FORPROSJEKT

Transaksjonsmonitor for PayEx

Ali Arfan s301599
John Husfloen Håvardstun s305053
Kent Erlend Bratteng Knudsen s300358
Shamil Magomadov s305052

Gruppe 19 | Høgskolen i Oslo og Akershus

Presentasjon

Oppdragsgiver	PayEx Norge AS, Kongens Gate 6, 0153 Oslo
Kontaktperson:	Dani Alexander Berentzen, Teamleder POS Server dani.alexander.berentzen@payex.com
Prosjekttittel	PayEx transaksjonsmonitor
Oppgave	Erstatte dagens monitor over gjennomførte betalinger. Betalingene foretas av kunder som bruker PayEx sine betalingsløsninger.
Bedriftsnettside	http://payex.no/
Gruppe 19	Ali Arfan | s301599 John Husfloen Håvardstun | s305053 Kent Erlend Bratteng Knudsen | s300358 Shamil Magomadov | s305052
Veileder	Torunn Gjester https://www.hioa.no/tilsatt/Torunn%20Gjester
Gruppenettside	:

Oppdragsgiver

Swedbank PayEx er et nordisk konsern med cirka 500 medarbeidere i fire land. PayEx har heldekkende betalingsløsninger for e-handel, mobile plattformer og fysisk handel, og mange typer tjenester innen rating og billing, faktura- og reskontrohåndtering, inkasso, fysisk og elektronisk betaling, kreditthåndtering og finansiering og lojalitet.

Gruppe

Vi er en gruppe på fire personer som alle studerer til Dataingeniør ved HIOA. Vi har jobbet mye sammen i gruppe og har erfaring med blant annet: databasedesign og -utvikling, Java og webdesign. Kent har hatt valgfaget Webapplikasjoner der de benyttet .NET og Angular, mens de tre andre hadde Apputvikling der de lærte å utvikle apper for Android-plattformen. Shamil har fordypet seg i Matematikk 3000, mens de tre andre valgte Datasikkerhet.

Sammendrag

I dette prosjektet har vi fått i oppgave å implementere en ny frontend- og backend-løsning som skal erstatte den eksisterende POS-monitoren PayEx bruker i dag. Den nye løsningen skal være en webapplikasjon som skal kunne vise transaksjonsstatistikk og annen relevant informasjon i real-time samt være tilgjengelig for flere brukere.

Frontend vil utvikles med JavaScript der vi bruker rammeverket Vue. Backend vil benytte seg av AMQP og InfluxDB med mulighet for tidsserieanalyser. Autentisering og autorisering vil skje mot Active Directory.

Det blir lagt opp til Scrum utviklingsmetodikk med sprinter på 2 uker hvor vi vil bruke JIRA for å organisere backlog og sprinter. GitHub vil bli brukt under utviklingsprosessen for versjonskontroll av systemet.

Dagens situasjon

For å se antall POS (Points of Sale) transaksjoner som har blitt utført i løpet av dagen, benytter PayEx seg av en egenutviklet løsning de har døpt PosPay Monitor. Denne applikasjonen kan vise gjennomsnittlig antall transaksjoner per sekund, hvor lang tid siste transaksjon tok, hvor lang tid som har gått siden siste transaksjon ble utført og totalt antall utførte transaksjoner, samlet eller per kunde, samt noe generell ytelsesinformasjon og ressursbruk.

Ulempen med dagens løsning er at den mangler mekanismer for brukerautentisering og -autorisering, og kan derfor ikke brukes til å filtrere data basert på tilgang. Dette begrenser mulighetene PayEx har for å tilgjengeliggjøre applikasjonen for flere brukere, eller å kunne tilby den som en tjeneste til kundene sine. Av den grunn må man i dag være pålogget det interne nettverket til PayEx for å kunne bruke applikasjonen.

Fordi PosPay Monitor ble utviklet som en skrivebordsapplikasjon, er det tungt å tilpasse applikasjonen andre typer plattformer og enheter. Nettopp derfor ønsker PayEx at vi skal utvikle en web-basert løsning sammen med et API som også gjør det mulig å enkelt utvikle andre typer klienter, som apps, til løsningen frem i tid. Det er også ønskelig fra PayEx sin side at løsningen skal kunne hente innhold fra flere kilder for å kunne vise f.eks. eCommerce i fremtiden, og være fleksibel med tanke på å vise nye kunder, samt kunne gi varsler ved avvik i transaksjonsbildet.

Dagens løsning bruker en monitorserver som mottar transaksjonsdata fra transaksjonsserverne i form av UDP-pakker, og sender data videre til klientene, som viser de løpende transaksjonene, også i form av UDP. En av ulempene med UDP-protokollen er at man ikke er garantert at alle pakker kommer frem eller at de kommer i rekkefølge.

Mål og rammebetingelser

Mål

Hovedmålet med oppgaven er å erstatte dagens transaksjonsmonitor som brukes av teamet ”POS Server”. Denne monitoren viser informasjon om hvor mange betalinger som til enhver tid prosesseres, og viktig ytelsesinformasjon. Den nye løsningen skal være en webapplikasjon, og vise transaksjonsstatistikk og annen relevant informasjon i real-time eller så nært dette som mulig. Produktet skal være en webapplikasjon.

Fra PayEx har vi fått en liste over innledende krav og ønsker:

- Ha en reaktiv frontend, slik at dataene blir tegnet så fort som mulig i browser med minst mulig overhead
- Separert frontend fra backend (muliggjør andre typer klienter senere)
- High performance serialisering (protobuf eller andre binære protokoller)
- Frontend må kunne vise ulik data basert på rettighetsnivå og/eller egne brukerpreferanser
- Må ikke påvirker systemene som overvåkes (ikke kunne ta ned/påvirke systemene unødvendig negativt)
- Må være real-time, slik at man kan se trender ettersom de oppstår
- Må kunne være tilgjengelig over offentlig internett, gjennom sikre kanaler (TLS, autentisering/autorisering)
- Så lite state i backend som mulig. Dette muliggjør at man kan spinne opp nye instanser av backenden hvor som helst hvis en eller flere instanser dør.
- Ha et godt UI/UX, slik at løsningen faktisk kan brukes
- Gi muligheten til å gi et godt oversiktsbilde når man tar et kort kikk (<5 sek) på monitoren
- Hvis mulig: Mulighet for å sette opp notifikasjoner for gitte events som man kan definere. F.eks. hvis det tar over N millisekunder å prosessere en transaksjon, send en SMS til vakttelefonen, eller hvis innløser X har lang svartid, send et varsel på Slack.

Bortsett fra de innledende kravene står vi fritt til å velge fremgangsmåte, utviklingsspråk og design, men de skal godkjennes av PayEx før vi setter i gang med utviklingen.

Løsninger

Løsningen er en webapplikasjon med moderne backend og frontend. I utviklingsprosessen vil vi bruke GitHub for versjonskontroll av systemet. Ved bruk av GitHub vil vi kunne holde styr på forandringer i systemet, slik at vi kan finne tilbake til tidligere versjoner av systemet, dele endringer mellom gruppemedlemmene og hvis det skjedde en feil i systemet kan vi sjekke hvilken endringer som forårsaket dette.

Frontend

Frontend er det brukeren ser når han besøker websiden.

- Designet skal være moderne, responsivt og reaktivt.
- Skal kommunisere sikkert med backend.
- Vise grafer, data og status på transaksjonene som foretas.
- God oversikt med et kjapt blikk, men:
- Mulighet for å se mer detaljert informasjon om de forskjellige gruppene.
- Autentisering og autorisering, med visning basert på dette.

Programmeringsspråk, teknologier og rammeverk

- HTML5
- CSS
    - Muligens rammeverk - skal diskuteres nærmere med UI/UX-eksperter hos PayEx.
- JavaScript
    - AJAX
    - JSON
    - Vue.js
- GitHub for versjonskontroll.

HTML er et deklarativt “markup language” som brukes for å strukturere websider. At det er deklarativt betyr at vi beskriver hva vi ønsker vise (f.eks. en liste eller bilde), men ikke hvordan, og mangler derfor mye funksjonalitet for å håndtere data og skape interaktivitet. For å skape en interaktiv side med oppdateringer i real-time kommer vi til å bruke JavaScript, og for å definere utseendet vil vi bruke CSS.

JavaScript er det dominerende programmeringsspråket på klientsiden (nettlesere) på internett. Uten muligheten til å kjøre logikk på klienten, ville det blitt veldig vanskelig å få til en tilfredsstillende løsning; med JavaScript kan vi løpende hente oppdateringer og visse disse.

Vue.js eller bare Vue er et enkelt og effektivt rammeverk i JavaScript for å bygge store og skalerbare webapplikasjoner. I motsetning til de fleste andre rammeverk, er Vue bygget opp fra grunnen av til å være enkelt å adoptere trinnvis. Det vil si at det er enkelt å benytte side om side med andre biblioteker eller i eksisterende prosjekter. Med de valgfrie kjernebibliotekene er Vue også veldig godt egnet til utvikling av SPA-er (Single Page Application) - websider som oppdaterer og bytter ut deler av seg selv i stedet for å laste ned en helt ny side fra serveren.

Backend
Backend er den delen av tjenesten som ikke kjører hos klienten (nettleseren). I backend vil det ligge en database med transaksjonsdata og tjenester som skal håndtere bl.a. autentisering, autorisering og kommunikasjon med klientene.

Programmeringsspråk, teknologier og rammeverk

- InfluxDB
- Java
    - Spring Boot
- NGINX
    - TLS med Let's Encrypt
    - Autentisering mot Azure Active Directory (LDAP)
- AMQP for filtrering og sending av data.
    - RabbitMQ

Backend skal hoste alle tjenestene som klientene kobler seg til. Det er da viktig at det er stabilt og data ikke forsvinner. Den eksakte serverarkitekturen er ikke satt, men hvis vi trenger en generisk webserver vil vi bruke NGINX, da dette er en stabil, høytytende, fleksibel og velprøvd webserver. Klientene får transaksjonsdata fra en monitorserver som er skrevet ved hjelp av Java-rammeverket Spring. Dette rammeverket er et solid rammeverk som gjør det lettere å håndtere avhengigheter. Du kan enkelt redusere koblingene mellom komponentene for å få et mer robust system. Du kan da koble til nye komponenter senere eller bytte dem ut. Det som kan være ulempen med Spring er at mye må deklareres i XML-filer, noe som kan føre til lite oversiktlighet, men dette har blitt forbedret i nyere versjoner.

I forarbeidet med oppgaven ble vi presentert med to muligheter for å få transaksjonsdata:

1. **Fra en database**

Transaksjonsdataene lagres i en tidsseriedatabase, som er spesielt egnet til å lagre og behandle tidsbaserte data som f.eks. hendelser, og tilbyr funksjoner spesielt for analyse av tidsserier. Da kan vi finne ut for eksempel når på dagen det er flest transaksjoner og tilrettelegge for avviksanalyse av dataene.

2. **Fra en meldingskø (RabbitMQ)**

RabbitMQ er en implementasjon av AMQP (Advanced Messaging Queuing Protocol), en protokollspesifikasjon for å sende meldinger i applikasjonslaget som tilbyr køer, pålitelighet og sikkerhet. Meldingene sendes asynkront, og hver melding har en id, som sørger for at alle meldingene kan bekreftes mottatt. AMQP har et køsystem der flere mottakere kan ta imot meldinger i eget tempo, og sørger for at alle meldingene blir levert til alle som abonnerer på dem. Hver gang en melding blir send til AMQP (her: transaksjonsdata) blir meldingen lagt i køene og sendt så fort som mulig; dette gjør at dataene blir sendt i tilnærmet real-time, noe som gjør at AMQP kan være kjappere enn å hente data fra databaser basert på intervaller.

Det er nødvendig med autentisering og autorisasjonskontroll slik at bare de som skal se dataene kan se dem. Da kan man også skreddersy innholdet for brukere basert på roller eller grupper. Slik vi har tenkt å løse dette er å autentisere mot Active Directory, der ansatte logger inn hos PayEx. De vil da bli gruppert på hvilke rettigheter den tilhørende gruppen(e) til brukeren har. Når transaksjonsdata skal vises på monitoren kommer den til å bli sendt ut til alle gruppene som har rettigheter til å se denne.

Analyse av virkninger

Ved at transaksjonsmonitoren blir skrevet om som en webapplikasjon, tilbyr vi PayEx en mer moderne og praktisk løsning som kan lettere videreutvikles (apps o.l.) og vedlikeholdes. Transaksjonsmonitoren kan også ta i bruk andre datakilder hvis nødvendig.

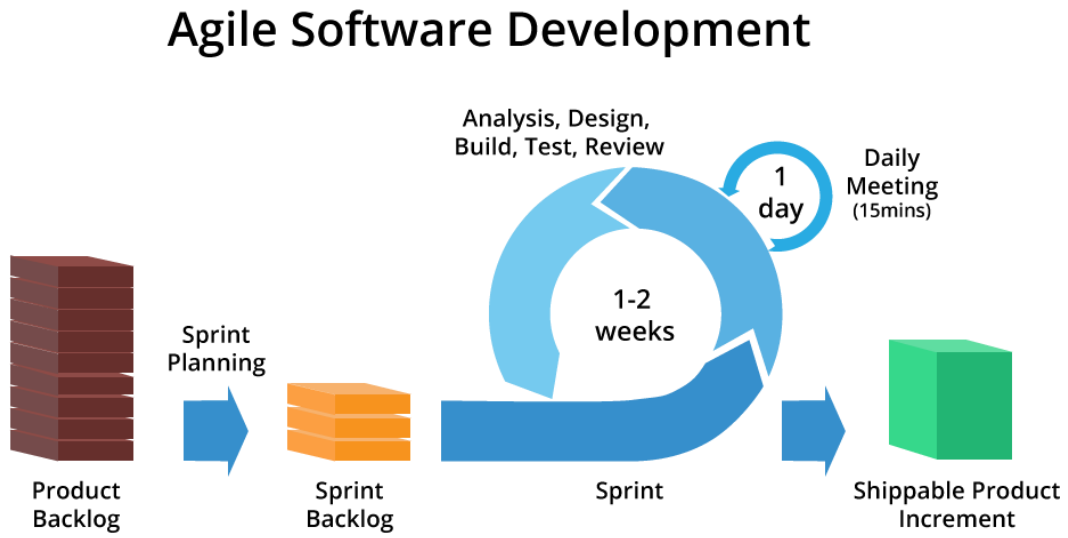
(Raskere) varsling: Ved å hente ut historiske transaksjonsdata fra de forskjellige kundene, kan vi generere statistikk om hvor mange transaksjoner som har blitt utført sist uke, dag eller måned. Ved å sammenligne øyeblikksbildet med statistikken, muligens ved hjelp av maskinlæring, kan vi lage et varslingssystem som sender ut varsler når det er avvik i nåsituasjonen. Sammenlignet med tidligere løsning, som var at man selv måtte følge med og vurdere statistikken på monitoren, vil denne løsningen føre til raskere varsler.

Autentisering og autorisering: Ved hjelp av autentisering vil vi beskytte systemet fra fremmede og lar bare de kundene eller ansatte ved PayEx som har korrekte rettigheter bruke systemet. Ved autorisering vil ansatte eller kunder kun få tilgang til de ressursene som de har rett til. For å sørge for at sikkerheten ivaretas når systemet nå skal være tilgjengelig utenfra det interne nettverket til PayEx, vil all kommunikasjon til backend krypteres med TLS.

RabbitMQ vs Databaser: Valget står mellom å bruke RabbitMQ for å transaksjonsdataene eller koble direkte til en database og hente ut data som trengs. Dette skal fastspikres av PayEx, og i skrivende stund lener de seg mot bruk av RabbitMQ.

Arbeids- og fremdriftsplan

Det er fastsatt at vi skal sitte hos PayEx å jobbe med prosjektet tre dager i uken, eventuelt fire ved behov. Vi vil benytte oss av en smidig Scrum-utviklingsmetodikk for prosjektet der vi planlegger, designer, utvikler, og evaluerer. Det blir lagt opp til sprinter på to uker der vi tar for oss overkommelige oppgaver og lager et helhetlig inkrement i løpet av denne tiden. Vi skal bruke kommunikasjonsplattformen Slack for å holde kontakt utenom fysisk tilstedeværelse.



Product backlog: Alle oppgavene som ønskes gjort for å utvikle systemet.
Sprint backlog: Oppgavene som skal prøves å ferdigstilles i løpet av sprint
Sprint: Jobbe med oppgavene i 2 ukers tid.
Shipable Product increment: Ferdigstilt produkt som har blitt evaluert og godkjent.

Vi skal bruke JIRA som et hjelpemiddel for å holde orden på planleggingen. Ved bruk av Jira kan vi:

- Sprint planlegging
- Brukerhistorier
- Distribusjon av oppgaver til gruppemedlemmer
- Prioritering av oppgaver.

Månedsplan:

Periode	Arbeid
Januar	Planlegging, styringsdokumenter
Februar - Mai	Utvikling & dokumentasjon
Mai	Levering av bachelor rapport
Slutten av Mai - Juni	Presentasjon

Vedlegg

[1] Skisse over hvordan systemet kan se ut.