Systemintegration

TRACK OR BE TRACKED

Frey Clante | Systemintegration | 22-05-2024

Indholdsfortegnelse

INDLEDNING

Overvågning og tracking på internettet er en af nutidens hotte emner, der fylder mere og mere i folks bevidsthed, men de færreste gør noget ved det fordi de færdigheder og den viden der skal til for at afdække denne til tider ulovlige tracking for menigmand opfattes som ren magi.

Baggrund

TV 2 – dokumentaren "Spionerne bag skærmen" fra maj 2023 satte fokus på hvor meget data der indsamles via mobilspil til børn og hvordan industrien bevidst forsøger at undgå at deres spil klassificeres som indhold til børn fordi de dermed rammes af en mængde af begrænsninger i forhold til hvad de må høste af oplysninger.

Formål

Grundet netop manglen på gennemsigtighed ift. hvilke data og andre informationer der deles med Tech-Virksomhederne har en ildsjæl, der til daglig arbejder som Software Udvikler i en større unavngiven virksomhed, efter at have se TV2 – dokumentaren "Spionerne bag skærmen", fået en idé til en løsning der skal gøre det muligt for folk at se hvad der deles i nær realtid fra deres mobil-telefoner.

Udvikleren har en idé om at en MVP bør indeholde et simpelt user-interface med mulighed for at se egne data der strømmer ud, samt at se data fra andre, der har indvilliget i dette f.eks. som en Buddy-tracker for folk der gerne vil sikre sig at deres venner eller børn når sikkert frem eller hjem efter en tur i byen.

PROBLEMFORMULERING

Hovedspørgsmål

Hvordan integreres dataindsamling på en mobiltelefon med andre mobiltelefoner, så disse kan aflæses i nær realtid?

Underspørgsmål

- Hvordan håndteres manglende mobildækning og caching af udgående data indtil der igen kan opnås stabil forbindelse så data kan afsendes med stor leveringsgaranti?
- Hvordan sikres det at datasikkerheden opretholdes og uvedkommende ikke kan sniffe data.

Afgrænsning

Jeg vil i rapporten stille skarpt på integrationen imellem klient og Azure Eventhub.

Der vil ikke blive implementeret *"Enterprise grade authentication"*.

OPGAVENS UDFORMNING

Løsningen kan opdeles i følgende dele

- Androind Client App
    - Indsamler events fra diverse kilder
        - I denne løsning har jeg valgt at begrænse det til lokationsdata
    - Implementeret i Kotlin
- .NET Azure Function
    - Har ansvar for at generere en SAS-Token til at få adgang til Azure Eventhub (read/write)
    - Denne integration er et synkront Web-API
- Azure Eventhub
    - Data broker
    - Anvender en Storage Account til at gemme metadata, historik osv.
    - Lav retention time (1 time) da lokationsdata stort set er for gamle når de modtages

EMNEBEHANDLING

Integrationstyper

Messaging:

Beskrivelse: Afsendere (publishers) sender meddelelser til en kanal, og modtagere (subscribers) lytter til kanalen for at modtage meddelelser.

Fordele: Decoupling mellem afsendere og modtagere, hvilket giver høj fleksibilitet.

Ulemper: Mere kompleks fejlhåndtering og sværere at sikre data levering.

Anvendelse Af Patterns

Event Streaming Pattern

Azure Event Hubs bruges til at streame store mængder data i realtid. Dette pattern tillader kontinuerlig indsamling og behandling af data fra forskellige kilder, som IoT-enheder, applikationslogfiler, og social media feeds.

Partitioning Pattern

Data opdeles i partitioner for at opnå parallel databehandling. Dette pattern er implementeret i Azure Event Hubs ved at bruge partitioner til at fordele belastningen og forbedre skalerbarheden.

Competing Consumers Pattern

Flere forbrugere læser fra samme event stream for at øge systemets gennemløb. Azure Event Hubs understøtter dette pattern ved at tillade flere forbrugere at læse fra forskellige partitioner samtidig.

Mulige Teknologier til Implementering

Azure Eventhub

- Skalerbarhed Azure Event Hubs er designet til at håndtere millioner af events per sekund, hvilket gør det ideelt til at indsamle store mængder data.
- Kompatibiliet Integration med andre Azure-services såsom Stream Analytics, Azure Functions og Data Lake.
- Sikkerhed**:** Understøtter Managed Identities, og SAS tokens.
- Simpel Administration Azure Eventhub er en PaaS (Platform as a Service) løsning, som kræver minimal administration og vedligeholdelse.

RabbitMQ

Fordele: Fleksibel og understøtter forskellige messaging protokoller (AMQP, MQTT, STOMP).

Ulemper: Ikke så skalerbar som Azure Event Hubs, kræver mere manuel konfiguration og administration.

ZeroMQ

Fordele: Høj ydeevne og lav latenstid.

Ulemper: Det er et messaging library og ikke en server, hvilket kræver mere manuel styring af messages og netværkskommunikation.

Kafka

Fordele: Høj skalerbarhed og pålidelighed, velegnet til realtids streaming og log-håndtering.

Ulemper: Mere komplekst at administrere end Azure EventHub, kræver desuden opsætning og vedligeholdelse af cluster-infrastruktur.

Implementering

Valget faldt på Azure Event Hubs fordi:

Skalerbarhed og Pålidelighed: Azure Event Hubs er designet til at håndtere millioner af events per sekund, hvilket gør det ideelt til applikationer, der kræver høj ydeevne og pålidelighed.

Integration: Nem integration med andre Azure Services gør det muligt at anvende andre Azure-services til dataanalyse, lagring og visualisering i fremtiden.

Sikkerhed: Azure Event Hubs tilbyder avancerede sikkerhedsfunktioner, herunder Virtual Network integration, Managed Identities og Shared Access Signatures (SAS) tokens. Disse er relativt *"short-lived" og det var et bevidst valg i.fht. tid og hensynet til at andre, der ikke har adgang til min subscription i Azure skal kunne bruge appen. Derfor er det også lavet som anonymt auth.*

Client Libraries til Java/Kotlin

Azure Event Hubs tilbyder client libraries til Java/Kotlin, hvilket gør det nemt at integrere med eksisterende applikationer udviklet i disse sprog.

Disse libraries håndterer forbindelser, batch processing og Exceptions, hvilket forenkler udviklingsprocessen.

Token Håndtering med Azure Function i C#

Azure Function er implementeret i C# for at generere SAS tokens til Authentication og Authorization.

KONKLUSSION

Problemformuleringens spørgsmål føler jeg er besvaret, men implementeringen af bl.a. event caching i tilfælde af forbindelsessvigt på det mobile Device udestår.

Opgaven var spændende og jeg kan kun ærgre mig over ikke at være gået i gang noget før. Undskyldningerne er mange men nogle gange kommer livet i vejen. Jeg vil forsøge at færdiggøre projektet til eksamen. Jeg er godt klar over at min rapport ikke er meget værd.

LITTERATURLISTE

- Enterprise Integration Patterns Hophe/Wolf – 2004
- About Eventhubs → https://learn.microsoft.com/en-us/azure/event-hubs/event-hubs-about

BILAG

- Arkitektur Overblik - https://github.com/fclante/system-integration/blob/main/docs/image/Architecture.png
- Azure Eventhub Arkitektur Overblik - https://github.com/fclante/system-integration/blob/main/docs/image/event-streaming-platform.png
- Source Code → https://github.com/fclante/system-integration