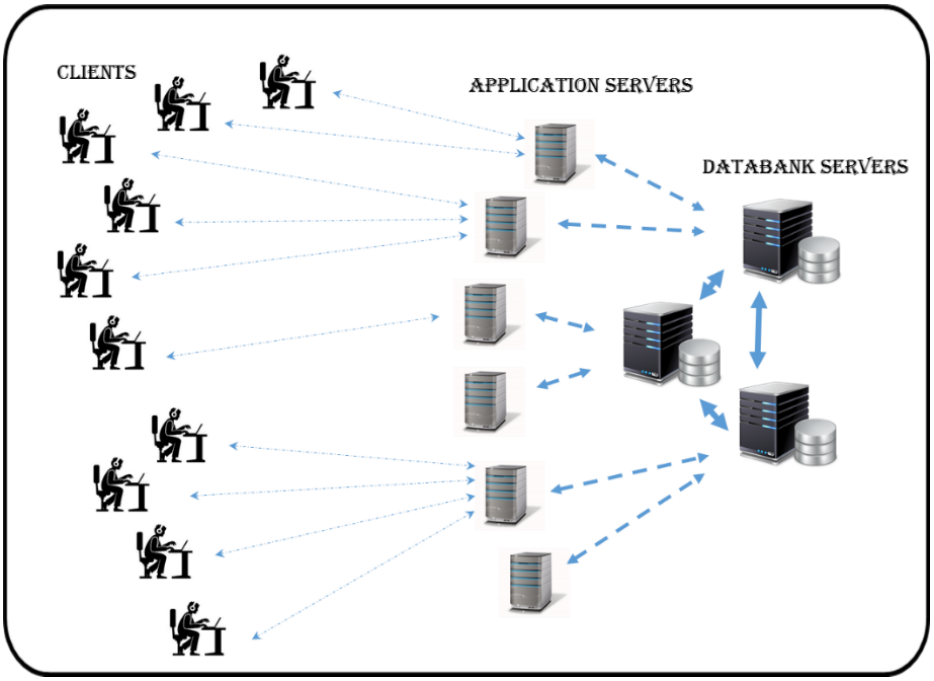
Opdracht Gedistribueerde Systemen (2017-2018)

Implementeer een gedistribueerd systeem dat spelers (of clients) toelaat om een eenvoudig spel te spelen, namelijk het kaartspel “UNO”. Elke speler registreert zich eerst bij de server. Bij registratie kunnen spelers een login en paswoord aanmaken. Na registratie kan een speler zich aanmelden. Tijdens het aanmelden geeft de speler zijn login en paswoord in, en krijgt een sessietoken terug. Dit is een uniek random getal dat een gebruiker gedurende zijn sessie kan gebruiken. Het token blijft 24 uren geldig, of tot de gebruiker zich expliciet afmeldt. Na het aanmelden kan hij spelletjes spelen. Hiervoor wordt door de gameservers een lobby bijgehouden. De lobby bevat alle spellen die reeds zijn aangemaakt, al dan niet reeds gestart. In de lobby kan de gebruiker ervoor kiezen om een nieuw spel aan te maken, of om deel te nemen aan een bestaand spel (waar nog een plaats vrij is). Bij het aanmaken van een nieuw spel kan een speler aangeven tegen hoeveel andere spelers hij het wil opnemen. Heel concreet kan hij deelnemen aan een spel van 2, 3 of 4 spelers. Bovendien kan een speler meekijken naar spelletjes die tussen andere spelers aan de gang zijn. Voor de implementatie van het UNO spel wordt de volgende basis functionaliteit verwacht:

- Alle normale kaarten (kleuren en cijfers) + minstens 2 speciale kaarten (volgorde omwisselen, volgende speler overslaan, kleur kiezen, kleur kiezen en laat tegenstander 4 kaarten trekken, laat tegenstander 2 kaarten trekken)
- Na elk spel wordt er ook een scorebord bij gehouden. De verkregen score wordt bepaald door de kaarten die de tegenstander nog in de hand heeft.
    - Normale kaarten (0-9) → punten gelijk aan cijfer op de kaart
    - 2 trekken, volgorde omdraaien, speler overslaan → 20 punten
    - 4 trekken, kleur kiezen → 50 punten
- Tijdens speciale gelegenheden zoals feestdagen worden de kaarten versierd. Voorzie functionaliteit om bij deze gelegenheden speciale versies van de kaarten weer te geven.

Bij de uitvoering van de opdracht wordt Java RMI gebruikt als middleware communicatietechnologie. Een aantal andere middleware services zullen zelf geïmplementeerd moeten worden met het oog op schaalbaarheid, performantie en betrouwbaarheid.

Architectuur. De architectuur moet minimaal aan de volgende voorwaarden voldoen:



- Een beperkt aantal databank servers (typisch een 3 of 4 tal). Deze servers sturen een databank aan. De databank bevat eerst en vooral gegevens die op meerdere spellen van toepassing zijn, zoals de overall ranking van de spelers. Ten tweede bevat de databank gegevens van spelers: login en paswoord, het huidige sessietoken, … Ten derde wordt info omtrent alle spellen in een lobby bijgehouden. Merk op dat de persistente gegevens op alle replica's worden bijgehouden. Het doel van de replicatie is enerzijds de performantie te verhogen, en bij uitval van 1 server toch nog te kunnen functioneren. Elke databank server heeft connectie met een lokale kopie van de databank. De databank zelf mag met SQLite technologie worden uitgebouwd. De databank servers halen enkel data op uit hun lokale kopie, en communiceren enkel met de application servers, en niet rechtstreeks met de clients. Vanzelfsprekend kunnen ze onderling ook interageren om een welbepaald consistentiemodel te realiseren.
- Een variërend aantal application servers. Per 20 actieve spellen wordt een nieuwe application server opgestart. Zo blijft het systeem performant, en ondervinden gebruikers weinig vertragingen. We veronderstellen dat voldoende hardware ter beschikking is om nieuwe application servers op te starten. Indien spelers verdwijnen kan opnieuw beslist worden om een server te sluiten, en spellen te herorganiseren. Bij het sluiten van een server kunnen spellen die aan de gang zijn op een andere server geplaatst worden. De application servers zijn de front-end naar clients, en ondersteunen de afhandeling van acties die door gebruikers zijn geïnitieerd. De application servers kunnen gegevens uit de databank cachen. Na opstart worden ze gelinkt aan één specifieke databank server die ze gedurende hun ganse lifetime consulteren. Na het uitvallen van zo een databank server kunnen ze gekoppeld worden aan een andere.
- Dispatcher. De dispatcher koppelt gebruikers na opstart met een bepaalde application server.

- De clients. Elke speler kan een client opstarten. Dit is het programma dat door een gebruiker geïnstalleerd wordt. De client software communiceert met de dispatcher en een application server, dus niet rechtstreeks met de databank servers.

De opdracht bestaat uit het ontwerp en de realisatie van een gedistribueerd systeem voor deze killer applicatie. Besteed de nodige aandacht aan de volgende ontwerpbeslissingen:

- **Ontwerp van de databank.** Een doordachte organisatie van de databank.
- **Definitie van APIs.** De definitie van een goede API voor zowel application servers, de databank servers en de dispatcher. De API bepaalt de methodes en parameters die kunnen opgeroepen worden.
- **Consistentie modellen en replicatie.** Selectie van een goed consistentie model en realisatie van een consistentie protocol bij de databank servers. De realisatie vereist dat databank servers onderling zullen communiceren met elkaar. Definieer hiervoor eveneens een goede API.
- **Caching.** Selectie en realisatie van een goede caching strategie bij zowel de application servers als de clients. De application servers cachen gegevens uit de databank. De clients cachen ook gegevens bij de spelers (zoals kaarten, scores…).
- **Aandacht voor beveiliging.** We veronderstellen dat de communicatiekanalen veilig zijn (integrity protected en confidentieel), en dat beide partijen bij communicatie elkaar hebben geauthenticeerd tijdens het opzetten van communiatiekanalen. Je hoeft daar dus geen aandacht aan te besteden. Wel is het belangrijk dat een mechanisme wordt geïmplementeerd om paswoorden en tokens veilig op te slaan.
- **Recovery.** Ga na in hoeverre uitval van machines kan leiden tot verlies van gegevens, of het crashen van spellen. Is het mogelijk om het systeem terug in een consistente toestand te brengen na het crashen van een client machine, een application server of een databank server? Op welke manier kan het systeem in dergelijk geval terug in een consistente toestand worden gebracht? De crash en recovery strategieën moet je niet implementeren.

Opleveringen. De opdracht wordt in groepjes van 2 studenten uitgevoerd en de toepassing wordt uitgebouwd in twee iteraties.

Iteratie 1. In een eerste iteratie wordt de toepassing ontwikkeld zonder gebruik te maken van redundantie bij applicatie servers en databank servers. Dit kan reeds na labo 2. Van deze toepassing wordt de code opgeleverd via een link in email op 7 november 2017, en een demo gegeven op 9 november 2017. Per groep wordt 15 minuten voorzien (1/3 demo en 2/3 vragen en suggesties).

Iteratie 2. Daarna wordt het ontwerp van de toepassing herzien met het oog op redundantie (replicatie en consistentie), performantie (caching, server-side replication) en beveiliging (authenticatie en autorisatie). De finale oplevering wordt ingediend op 19 december 2017. Daarbij wordt het volgende opgeleverd:

- Een rapport met het ontwerp van de toepassing. Dit ontwerp omvat:
    - Een beschrijving van de architectuur [1 pagina]
    - Een beschrijving van de databank [1 pagina]
    - Een toelichting bij de belangrijkste ontwerpbeslissingen op vlak van (a) consistentie en replicatie, (b) caching, (c) beveiliging en (d) recovery [2 pagina's]
    - Een overzicht van de API van de dispatcher, de application server, en de database server, en documentatie over de methodes en parameters [afhankelijk van grootte van API]
    - Kritisch relflectie, met analyse van sterktes, zwaktes en mogelijke uitbreidingen [1 pagina]
- De code en documentatie om de toepassing op te starten.

Beiden worden via een link ter beschikking gesteld. De link wordt gemaild naar:
vincent.naessens@kuleuven.be

Demonstratie. De finale versie van het practicum wordt gedemonstreerd op 21 december 2017. Hierbij wordt het volgende verwacht:

- Slide presentatie [max. 15 slides – max. 20 minuten] die de belangrijkste bijdrages van de toepassing in de verf zet:
    - Overzicht van de functionaliteit en architectuur
    - Overzicht van de belangrijkste ontwerpbeslissingen op vlak van van (a) consistentie en replicatie, (b) caching, (c) beveiliging en (d) recovery
    - Enkele relevante implementatiedetails
    - Kritische reflectie
- Een demonstratie van de toepassing
- Beantwoorden van vragen

Voor elke groep wordt 40 minuten gereserveerd.