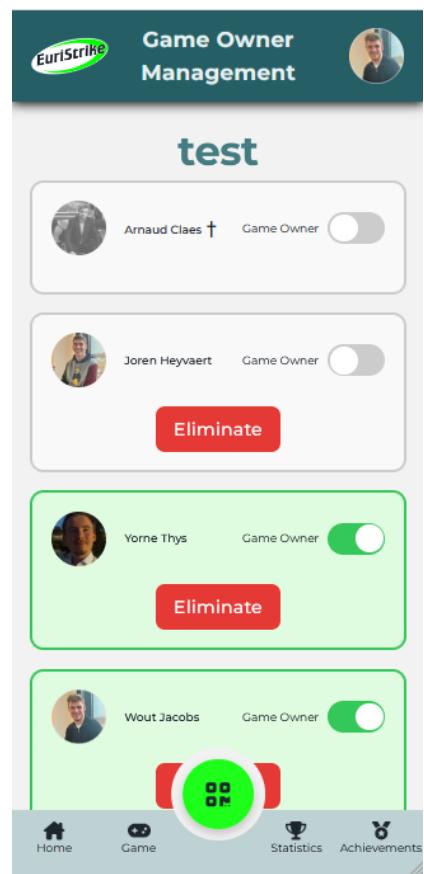
De AdminHome-pagina is de centrale beheeromgeving voor de game owner. Op deze pagina worden alle huidige spelers van een spel opgehaald en weergegeven, zodat de game owner een volledig overzicht heeft van wie er actief is en wie niet een voorbeeld hier van is *Figuur 19*. Via deze interface kan de beheerder:

- De status van elke speler aanpassen door de toggle in de GameOwnerCard te bedienen.
- Spelers elimineren indien zij zich niet aan de spelregels houden of per ongeluk zijn toegevoegd.
- In real-time zien welke wijzigingen er plaatsvinden in de status en statistieken van de spelers.

3.13.3. Backend Verwerking

Op de serverzijde wordt het eliminatieproces afgehandeld door een specifieke use case, genaamd EliminateTargetByUserId. Deze use case is geïmplementeerd met behulp van MediatR en Entity Framework Core en bevat de volgende stappen:

- Validatie en Identificatie: De request ontvangt een Base64-gecodeerde target-ID, die wordt gedecodeerd om het externe gebruikers-ID te achterhalen. Vervolgens wordt gecontroleerd of de target daadwerkelijk de huidige target is van de aanvallende speler en of beide spelers in een actieve game zitten.
- Verwerking op Basis van Game Mode:
 - Ranked Games (Elo): Voor games in ranked modus wordt het Elo-algoritme toegepast om de rankings van zowel de aanvaller als de target bij te werken. Afhankelijk van de aard van de eliminatie (handmatig of automatisch) wordt vervolgens de targetstatus aangepast en wordt de nieuwe target via een aangepaste shuffling herverdeeld.
 - Standard Games: In standaardgames wordt de target simpelweg uitgeschakeld. Daarnaast wordt de ranking van de overgebleven spelers gerestructureerd, zodat de laatst overblijvende speler automatisch de hoogste ranking krijgt.
- Statistieken Bijwerken: Naast de statuswijzigingen worden ook de statistieken van de betrokken spelers (zoals kills en deaths) geüpdatet. Dit omvat zowel individuele spelerstatistieken als spelbrede statistieken, waarbij de relevante data in de database wordt opgeslagen.



Figuur 19: beheer spel eigenaar

3.13.4. Samenvatting

De beheerfunctionaliteit voor game owners stelt de spelbeheerder in staat om actief te bepalen wie er mee het spel kan beheren of als deelnemer moet doorstromen. Via de intuïtieve GameOwnerCard kunnen zij:

- Spelers status aanpassen en hen als extra beheerders aanwijzen.
- Spelers elimineren wanneer dit noodzakelijk is.
- Real-time de impact van deze acties zien in de Admin Home pagina, waar alle spelers overzichtelijk worden weergegeven.

De backend ondersteunt dit proces door op een veilige en betrouwbare manier de eliminaties te verwerken, de rankings te berekenen (afhankelijk van de spelmodus) en de bijgewerkte statistieken op te slaan.

Hierdoor zorgt de integratie van deze onderdelen voor een dynamisch en flexibel beheer van het spel, wat bijdraagt aan een betere spelervaring en het naleven van spelregels.

3.14. Rules

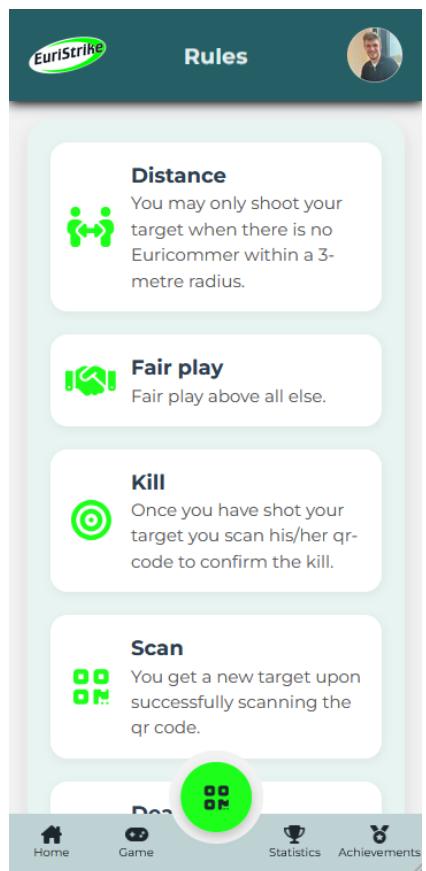
De Rules pagina toont alle spelregels die van toepassing zijn, afhankelijk van in welke gamemode je je bevindt. Deze regels zijn afkomstig uit een JSON-bestand waarin elke regel gelinkt is aan één of meerdere gamemodes (zoals *Ranked*, *Standard*, *Free For All*, of *Any*). Zie *Figuur 20*. Wanneer de pagina wordt geopend, wordt de naam van de pagina ingesteld op “Rules” en wordt op basis van de geselecteerde game (uit de context) alle relevante informatie opgehaald. Aan de hand van deze informatie wordt bepaald in welke gamemode de speler zich bevindt. Alle regels uit het JSON-bestand worden vervolgens gegroepeerd op basis van hun bijhorende gamemodes. De volgorde waarin deze regels worden getoond is als volgt:

1. Eerst de regels die gelden voor iedereen (*Any*).
2. Daarna de regels specifiek voor de actieve gamemode.
3. Als er geen game actief is, worden de regels voor alle gamemodes weergegeven in de standaard volgorde.

Elke regel bevat een naam, beschrijving en icoon. Deze worden weergegeven in aparte secties per gamemode. De iconen worden automatisch weergegeven op basis van de opgegeven naam uit het JSON-bestand.

Indien er geen regels beschikbaar zijn voor de actieve gamemode, wordt er een melding getoond dat er geen regels gevonden zijn.

Bij een fout tijdens het ophalen van de gamegegevens wordt er een duidelijke foutmelding getoond aan de gebruiker via een toast-notificatie. Kortom: deze pagina zorgt ervoor dat spelers snel en visueel overzichtelijk alle relevante regels kunnen nalezen, aangepast aan de context van hun actieve spel.



Figuur 20: spel regels

3.15. Manage Admin

In de Gotcha-applicatie bestaat er een aparte beheerpagina genaamd Make Admin, die uitsluitend toegankelijk is voor superadmins. Deze pagina maakt het mogelijk om te bepalen welke andere Euricomers beheerdersrechten (adminrechten) krijgen binnen de applicatie zie *Figuur 21*. Admins hebben toegang tot belangrijke onderdelen van het systeem zoals het beheren van games, gebruikers, statistieken, en andere beheermodules. Deze functionaliteit is essentieel voor het veilig en schaalbaar delegeren van verantwoordelijkheden binnen de organisatie.

3.15.1. Functionaliteit van de toggle-switch

Op de Make Admin-pagina wordt een lijst weergegeven van alle medewerkers en stagiairs binnen Euricom. Deze gebruikers worden opgehaald via een externe service, waarbij het systeem automatisch onderscheid maakt tussen employees en interns. Dit zorgt ervoor dat ook gebruikers zonder een intern Gotcha-account zichtbaar en beheerd kunnen worden. Indien nodig wordt er automatisch een intern profiel aangemaakt wanneer de adminstatus wordt aangepast.

Elke gebruiker wordt gepresenteerd in een afzonderlijke IsAdminCard-component, die de volgende informatie bevat:

- Profielfoto: Een afbeelding van de gebruiker voor snelle visuele herkenning.
- Volledige naam: De voor- en achternaam van de gebruiker worden weergegeven.
- Toggle-switch: Een schakelaar die aangeeft of de gebruiker momenteel een admin is (aan) of niet (uit).

De functionaliteit is zodanig opgezet dat de superadmin via de toggle-switch eenvoudig de adminstatus van een gebruiker kan in- of uitschakelen. Deze wijziging wordt onmiddellijk doorgestuurd naar de backend via een API-aanroep naar setAdmin. Indien de betreffende gebruiker nog geen Gotcha-profiel heeft, wordt deze automatisch aangemaakt via de getOrCreateUser-functie.

De pagina bestaat uit de volgende belangrijke onderdelen:

1. Gegevensophaling: Bij het openen van de pagina worden alle externe gebruikers automatisch geladen en weergegeven. Tevens wordt voor elke gebruiker nagegaan of ze al een gekoppeld Gotcha-profiel hebben.
2. Automatisch aanmaken van gebruikers: Wanneer een gebruiker nog geen intern profiel heeft, wordt dit automatisch aangemaakt zodra de adminstatus wordt aangepast.
3. Zoek- en filterfunctionaliteit: De superadmin heeft toegang tot een zoekveld om snel specifieke gebruikers terug te vinden op basis van naam. Dit maakt het beheren van grote gebruikerslijsten eenvoudiger.
4. Real-time statusupdates: Wanneer een wijziging wordt doorgevoerd, wordt de UI onmiddellijk aangepast zodat deze overeenkomt met de actuele status van de gebruiker. Deze status wordt visueel weergegeven via een actieve of inactieve styling van de kaart.

3.15.2. Ontwerp en synchronisatie

De pagina is ontworpen met het oog op eenvoud, snelheid en betrouwbaarheid. Dankzij de automatische synchronisatie van gebruikersstatussen met de backend en de mogelijkheid om snel te filteren of zoeken, behoudt de superadmin altijd het overzicht. Bovendien voorkomt de getOrCreate-logica fouten bij het toekennen van rechten aan gebruikers die nog niet eerder met de applicatie in aanraking zijn gekomen.

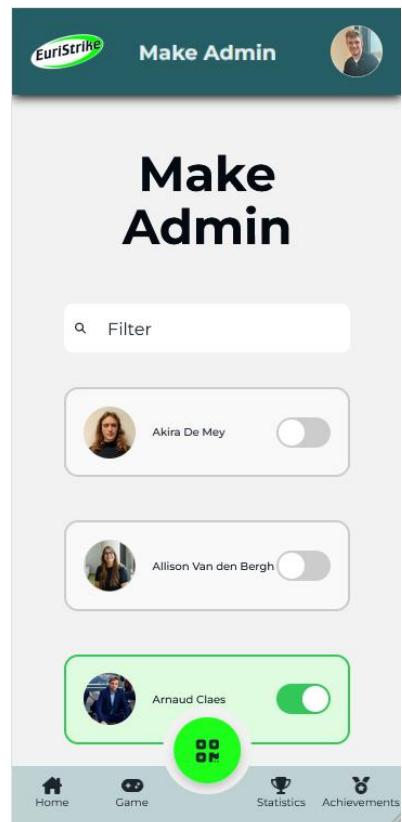
Tot slot speelt de IsAdminCard-component een centrale rol in de gebruikersinterface. Deze component beheert zowel de weergave van de gebruikersinformatie als de adminstatuslogica. Bij het activeren van de toggle wordt direct een visuele update getoond (bijvoorbeeld een groene actieve status), en wordt het bijbehorende gebruikersobject bijgewerkt in de interne staat van de pagina. Op deze manier blijft alles coherent en up-to-date zonder dat de pagina handmatig herladen hoeft te worden.

Dankzij deze pagina kunnen superadmins op een efficiënte en gecontroleerde manier bepalen wie de Gotcha-applicatie verder mag beheren. Dit draagt bij aan een flexibeler en schaalbare governance-structuur binnen de applicatie en zorgt ervoor dat er altijd voldoende bevoegde personen beschikbaar zijn om het platform correct aan te sturen.

3.15.3. Backend: Wijziging van de Adminstatus

De backend handelt het PUT-verzoek af wanneer de superadmin de adminstatus van een gebruiker wil wijzigen. Het systeem verwerkt dit verzoek door de ID van de gebruiker en de nieuwe adminstatus (true/false) door te geven. Het proces in de backend werkt als volgt:

1. Het Request wordt gestuurd naar de backend, waarbij de UserId en de IsAdmin status worden meegegeven.
2. De Handler haalt de gebruiker op via hun ID uit de database.



Figuur 21: beheer admin

3. De adminstatus van de gebruiker wordt geüpdateet naar de nieuwe waarde.
4. De wijziging wordt opgeslagen in de database, en een UserDto wordt geretourneerd met de bijgewerkte gegevens van de gebruiker.

Door deze logica kan de superadmin de gebruikersstatus efficiënt beheren, waarbij de applicatie automatisch controleert of de gebruiker bestaat. Indien de gebruiker nog geen profiel heeft, wordt deze automatisch aangemaakt via de getOrCreateUser-functie. Het resultaat wordt vervolgens teruggestuurd naar de front-end, waar de adminstatus direct wordt aangepast.

Dit proces maakt gebruik van de IuserRepository voor het ophalen en bijwerken van gebruikers en zorgt ervoor dat alle wijzigingen consistent worden doorgevoerd, wat resulteert in een veilige en betrouwbare eadminbeheerfunctionaliteit binnen de applicatie.

3.16. Ranked game

Binnen de Gotcha-applicatie is een aparte spelmodus geïntegreerd genaamd Ranked Mode, bedoeld voor spelers die op een competitieve manier willen deelnemen aan een game. In tegenstelling tot de standaardmodus worden in Ranked Mode de prestaties van spelers voortdurend geëvalueerd op basis van hun eliminaties en overleving. Het systeem maakt gebruik van een aangepaste Elo-scoreberekening om rankings toe te wijzen en bij te werken. Voor meer info over wat Elo is kan je deze vinden in de analyse van game mode 2.

Deze spelmodus is ontwikkeld met het oog op eerlijke competitie, waarbij elke succesvolle eliminatie en eliminatiepoging directe impact heeft op de speler zijn of haar positie binnen de ranglijst.

3.16.1. Spelverloop en Toewijzing van Targets

Wanneer een speler een andere speler elimineert in Ranked Mode, wordt dit geregistreerd als een directe competitieve interactie. De applicatie vergelijkt de scores van zowel de aanvaller als het doelwit, en berekent op basis daarvan nieuwe rankings.

Na een succesvolle eliminatie krijgt de aanvaller automatisch een nieuw doelwit toegewezen via een intern systeem dat spelers herschikt op basis van hun Elo-score. Deze herschikking houdt rekening met zowel eerlijkheid als uitdaging: hoe hoger de rang, hoe moeilijker het wordt om nieuwe doelwitten te krijgen.

3.16.2. Automatische Elo-berekening

Het centrale element van de Ranked Mode is het Elo-algoritme, dat scores toekent op basis van de relatieve sterkte van spelers:

- Als een speler met een lagere score een hogere uitschakelt, ontvangt hij of zij een grotere boost.
- Als een speler met een hogere score een lagere uitschakelt, stijgt de ranking slechts minimaal.
- Verliezen van een speler met een lagere score resulteert in een grotere daling van de ranking.

Dit dynamisch systeem zorgt ervoor dat de ranking voortdurend accuraat blijft en spelers motiveert om beter te presteren.

3.16.3. Backend: Verwerking van Eliminaties

Wanneer een speler een tegenstander elimineert in Ranked Mode, wordt dit verwerkt via een backend-flow die de volgende stappen omvat:

1. De front-end stuurt een API-aanroep met de gegevens van zowel de aanvaller als het doelwit.
2. De backend haalt beide spelers op via hun unieke gebruikers- en spel-ID.
3. De backend voert de Elo-berekening uit, en werkt vervolgens de ranking van beide spelers bij.
4. De status van het doelwit wordt aangepast naar "uitgeschakeld", en het doelwit van de aanvaller wordt opnieuw toegewezen.
5. De backend verwerkt aanvullende statistieken zoals kills en deaths, en synchroniseert dit met de algemene gebruikersdata.

Indien nodig wordt ook automatisch een nieuwe target geselecteerd uit een herschikte lijst van spelers binnen dezelfde game, gebaseerd op hun rang en status.

3.16.4. Gebruikersinterface

Aan de kant van de gebruikersinterface is de Ranked Mode naadloos geïntegreerd in de standaard gameflow, met enkele specifieke UI-aanpassingen:

- Live ranking updates: Spelers zien onmiddellijk hun nieuwe score na een eliminatie via real-time visuele feedback.
- Doelwitvernieuwing: Bij een succesvolle kill krijgt de speler onmiddellijk een nieuw doelwit te zien, zonder herladen van de pagina.
- Feedback en status: Via animaties, kleurcodes of meldingselementen wordt visueel weergegeven of de actie succesvol was en wat de impact op de ranking is.
- Fair matchmaking: Dankzij de dynamische selectie van nieuwe doelwitten via de backend is de kans op oneerlijke of te gemakkelijke doelwitten minimaal.

3.16.5. Elo-berekening in Ranked Mode

Elke speler in de Ranked Mode heeft een huidige elo score (rating), die wijzigt naargelang ze een andere spelers uitschakelen of zelf worden uitgeschakeld. De wijziging in deze score wordt bepaald op basis van een wiskundige kansberekening en het resultaat van de actie.

Vooraleer de score wordt aangepast, berekent het systeem de kans dat een speler zou winnen op basis van de huidige scores van beide spelers.

- Als twee spelers een gelijkaardige score hebben, is de kans ongeveer 50% voor elk.
- Als een speler met een veel lagere score een sterke speler uitschakelt, is de kans klein, en wordt hij of zij sterker beloond.

Deze kans wordt bepaald aan de hand van het verschil tussen de twee scores. Hoe groter het verschil, hoe meer de kans opschuift naar de sterkere speler.

Vervolgens vergelijkt het systeem de werkelijke uitkomst (win of verlies) met de berekende kans. Op basis daarvan wordt de score aangepast:

- Win: de score gaat omhoog.
- Verlies: de score daalt.

```
/*
 * formula:
 * R_NewR_Old+K (W-E)
 * E=1/(1+10^(R2-R1 )/400)
 *
 * R = Rating
 * K = factor how quickly your rating can go up or down after a game.
 * W = actual result
 * E = The chance with which you can win a game.
 */
private const int ratingSensitivity = 20;

/*
 * actual result (e.g. 1 if you win 0 if you lose)
 * this can change depending on the speed of the game
 */
2 references
private static class gameResult
{
    public const int Win = 1;
    public const int Lose = 0;
}

7 references
private static (double newKillerElo, double newTargetElo) UpdateEloRating (double killerElo, double targetElo)
{
    double ProbKiller = Probability((int)targetElo, (int)killerElo);
    double ProbTarget = Probability((int)killerElo, (int)targetElo);

    killerElo = killerElo + ratingSensitivity * (gameResult.Win - ProbKiller);
    targetElo = targetElo + ratingSensitivity * (gameResult.Lose - ProbTarget);

    return (killerElo, targetElo);
}

/*
 * The chance with which you can win a game.
 */
2 references
private static double Probability(int opponentRating, int ownRating)
{
    return 1.0 / (1 + Math.Pow(10, (opponentRating - ownRating) / 400.0));
}
```

Figuur 22: elo rating bepalen

De mate waarin de score verandert, wordt bepaald door een vaste factor in het systeem. Deze waarde bepaalt hoe "gevoelig" de score is voor wijzigingen.

- Bij een hoge gevoeligheid veranderen scores sneller.
- Bij een lage gevoeligheid blijven scores stabiever over tijd.

In Gotcha is deze gevoeligheid ingesteld op een gemiddelde waarde om een goede balans te vinden tussen stabiliteit en competitie.

Na het vergelijken van verwachting en uitkomst wordt de nieuwe score berekend:

- De winnaar krijgt punten toegekend, in verhouding tot hoe onverwacht de winst was.
- De verliezer verliest punten, ook in verhouding tot hoe onverwacht het verlies was.

Beide spelers krijgen dus altijd een nieuwe score toegewezen na elke interactie. In *Figuur 22* zie je de berekening

3.16.6. Unit Test elo

De unit tests rondom de EloUtils.UpdateEloRating-methode zijn van cruciaal belang om de betrouwbaarheid en correctheid van de ELO-berekeningen te waarborgen. Aangezien de elo-rating een kerncomponent is van de spelersbeoordeling binnen de Gotcha-applicatie, dienen wijzigingen aan deze logica zorgvuldig getest te worden.

De tests controleren diverse scenario's waarbij de initiële ELO-waarden van de "killer" en "target" verschillen, en valideren of de nieuwe ratings conform de verwachte resultaten worden berekend. De testcases zijn zodanig opgezet dat ze zowel normale als randgevallen omvatten, zoals:

- Gelijke initiële ratings, waarbij de winnaar precies de juiste hoeveelheid punten wint en de verliezer een gelijkwaardige hoeveelheid verliest.
- Scenarios waarbij de winnaar een hogere of lagere rating heeft dan de verliezer, met daarbij een proportioneel aangepaste winst- of verliesscore.
- Zekerstelling dat de som van de ratingwijzigingen altijd nul blijft (zero-sum principe).
- Validatie dat de rating van de winnende speler altijd stijgt, en die van de verliezende speler altijd daalt.

De ELO-berekening is een wiskundige kernfunctie die direct invloed heeft op de ranking en matchmaking van spelers. Kleine fouten of regressies in deze logica kunnen leiden tot oneerlijke ratingveranderingen, wat de spelervaring kan ondermijnen en het vertrouwen in het platform kan schaden.

Door het opstellen van uitgebreide en gedetailleerde unit tests wordt gegarandeerd dat elke aanpassing aan de EloUtils-implementatie geen onverwachte neveneffecten veroorzaakt. Dit maakt het mogelijk om met vertrouwen refactoringen, optimalisaties of functionele uitbreidingen door te voeren, terwijl de correcte werking van het systeem wordt gewaarborgd.

3.17. Free for all

De Free For All-modus binnen de Gotcha-applicatie is ontworpen om spelers een dynamische en onvoorspelbare spelervaring te bieden. In tegenstelling tot andere modi zoals Ranked of Classic, worden in deze modus geen doelwitten toegewezen. Elke speler is zowel jager als prooi, en kan elk ander actief spelend lid uitschakelen.

Deze spelvorm vereist een andere strategie van de deelnemer: het vergt oplettendheid, timing en inzicht in het gedrag van andere spelers. Omdat er geen target wordt toegewezen, is het initiatief volledig aan de speler zelf.

3.17.1. Spelverloop zonder toegewezen targets

In de Free For All-modus zijn alle actieve deelnemers gelijktijdig elkaars tegenstander. Zodra de game van start gaat, is elke speler vrij om andere spelers uit te schakelen, ongeacht hun rang of status. De applicatie registreert elke succesvolle eliminatie, en past de score van de speler aan op basis van vaste puntenmechanismen.

Doordat er geen target wordt toegewezen, vervalt de noodzaak tot herschikking van spelers of het opnieuw toewijzen van doelwitten. Dit vermindert de complexiteit aan de backendzijde en verhoogt tegelijkertijd de intensiteit van het spel.

3.17.2. Eliminatieregels en scoreverwerking

Bij een succesvolle eliminatie ontvangt de speler punten op basis van vooraf gedefinieerde parameters:

- Elke uitschakeling levert een vast puntenaantal op.

- Optioneel kunnen bonuspunten worden toegekend voor killstreaks of eliminaties binnen een bepaald tijdsvenster.

Er wordt geen Elo-berekening toegepast zoals in Ranked Mode. In plaats daarvan werkt Free For All met een gestandaardiseerd puntensysteem dat de nadruk legt op kwantiteit en consistentie in prestaties.

3.17.3. Backendverwerking

De backendstructuur voor Free For All is aangepast op de eenvoud en snelheid van verwerking:

1. Bij elke eliminatie stuurt de front-end een API-aanroep met de gegevens van de aanvaller en het slachtoffer.
2. De backend valideert of beide spelers actief zijn in het spel en of de eliminatie geldig is.
3. De status van het slachtoffer wordt aangepast naar "uitgeschakeld".
4. De score van de aanvaller wordt verhoogd met een vast puntenaantal.
5. Statistieken zoals kills en deaths worden bijgewerkt en gesynchroniseerd met het gebruikersprofiel.

De afwezigheid van targetbeheer zorgt voor minder belasting van de serverlogica, wat leidt tot snellere verwerkingsstijden en betere schaalbaarheid.

3.17.4. Gebruikerservaring

De Free For All-modus is geïntegreerd in dezelfde gebruikersinterface als andere modi, met specifieke aanpassingen om het onderscheid duidelijk te maken:

- Geen targetweergave: De speler ziet in plaats daarvan een algemene melding dat iedereen een potentiële tegenstander is.
- Live leaderboard: Een dynamisch scorebord geeft real-time updates van de prestaties van alle spelers.
- Eliminatiefeedback: Via visuele meldingen krijgt de speler onmiddellijke terugkoppeling bij een succesvolle of mislukte poging.

Deze aanpak maakt Free For All tot een toegankelijke maar intensieve spelmodus, waarin spelers worden uitgedaagd om constant alert te blijven. De eenvoud van de spelregels gecombineerd met de open structuur maakt het een populaire keuze voor kortere of snellere spelsessies.

3.18. Qr-code pdf

De QR-code generatie en downloadfunctionaliteit binnen de applicatie is ontworpen met het oog op gebruiksgemak, schaalbaarheid en visuele kwaliteit zoals in *Figuur 23*. Deze functionaliteit stelt een beheerder in staat om met één klik een PDF-bestand te genereren waarin voor elke gebruiker een gepersonaliseerde QR-code opgenomen is. De volledige flow, van React-component tot .NET backend service, is modular opgebouwd, wat zorgt voor een goede onderhoudbaarheid en herbruikbaarheid.

3.18.1. Front-end-integratie: een centrale downloadknop

In de beheerinterface van het spelbeheer is een opvallende knop opgenomen met het label "*Download QR-codes PDF*". Deze knop is centraal gepositioneerd in de header van de pagina en maakt gebruik van een herbruikbare ButtonComponent. Wanneer de knop wordt geactiveerd, wordt via een API-aanroep een gegenereerd PDF-bestand gedownload dat alle QR-codes bevat van de gebruikers die deelnemen aan het spel. Tijdens het genereren verschijnt er een overlay met een visuele spinner en een begeleidende tekst om de gebruiker op de hoogte te houden van de voortgang. Deze UX-keuze draagt bij aan transparantie en gebruiksvriendelijkheid.

De functionaliteit maakt gebruik van gameUserApi.downloadQrCodePdf, een methode die op haar beurt de backend aanspreekt en een PDF-bestand retourneert. Dit bestand wordt vervolgens automatisch aangeboden voor download via een gegenereerde link in de browser, waarbij de bestandsnaam dynamisch aangepast wordt op basis van de naam van het spel.

3.18.2. Backendlogica: QR-code generatie en PDF-samenstelling

De QR-code generatie gebeurt op serverniveau met behulp van de QRCode.Core bibliotheek. Elke QR-code bevat een base64-gecodeerde payload die het externe gebruikers-ID van de speler bevat, voorafgegaan door het label "euri:". Dit zorgt ervoor dat de QR-code eenvoudig te koppelen is aan een unieke gebruiker binnen het systeem. Bovendien wordt een bedrijfslogo (indien beschikbaar) centraal geïntegreerd in de QR-code, wat bijdraagt aan een professionele en herkenbare uitstraling. De plaatsing, groote en achtergrondkleur van het logo zijn zorgvuldig gekozen om visuele interferentie met de QR-code zelf te vermijden.

De gegenereerde QR-codes worden vervolgens samengebracht in een gestileerde PDF, opgebouwd met behulp van de QuestPDF bibliotheek. Per pagina worden maximaal drie gebruikers getoond, elk met hun naam en bijbehorende QR-code. Dit layoutprincipe maakt optimaal gebruik van de beschikbare ruimte en zorgt voor een duidelijke en gestructureerde presentatie. Indien het aantal gebruikers niet deelbaar is door drie, worden lege kolommen toegevoegd om visuele balans te behouden. Elke pagina wordt onderaan voorzien van een paginanummering voor extra overzicht.

De volledige PDF wordt gegenereerd in geheugen via een MemoryStream, zonder tijdelijke opslag op schijf. Dit verhoogt de performantie en vermindert extra I/O-complexiteit.

3.18.3. API-endpoint: veilige en dynamische bestandsnaam

De gegenereerde PDF wordt via een specifiek endpoint (/game/{gameId}/downloadQrCodes) opgevraagd. Dit endpoint maakt gebruik van een MediatR-request die de benodigde game- en gebruikersinformatie

Gerben Schepers



Wim Van Hoye



Kurt Torfs



Dirk Maegh



Peter Cosemans



Sonja Buts



Kristof Degrave



Joeri Druyts



Bart Debeuckelaere



Stijn Geens

Jordy Rymenants

Matthias Ooye

Figuur 23: pdf met Qr-codes

ophaat. Als de opgegeven gameId ongeldig is of het spel niet bestaat, retourneert de API een gepaste foutmelding.

Om een veilige bestandsnaam te garanderen bij het downloaden, wordt de naam van het spel ontdaan van alle tekens die niet toegestaan zijn in bestandsnamen. Hierdoor wordt voorkomen dat gebruikers tegen compatibiliteitsproblemen aanlopen bij het openen of opslaan van het bestand op verschillende besturingssystemen.

3.18.4. Onderhoudbaarheid en uitbreidbaarheid

De volledige flow is opgebouwd volgens de principes van scheiding van verantwoordelijkheden. De QR-code generatie (QrCodeService), de PDF-opbouw (QrPdfGenerator) en de businesslogica (GenerateQrCodePdf.Handler) zijn elk ondergebracht in hun eigen component of service. Hierdoor kan elke laag afzonderlijk uitgebreid of vervangen worden zonder impact op de andere lagen. Zo kan in de toekomst eenvoudig ondersteuning toegevoegd worden voor alternatieve formaten (bv. ZIP met PNG's) of aanvullende metadata per gebruiker.

4. Besluit

De ontwikkeling van de Gotcha Applicatie voor Euricom's jaarlijkse DEV-Cruise heeft aangetoond hoe technologie kan bijdragen aan het versterken van teamdynamiek en samenwerking. Door gebruik te maken van innovatieve functies en een intuïtieve interface, biedt de applicatie een platform waar deelnemers op een speelse en competitieve manier met elkaar kunnen communiceren. De uitgebreide analyse en feedback van gebruikers hebben geleid tot een verfijnde en verbeterde versie van het spel, waarbij uitdagingen zoals zelftargeting en vroege eliminaties effectief zijn aangepakt.

De technische realisatie, waaronder de implementatie van een centrale layoutcomponent en de integratie van veilige authenticatie via Microsoft Entra ID, toont de veelzijdigheid en kracht van de gebruikte technologieën. Deze aanpak heeft niet alleen gezorgd voor een robuuste en gebruiksvriendelijke applicatie, maar ook voor een flexibele basis die klaar is voor toekomstige uitbreidingen en verbeteringen.

Daarnaast heeft de Gotcha Applicatie een belangrijke rol gespeeld in het bevorderen van de teamgeest en het versterken van de onderlinge banden tussen de deelnemers. Door de verschillende spelmodi, zoals "Free for All" en "Ranked Mode", konden deelnemers hun vaardigheden testen en zich meten met anderen in een competitieve maar vriendschappelijke omgeving. Dit heeft niet alleen bijgedragen aan een verhoogde betrokkenheid, maar ook aan een betere samenwerking en communicatie binnen de teams.

De feedback van de gebruikers heeft een cruciale rol gespeeld in de verdere verfijning en verbetering van de applicatie. Door te luisteren naar de ervaringen en suggesties van de deelnemers, konden we gerichte aanpassingen doorvoeren die de gebruiksvriendelijkheid en functionaliteit van de applicatie verder hebben verbeterd. Dit iteratieve proces van continue verbetering heeft geleid tot een applicatie die niet alleen voldoet aan de huidige behoeften, maar ook flexibel genoeg is om in de toekomst aan te passen aan nieuwe eisen en wensen.

Kortom, de Gotcha Applicatie is een waardevolle toevoeging aan Euricom's DEV-Cruise, die niet alleen de teamgeest bevordert, maar ook een voorbeeld stelt van hoe technologische innovatie kan bijdragen aan een betere samenwerking en productiviteit binnen een organisatie. De succesvolle realisatie van dit project is te danken aan de nauwe samenwerking tussen de verschillende teams, de inzet van geavanceerde technologieën en de voortdurende focus op gebruiksvriendelijkheid en efficiëntie. Met deze solide basis is de Gotcha Applicatie klaar voor toekomstige uitbreidingen en verbeteringen, waardoor het een blijvende impact zal hebben op de teamdynamiek en samenwerking binnen Euricom.

LITERATUURLIJST

- (4) Euricom: About | LinkedIn. (z.d.). Geraadpleegd 15 maart 2025, van <https://www.linkedin.com/company/euricom/about/>
- admin, D. (2013, september 9). Gotcha! Spelregels—Een topper in onze groepsaccommodatie. *Ferme du Château*. <https://www.fermeduchateau.be/handleiding/gotcha/>
- Advanced usage QR Code renderers. (z.d.). GitHub. Geraadpleegd 3 mei 2025, van <https://github.com/codebude/QRCoder/wiki/Advanced-usage---QR-Code-renderers>
- akashdubey-ms. (2023, oktober 10). Introduction to Blob (object) Storage—Azure Storage. <https://learn.microsoft.com/en-us/azure/storage/blobs/storage-blobs-introduction>
- An application programming interface for CellNetAnalyzer—ScienceDirect. (z.d.). Geraadpleegd 9 maart 2025, van <https://www.sciencedirect.com.am.thomasmore.e-bronnen.be/science/article/pii/S0303264711000402>
- ardalis. (2023, maart 7). Common web application architectures—.NET. <https://learn.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/common-web-application-architectures>
- Azis, A., Triayudi, A., & Handayani, E. T. E. (2024a). Application of Academic Potential Test for New Student Admission Using Fisher-Yates Shuffle Algorithm. *SAGA: Journal of Technology and Information System*, 2(1), Article 1. <https://doi.org/10.58905/saga.v2i1.254>
- Azis, A., Triayudi, A., & Handayani, E. T. E. (2024b). Application of Academic Potential Test for New Student Admission Using Fisher-Yates Shuffle Algorithm. *SAGA: Journal of Technology and Information System*, 2(1), Article 1. <https://doi.org/10.58905/saga.v2i1.254>
- Azure Blob Storage | Microsoft Azure. (z.d.). Geraadpleegd 3 maart 2025, van <https://azure.microsoft.com/nl-nl/products/storage/blobs>
- Azure SQL - serie SQL-clouddatabases | Microsoft Azure. (z.d.). Geraadpleegd 18 maart 2025, van <https://azure.microsoft.com/nl-nl/products/azure-sql>
- @azure/msal-browser. (2025, maart 4). Npm. <https://www.npmjs.com/package/@azure/msal-browser>
- @azure/msal-react. (2025, februari 18). Npm. <https://www.npmjs.com/package/@azure/msal-react>
- Belshe, M., Peon, R., & Thomson, M. (2015). RFC 7540: Hypertext Transfer Protocol Version 2 (HTTP/2). RFC Editor.
- Bootstrap. (2024, februari 20). Npm. <https://www.npmjs.com/package/bootstrap>
- Browse Fonts. (z.d.). Google Fonts. Geraadpleegd 6 maart 2025, van <https://fonts.google.com/>
- cilwerner. (2024, november 21). Overview of the Microsoft Authentication Library (MSAL)—Microsoft identity platform. <https://learn.microsoft.com/en-us/entra/identity-platform/msal-overview>
- Clark, T. (2025). Implementing CI/CD Safely. In *CI/CD Unleashed* (pp. 97-133). Apress, Berkeley, CA. https://doi.org/10.1007/979-8-8688-1209-5_4
- contributors, M. O., Jacob Thornton, and Bootstrap. (z.d.). Bootstrap. Geraadpleegd 2 maart 2025, van <https://getbootstrap.com/>
- Csató, L. (2024). Club coefficients in the UEFA Champions League: Time for shift to an Elo-based formula. *International Journal of Performance Analysis in Sport*, 24(2), 119-134. <https://doi.org/10.1080/24748668.2023.2274221>
- Dawood, M., Tu, S., Xiao, C., Haris, M., Alasmary, H., Waqas, M., & Rehman, S. U. (2024). The impact of Domain Name Server (DNS) over Hypertext Transfer Protocol Secure (HTTPS) on cyber security: Limitations, challenges, and detection techniques. *Computers, Materials and Continua*, 80(3), Article 3. <https://doi.org/10.32604/cmc.2024.050049>
- De Clercq, J. (2002). Single Sign-On Architectures. *Infrastructure Security*, 40-58. https://doi.org/10.1007/3-540-45831-X_4
- Duignan, M. (2003). *Evaluating Scalable Vector Graphics for Software Visualisation* [Thesis, Open Access Te Herenga Waka-Victoria University of Wellington]. <https://doi.org/10.26686/wgtn.16915804.v1>
- Eibl, M., & Gaedke, M. (Red.). (2016). *Studierendensymposium Informatik 2016* der TU Chemnitz. Universitätsverlag Chemnitz.
- Font Awesome. (z.d.). Geraadpleegd 3 maart 2025, van [@fortawesome/fontawesome-free. \(2024, december 16\). Npm. https://www.npmjs.com/package/@fortawesome/fontawesome-free](https://fontawesome.com/start)
- Get Started with Material Tailwind—React & Tailwind CSS Components Library. (z.d.). Geraadpleegd 2 maart 2025, van <https://www.material-tailwind.com/docs/react/installation>

gewarren. (2024, januari 30). *Inleiding tot .NET - .NET*. <https://learn.microsoft.com/nl-nl/dotnet/core/introduction>

Gotcha—Scoutpedia.nl. (z.d.). Geraadpleegd 29 maart 2025, van <https://nl.scoutwiki.org/Gotcha>

Heroicons. (z.d.). Heroicons. Geraadpleegd 3 maart 2025, van <https://heroicons.com>

Heroicons. (2024, november 18). Npm. <https://www.npmjs.com/package/heroicons>

Het opzetten van een CI/CD Pipeline met Azure DevOps voor .NET-projecten. (z.d.). Geraadpleegd 3 maart 2025, van <https://garansys.nl/blogs-klantcases/het-opzetten-van-een-cicd-pipeline-met-azure-devops-voor-net-projecten>

Home. (z.d.). GitHub. Geraadpleegd 3 mei 2025, van <https://github.com/codebude/QRCoder/wiki/Home>

How to Play Gotcha. (z.d.). Geraadpleegd 29 maart 2025, van <https://gotcha.game/howtoplay.html>

Introducing JSX – React. (z.d.). Geraadpleegd 9 maart 2025, van <https://legacy.reactjs.org/docs/introducing-jsx.html>

Itext7 9.1.0. (z.d.). Geraadpleegd 3 mei 2025, van <https://nuget.org/packages/itext7>

Kellenberger, K., & Everest, L. (2021). Expanding on Data Type Concepts. In *Beginning T-SQL* (pp. 527-571). Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-6606-9_14

Liao, M., Liu, X., & Jia, T. (2024). Characterizing temporally fragmented human activity networks in cyber space using uniform resource locator (URL) data. *International Journal of Digital Earth*, 17(1), 2295986. <https://doi.org/10.1080/17538947.2023.2295986>

Lucide Icons. (z.d.). Lucide. Geraadpleegd 3 maart 2025, van <https://lucide.dev>

Lucide-react. (2025, februari 28). Npm. <https://www.npmjs.com/package/lucide-react>

Making a Progressive Web App | Create React App. (2022, juni 30). <https://create-react-app.dev/docs/making-a-progressive-web-app/>

Material Design. (z.d.). Material Design. Geraadpleegd 2 maart 2025, van <https://m3.material.io>

@material-tailwind/react. (2024, september 3). Npm. <https://www.npmjs.com/package/@material-tailwind/react>

Math.random()—JavaScript | MDN. (z.d.). Geraadpleegd 18 maart 2025, van https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Math/random

Math.random()—JavaScript | MDN. (2025, februari 11). https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Math/random

mijacobs. (2025, februari 3). Basislijnarchitectuur voor Azure Pipelines—Azure Pipelines. <https://learn.microsoft.com/nl-nl/azure/devops/pipelines/architectures/devops-pipelines-baseline-architecture?view=azure-devops>

@mui/material. (2025, februari 26). Npm. <https://www.npmjs.com/package/@mui/material>

Neumann, C., & Fischer, J. (2023). Extending Bayesian Elo-rating to quantify the steepness of dominance hierarchies. *Methods in Ecology and Evolution*, 14(2), 669-682. <https://doi.org/10.1111/2041-210X.14021>

PaPRIKa, the French Multicriteria Method for Mapping the Intrinsic Vulnerability of Karst Water Resource and Source – Two Examples (Pyrenees, Normandy) | Request PDF. (z.d.). ResearchGate. Geraadpleegd 18 maart 2025, van [https://www.researchgate.net/publication/344361254 PaPRIKa the French Multicriteria Method for Mapping the Intrinsic Vulnerability of Karst Water Resource and Source - Two Examples Pyrenees Normandy](https://www.researchgate.net/publication/344361254_PaPRIKa_the_French_Multicriteria_Method_for_Mapping_the_Intrinsic_Vulnerability_of_Karst_Water_Resource_and_Source_-Two_Examples_Pyrenees_Normandy)

PDFsharp 6.2.0-preview-3. (z.d.). Geraadpleegd 3 mei 2025, van <https://nuget.org/packages/PDFsharp/>

PdfSharpCore 1.3.67. (z.d.). Geraadpleegd 3 mei 2025, van <https://nuget.org/packages/PdfSharpCore/>

Pie Chart | react-chartjs-2. (z.d.). Geraadpleegd 2 maart 2025, van <https://react-chartjs-2.js.org/examples/pie-chart/>

Primereact. (2025a, januari 24). Npm. <https://www.npmjs.com/package/primereact>

Primereact. (2025b, januari 24). Npm. <https://www.npmjs.com/package/primereact>

PrimeReact | React UI Component Library. (z.d.). Geraadpleegd 3 maart 2025, van <https://primereact.org>

Progressive Web Apps. (z.d.). Web.Dev. Geraadpleegd 2 maart 2025, van <https://web.dev/explore/progressive-web-apps>

QRCoder 1.6.0. (z.d.). Geraadpleegd 3 mei 2025, van <https://nuget.org/packages/QRCoder/>

Qrcode.react. (2024, december 11). Npm. <https://www.npmjs.com/package/qrcode.react>

Quasar Framework—Build high-performance VueJS user interfaces in record time. (z.d.). Quasar Framework. Geraadpleegd 2 maart 2025, van <https://quasar.dev/>

LIJST VAN FIGUREN

Figuur 1: data model	9
Figuur 2: use case diagram	11
Figuur 3: basis stijl	14
Figuur 4: navigatie	15
Figuur 5: header	16
Figuur 6: login pagina	17
Figuur 7: spel aanmaken	18
Figuur 8: spel aanmaken met tijd	19
Figuur 9: fisher yates shuffle	20
Figuur 10: home pagina	20
Figuur 11: geselecteerd spel	21
Figuur 12: doods bericht	22
Figuur 13: spelers overzicht	22
Figuur 14: overzicht van een speler	23
Figuur 15: statistieken van een game	24
Figuur 16: gebruikers statistieken	25
Figuur 17: qr code	26
Figuur 18: scan Qr-Code	28
Figuur 19: beheer spel eigenaar	30
Figuur 20: spel regels	31
Figuur 21: beheer admin	32
Figuur 22: elo rating bepalen	34
Figuur 23: pdf met Qr-codes	37

AI PROMPTS

- Zou je dit in dezelfde stijl kunnen herschrijven zoals ik het eerder deed in hoofdstuk 3.16 over Ranked Mode? Graag zakelijk, helder en zonder informele toon.
- Zou je deze alinea kunnen structureren met opsommingstekens, zodat het overzichtelijker wordt?
- Ik wil dit stuk tekst beter laten aansluiten bij de schrijfstijl van mijn vorige hoofdstuk, kun je dat doen?
- Zet dit stuk om in formele scriptiestijl met onderverdelingen per subonderwerp.
- Kun je deze ruwe tekst herschrijven in academische stijl, zoals in een scriptie?
- Kun je deze bulletpoints omzetten in een lopende tekst in formele stijl?
- Kun je deze informele uitleg van de entiteiten en attributen herschrijven zodat het professioneel klinkt?
- Ik heb een paragraaf over de LeaderBoard-component geschreven, kun je dat in heldere en consistente zinsstructuren herschrijven?
- Herschrijf dit stuk zodat het aansluit bij de stijl van een eerdere paragraaf.

POSITIEVE PUNTEN VAN AI-ANTWOORDEN

1. **Consistente schrijfstijl**
Antwoorden volgden een formele en scriptiewaardige toon, wat hielp bij het behouden van een consistente stijl doorheen het document.
2. **Structuur en opbouw**
De AI zorgde vaak voor een logische indeling van informatie in alinea's, stappenplannen of bulletpoints, wat hielp bij het verduidelijken van technische processen.
3. **Herschrijven en verbeteren van tekst**
Teksten die eerst wat rommelig of informeel waren, werden herschreven in een professioneel, academisch formaat.

NEGATIEVE PUNTEN VAN AI-ANTWOORDEN

1. **Soms te algemeen**
In sommige gevallen waren de antwoorden wat te breed of oppervlakkig, waardoor je zelf nog technische details moest toevoegen om het accuraat te maken.
2. **Moeilijk onderscheid tussen uniek en standaard**
De AI beschreef soms elementen alsof ze standaard waren, terwijl jouw implementatie (zoals de target-mechaniek of Free For All logica) uniek was. Dit kon verwarringen veroorzaken.
3. **Weinig creativiteit bij herhaling**
Wanneer je meerdere prompts gaf over hetzelfde onderwerp (bv. leaderboard of game modes), gaf de AI soms herhalende of gelijkvormige antwoorden, wat minder fris oogde.
4. **Nood aan fact-checking**
De antwoorden moesten altijd gecontroleerd worden op juistheid, zeker als ze over programmeerlogica of architectuur gingen. De AI kon aannames doen die niet klopten met jouw project.
5. **Mist soms context**
Zonder voldoende achtergrond (zoals hoe jouw game echt werkt), vulde de AI soms zelf aannames in die niet klopten met je implementatie.