Bitweb fullstack dev proovitöö

Teie poolt saadetud proovitöö oli väga hea proovikivi. Kuna ma pole kunagi sellise arhitektuuriga rakendust varem ehitanud, siis võttis see mul natuke aega. Mis minu silmis antud ülesandele hea keerukuse juurde lisas oli see, et pidevalt serveri poolset rakendust ehitades pidi mõtlema seda, et kuidas rakendus toimiks, siis kui näitkes Core ja Worker rakendusi töötab mitu tükki ning seda nii, et statistika loomine rakendusel sassi ei läheks.

Serveri poolne rakendus

Serveri poolse rakenduse loogika on lihtne. Kasutaja laeb üles teksti faili, teksti töötlemiseks faili sisu jagatakse 2000 reakaupa osadeks ning siis saadetakse edasi teenusele, kus teenus leiab ühest osast kõik sõnad ning sisestab need andmebaasi.

Sõnadest statistika tegemine oli paras peavalu, sest ma ei suutnud leida head viisi kuidas statistikat hoiustada. Kuna tööd tehes pidasin silmas ka seda, et Worker teenuseid võib olla mitu tükki, siis ei saanud sellise lähenemisega teha, et Worker muudab pidevalt tekstis esinevate sõnade sagedust andmebaasis. Seda sellepärast, et kui mitu Worker teenust korraga ühte teksti faili töötlevad, siis võib juhtuda näiteks olukordi kus näiteks kaks teenust samal ajal muudavad ühe sõna sagedust ning see tähendab, et sellega võib kaotsi minna mitu sõna. Seega Worker teenus töötleb teksti ära ning sisestab iga sõna tabelisse koos teksti faili unikaalse id-ga. Lahendus pole hea, aga antud juhul oli minu silmis see kõige parem variant. Selline lahendus lisaks võimaldab ka teksti töötlemise teenust jooksutada asünkroonselt. Hetkel teenus jooksutab MQ sõnumite kuulamist ja teksti töötlemist kolmel threadil.

Core rakendus jälgib kui palju sõnumeid on Worker teenus ära töötlenud. Enne kui Worker enda tööd alustab, loob Core andmebaasi kirje mis sisaldab teksti faili unikaalset id-d, mitmeks osaks tekst jaotati ning mitu osa on töödeldud. Kui kõik osad on töödeldud, siis Core saadab andmebaasi päringu mis laseb andmebaasil luua teksti

statistika ning sisestada tulemuste tabelisse kõik sõnad koos sageduse ja teksti unikaalse Id-ga. Pärast seda tühjendatakse tabel (hoiustab kõiki teksti faili sõnu) sõnadest mis on eelnevalt töödeldud teksti failiga seotud. Pärast seda on võimalik kasutajal teha läbi Core rakenduse päringuid teksti statistika kohta.

Kliendi poolne rakendus

Kuna serveri poolse lahenduse ehitamine võttis mul peaaegu 2 nädalat aega, siis kliendi poolse rakenduse loomiseks ei jäänud väga palju aega (päev või kaks). Loodud rakendus võimaldab kasutajal teksti fail üles laadida, panna kirja sõnu mida ta ei taha, et statistikas kajastuks ning valida kas ta soovib leida kõik võimalikud kirjavead. Pärast üleslaadmist saab kasutaja sisestada üleslaetud faili unikaalse id ning filtreerida, et millises vahemikus sõnade arvud võiksid sõna pilves olla. Sõna pilvel hiirega üle käies on kasutajal võimalik näha, et mis sõnaga tegemist on ning mitu korda sõna tekstis esineb.

Kliendi poolsel rakendusel kindlasti saab parandada Error handlingut. Lisaks raputan endale tuhka pähe sellega, et mu töös puuduvad testid. Tahtsin kindlasti esitada õigeks ajaks ning kui ma oleks teste kirjutama praegu hakanud, siis oleks arvatavasti töö valmis saanud järgmine nädal. Sellega seoses proovin ennast kindlasti parandada.

Kokkuvõttes oli väga hea prooviülesanne. Olen kindel, et minu loodud lahendus pole perfektne, sest pidevalt rakenduse arenduse käigus oli see mõte, et kas see on ikka kõige parem lahendus. Samas loodan, et antud lahendus aitab mind teie juurde juunior arendajaks saada ning juurde saada kogemusi ja tarkusi selliste süsteemide arendamisel.

Markkus Koodi