Implementation av Elasticsearch i Lina

Utformandet av en prototyp med fokus på att effektivisera sökningar och förkorta svarstider

Examensarbete inom Data- och Informationsteknik

KATARINA GUSTAVSSON

INSTITUTIONEN FÖR DATA- OCH INFORMATIONSTEKNIK
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET
Göteborg, Sverige 2022
www.chalmers.se

Examensarbete 2022

Implementation av Elasticsearch i Lina

Utformandet av en prototyp med fokus på att effektivisera sökningar och förkorta svarstider

KATARINA GUSTAVSSON

CHALMERS

Institutionen för Data- och Informationsteknik
Chalmers Tekniska Högskola
Göteborgs universitet
Göteborg, Sverige 2022

Implementation av Elasticsearch i Lina
Utformandet av en prototyp med fokus på att effektivisera sökningar och förkorta svarstider
KATARINA GUSTAVSSON

Handledare: Lars J Svensson, Institutionen för Data- och Informationsteknik
Examinator: Jonas A Duregård, Institutionen för Data- och Informationsteknik

Examensarbete 2022
Institutionen för Data- och Informationsteknik
Chalmers Tekniska Högskola
Göteborgs Universitet
SE-412 96 Göteborg
Telefon +46 31 772 1000

Omslag: Ett kollage av systemet Linas logga tillsammans med Elasticsearchs logga.

Typsatt i LaTeX
Tryckt av Chalmers Reproservice
Göteborg, Sverige 2022

Implementation av Elasticsearch i Lina
Utformandet av en prototyp med fokus på att effektivisera sökningar och förkorta svarstider
KATARINA GUSTAVSSON
Institutionen för Data- och Informationsteknik
Chalmers Tekniska Högskola
Göteborgs Universitet

Sammanfattning

Foxway använder i sitt dagliga arbete en egenutvecklad plattform kallad Lina. I takt med att företaget expanderar ökar kraven på de sökfunktioner som finns i systemet. Detta projekt har undersökt möjligheterna att implementera sökmotorn Elasticsearch i Lina för att effektivisera sökfunktionaliteten och förkorta svarstider. Målet med projektet var att den nya implementationen ska kunna göra samma typ av sökningar som systemet kan med nuvarande implementation men med förkortade svartider.

Med hjälp av Elasticsearch och Logstash har en prototyp utvecklats som en omarbetning av det existerande systemet. API-anrop som skickas från Linas användargränssnitt hämtar data, som på förhand indexerats genom Logstash, från Elasticsearch. Själva omarbetningen var lyckad men målet att förkorta svarstiderna uppnåddes inte.

Nyckelord: prototyp, sökmotor, Elasticsearch, Logstash, svarstider.

Förord

Denna rapport och projektet den är baserad på är resultatet av mitt examensarbete som utförts från vintern 2021 till våren 2022. Jag vill framföra ett tack till killarna i devteamet på Foxway, och i synnerhet Joel Rönnqvist, som bjöd in mig att göra mitt arbete hos dem. Stort tack till min handledare Lars J Svensson för att du tog mig under dina vingars skugga när jag behövde vägledning som mest. Utan ditt stöd hade jag inte varit här nu. Slutligen hjärtligt tack till Jassob Jonsson och Joakim Lönnerstedt som har korrekturläst och gett ovärderlig feedback på rapporten.

Katarina Gustavsson, Göteborg, juni 2022

Lista med Akronym

Nedan finns en lista över de akronym som har använts genom denna rapport, listade i alfabetisk ordning:

API	Application Programming Interface, sv. Applikationsprogrammeringsgränssnitt
CIL	Common Intermediate Language
CLR	Common Language Runtime
CRUD	Create Read Update Delete
DTO	Data Transfer Objects
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment, sv. Integrerad Utvecklingsmiljö
JDK	Java Development Kit
JRE	Java Runtime Environment
JSON	JavaScript Object Notation
JVM	Java Virtual Machine
MSSQL	Microsoft SQL Server
OAS	OpenAPI Specification
RDBMS	Relational Database Management System, sv. Relationsdatabas-hanteringssystem
RegEx	Regular expression, sv. Reguljära uttryck
REST	Representational State Transfer
RPC	Remote Procedure Calls
SaaS	Software as a Service
SOAP	Simple Object Access Protocol
SSMS	SQL Server Management Studio
SSPL	Server Side Public License
URI	Uniform Resource Identifier
XML	Extensible Markup Language

Innehåll

Figurer

Tabeller

1
Inledning

Följande kapitel ger en introduktion till projektets bakgrund, mål, frågeställningar och avgränsningar.

1.1 Bakgrund

Foxway Education arbetar med att leasa, serva och återvinna teknisk hårdvara i form av bland annat datorer och surfplattor [1]. Deras kunder består av både enskilda skolor som hela kommuner och användarna utgörs av elever och pedagoger [2]. I sitt arbete använder Foxway en egenutvecklad datahanteringsplattform som kallas Lina där alla verktyg och avtal finns registrerade och den används av både intern personal på företaget men även av deras kunder. Plattformen tillhandahåller ett antal olika funktioner men en stor del av dem kretsar kring att söka upp och ta fram specifik information om olika verktyg eller avtal [3]. Idag har Foxway Education drygt 700 000 sökbara verktyg registrerade i Lina.

I takt med att företaget expanderar blir både antalet verktyg och användare av den interna plattformen stadigt fler. Detta sätter press på de olika sökfunktionaliteternas effektivitet och har lett till att plattformens söktider har ökat. I nuläget skickas sökningar som SQL-förfrågningar direkt till deras databas, men då dessa i vissa fall kan bli mycket omfattande kan de ta upp till flera sekunder att utföra. Detta har skapat ett behov av en ny, effektivare sökmetod för att kunna förkorta sökningarnas svarstider.

Ett populärt verktyg för att lagra, analysera och söka bland data är Elasticsearch vilket används av välkända företag som bland andra Netflix, Ebay, GitHub och Microsoft [4]. Elasticsearch är en Java-baserad open source-sökmotor som bland annat använder sig av inverterade index för att ge snabba sökresultat [5].

1.2 Syfte

Syftet med projektet är att undersöka vilka möjligheter som finns att på ett smidigt sätt implementera sökmotorn Elasticsearch i plattformen Lina utan att behöva göra större förändringar i plattformens nuvarande uppbyggnad och struktur. Detta för att effektivisera sökfunktionaliteten och förkorta svarstider.

1.3 Mål

Effektiviseringen av sökfunktionaliteten möjliggörs genom Elasticsearchs stöd för att söka i specifika fält, som exempelvis efternamn och serienummer, utöver att också kunna utföra fulltextsökningar. Svarstider ska förkortas jämfört med den tid olika sökningar tar för nuvarande implementation. I absoluta mått mätt ska ingen svarstid vara längre än 4 sekunder, vilket är nuvarande maxtid, och idealt ska tiderna stadigt ligga under 1,5 sekunder. Ett antal sökfunktionaliteter av olika komplexitet och omfattning ska omarbetas för att testa den nya implementationens funktionalitet.

1.4 Frågeställningar

- Kan Elasticsearch effektivt implementeras i det nuvarande systemet?
- Kan en implementation av Elasticsearch förkorta sökningars svarstid?

1.5 Avgränsningar

Projektet kommer inte syfta till att bygga om något i användargränssnittet. Arbetet kommer inte ta några särskilda säkerhetsaspekter i beräkning för utformningen av prototypen, utan kommer endast fokusera på att använda Elasticsearch som lösning för hämtning av data. Skapande av ny och uppdatering av befintlig data kommer fortfarande ske direkt mot existerande databas. Målet för projektet är inte att omdirigera alla olika sökfunktioner till att använda Elasticsearch utan endast ett antal utvalda.

2

Teknisk Bakgrund

Detta kapitel ger information om de bakomliggande begrepp och system som använts under projektet. Begrepp med etablerade svenska namn kommer benämnas med dessa, nyare eller icke etablerade begrepp kommer medvetet benämnas med sina engelska namn för att undvika förvirring.

2.1 Elasticsearch

Elasticsearch är en analytisk sökmotor, det vill säga en sökmotor med en förmåga att indexera och söka bland en mängd olika sorters innehåll. Den har öppen källkod, är skriven i Java och kan användas för bland annat olika typer av sökningar, prestandaövervakning och logg- och säkerhetanalyser. Det finns officiella klienter för ett flertal stödda programspråk som till exempel Java, .NET (C#), Python och Ruby. Elasticsearch är baserat på Apache Lucene, vilket är ett högpresterande textbaserat sökbibliotek som även det är open source och skrivet i Java. Tack vare detta passar sökmotorn utmärkt för fulltext-sökningar och svarstiden är väldigt kort då latensen vid indexering typiskt är under en sekund [5],[6].

Elasticsearch lanserades första gången år 2010 och ägs av företaget Elastic. Elasticsearch använder en NoSQL-databas och till skillnad från relationella databaser stöds inte SQL-förfrågningar då datan lagras på ett ostrukturerat vis och inte på tabellform. Själva datan kan vara både strukturerad och ostrukturerad då det är möjligt att lagra den utan att datastrukturen fördefinieras [7]. Rådata förs in till Elasticsearch från till exempel loggar eller webbapplikationer. Datan analyseras och normaliseras innan den indexeras in i ett Lucene index, vilket beskrivs mer ingående nedan. Den indexerade datan kan sedan bland annat hämtas och summeras på olika vis med hjälp av en mängd funktioner i Elasticsearch API [5].

Elastic har utvecklat flera andra produkter för att förenkla och bredda användningen av Elasticsearch. Deras nyckelprodukter brukar tillsammans benämnas som Elastic Stack och består av Elasticsearch, Logstash, Kibana och Beats [8]. Dessa produkter är fria att använda under Server Side Public License (SSPL) men Elastic erbjuder även en avgiftsbelagd licens vilken inkluderar fler avancerade funktioner som användarhantering och machine learning parallellt med personlig teknisk support [5].

2.1.1 Lucene index

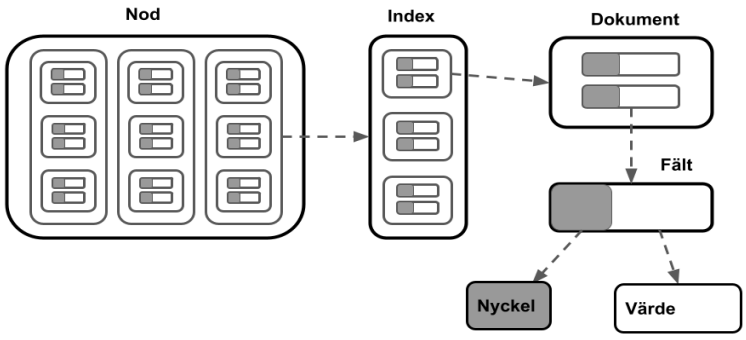
Apache Lucene är ett sökbibliotek i Java som möjliggör fulltext-sökningar. Elasticsearch är ett distribuerat system byggt ovanpå detta vilket ger ett ytterligare abstraktionslager. Elasticsearch REST API utnyttjar Lucenes funktioner och gör dem mer lätt-tillgängliga för utvecklare som vill nyttja dem [9]. Nyckeln till att sökmotorn är så snabb ligger i hur ett Lucene index fungerar.

Lucene använder sig av ett fulltext-index, även kallat inverterat index, vilket innebär att data som förs in i indexet sorteras upp i sökord tillsammans med de platser där orden förekommer i datan. Sökningar görs sedan genom dessa sökord, vilket kan jämföras med att söka genom ett register längst bak i en bok, istället för att allt innehåll ska sökas igenom från början till slut [10].

2.1.2 Beståndsdelarna i Elasticsearch

Den största enheten i en instans av Elasticsearch kallas för ett index, vilket ej är att förväxla med ett inverterat index. Ett index kan innehålla ett eller flera dokument och ett dokument är ett JSON-objekt. JSON är ett textbaserat datautbytesformat baserat på skriptspråket JavaScript och ett objekt innebär en samling osorterade fält [11]. Ett dokument består i sin tur av ett eller flera fält, vilket är den minsta enheten i Elasticsearch. Ett fält är ett datapar innehållande en nyckel vilket är namnet på fältet och ett värde, alltså datan för det specifika dokument som fältet tillhör. Indexering innebär att lägga till dokument, med tillhörande fält, till ett index och göra denna data tillgänglig för senare sökningar. Dessa sökningar hämtar sedan de dokument som bäst matchar sökspecifikationerna [10],[12]. Ett dokument kan till exempel bestå av ett finansbolag och dess fält av id, namn och adress. Ett index kan då vara en samling av olika finansbolag.

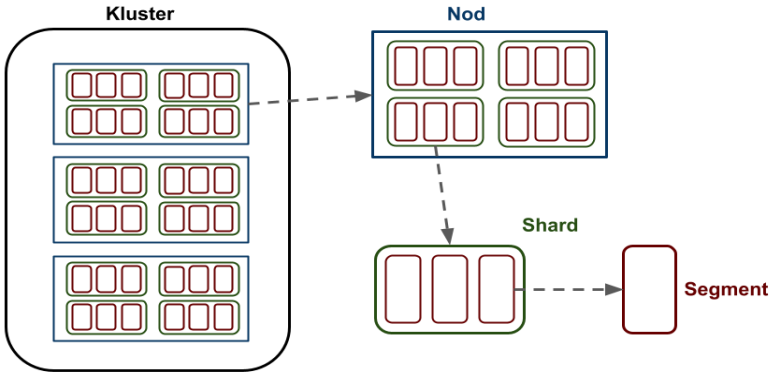
Varje instans av Elasticsearch kallas för en nod och en eller flera noder med samma identifierare kallas tillsammans för ett kluster. Det är varje enskild nods uppgift att lagra och indexera data i dess olika index. Olika noder kan i större kluster ha olika roller. Bland annat behövs en eller flera master-noder som administrerar alla noder och ansvarar för att lägga till och ta bort noder från deras kluster. Ett annat exempel är data-noder som har ansvar för att utföra sökningar och aggregationer [9].



Figur 2.1: Relationen mellan noder, index, dokument och fält

Varje index i Elasticsearch (alltså ett eller flera dokument) är uppdelat i shards. En shard består av ett Lucene index (alltså ett inverterat index) och dessa är fördelade mellan de noder som finns i ett kluster. Användning och fördelning av shards är vad som hjälper Elasticsearch att nå högre effektivitet och öka datans skalbarhet och tillgänglighet [9]. Varje shard är uppdelad i mindre delar kallade segment. Dessa är i sin tur små Lucene index och innehåller referenser till den faktiska datan. Alla segment söks igenom sekvensiellt. Ett segment som skapats går inte i efterhand att uppdatera utan det går endast att skapa nya segment, exempelvis när nya dokument indexeras. Mindre segment sammanfogas automatiskt efter en tid, eller manuellt om så önskas, för att upprätthålla sökningars effektivitet. Detta sker genom att de läggs in i ett tredje nytt segment och de föregående två sedan raderas [13].

Att ha data fördelad på olika noder vilka i sin tur finns på olika servrar gör varje nod till en felkritisk systemdel (eng. single point of failure). För att undvika detta och därigenom öka feltoleransen finns det hos Elasticsearch en haverisäker mekanism kallad replikor. Replikor är helt enkelt kopior av primära shards och är även de fördelade över noderna. Detta innebär att all data kan finnas på minst två ställen, beroende på hur många replikor användaren bestämmer att shards tillhörande olika index ska ha. Om en nod krashar finns det alltså reserver för de shards som gått förlorade och klustret fördelar självt om sina resurser för att återfå balans och redundans. Replikor används dessutom vid sökningar precis som primära shards och de kan därför också öka sökprestandan [9][14].



Figur 2.2: Relationen mellan kluster, nod, shard och segment

2.2 Elastic Stack

I samband med ElasticSearch nämns ofta Elastic Stack, vilket består av fyra produkter som ofta samverkar där alla ägs av företaget Elastic. Tidigare kallades samlingen för ELK Stack som ett akronym för Elastisearch, Logstash och Kibana. Senare tillkom även Beats, varav samlingens namnbyte [15], men i detta projekt nyttjas endast de förstnämnda tre produkterna.

2.2.1 Logstash

Logstash är en så kallad datainsamlingsmotor med öppen källkod, ett verktyg baserat på programmeringsmönstret Pipe-and-filter. Det är ett mönster för dataströmsprocesser, även kallat pipelines, för att samla, bearbeta och generera data, loggar eller dataevent. Programvaran är skriven i JRuby, som är en implementation av programmeringsspråket Ruby ovanpå Java Virtual Machine (JVM) [16], och kräver därför JVM för att fungera [17]. Inmatningsdatan kan vara ostrukturerad och till hjälp finns en mängd färdigkonstruerade filter och instickspprogram som är fria att använda. Logstash har realtidsbearbetning av sina dataströmmar och har kapacitet att samla data från en mängd olika sorters källor. Inläst data kan standardiseras eller omvandlas innan den skickas vidare till vald destination, till exempel Elasticsearch [18].

Logstash består av en pipeline och funktionaliteten är uppbyggd i tre steg. Första steget i pipelinen är input, vilket sätter upp en koppling för att ta emot datan. Nästa steg är filter där eventuell bearbetningen av datan sker, till exempel med hjälp av RegEx-mönster. Pipelinen avslutas med output där destination och formatering definieras [17].

2.2.2 Kibana

Kibana är ett verktyg för visualisering och granskning av data och fungerar som ett användargränssnitt till Elastic Stack. Det är en webbläsarbaserad applikation med öppen källkod utvecklad för att ge en visuell representation av data indexerad i Elasticsearch. Bland applikationens funktioner finns bland annat möjlighet till applikationsövervakning, logg-analyser och administration av Elastics program. Detta med hjälp av olika diagram och tabeller men också mindre inbyggda applikationer och olika filterfunktioner [19][20].

Ett hjälpmedel i Kibana är utvecklingsverktyget Dev Tools vilket innehåller funktioner för att interagera med den data som finns lagrad i Elasticsearch. En av funktionerna är Console vilket fungerar som en terminal och ger möjlighet att interagera med Elasticsearchs REST API. Bland annat genom att skicka olika förfrågningar och ta del av API-dokumentationen [21]. Terminalens syntax är mycket lik cURL, vilket är ett vida använt kommandoradsverktyg och bibliotek[22], och utgår från det kända akronymet CRUD: Create, Read, Update, Delete.

2.3 REST API

Application Programming Interface (API), eller applikationsprogrammeringsgränssnitt, är en samling funktioner som gör det enklare att dela data mellan applikation och server, operativsystem eller andra applikationer på ett smidigt och säkert sätt. Ett API är en mellanhand för kommunikation som gör att applikation och server aldrig är direkt exponerade mot varandra och många av dagens webbapplikationer är helt beroende av gränssnittens funktion [23]. Representational State Transfer

(REST) är ett programmeringsmönster med att antal regler som utvecklare kan följa när de skapar ett API. REST är generellt den föredragna modellen för APIer bland mjukvaruutvecklare och den mest använda idag [24].

APIer möjliggör specifika anrop för att skicka och ta emot information. Den förfrågande parten, som kan vara en användare eller en applikation, kallas för klienten och det objekt som APIet ber om information om kallas för resurs. Dataöverföringen börjar med att en klienten gör ett API-anrop för att få information. Denna förfrågan skickas från applikation till server via APIets Uniform Resource Identifier (URI). Om en giltig förfrågan tagits emot av den server som implementerar APIet görs ett anrop till resursen vilket i sin tur skickar tillbaka den eftersökta informationen via APIet. Datan kan efter det skickas till klienten och på så vis har direktkontakt mellan slutparterna undvikits [23].

REST APIer, även kallade RESTful API, använder kommunikationsprotokollet Hypertext Transfer Protocol (HTTP) och kommunicerar med hjälp av JSON. Därav kan ett API utföra de fyra anropen GET, PUT, POST och DELETE, vilka sammanfaller med CRUD [24][25]. REST-kompatibla system karaktäriseras av att de är tillståndslösa och att de separerar klient och server. Detta innebär att servern inte behöver veta något om klientens tillstånd, och vice versa, och att implementationen av den ena kan göras oberoende av den andra. Det medför att server och klient kan förstå alla meddelanden som mottas, även utan att se föregående meddelanden och att kod när som helst kan ändras på ena sidan utan att det påverkar driften av den andra [26]. Detta hjälper REST-applikationer att uppnå pålitlighet, hög prestanda och skalbarhet så komponenter kan förvaltas, uppdateras och återanvändas utan att påverka systemet som helhet [26].

Utöver REST finns ett antal andra olika sorters tekniska specifikationer eller protokoll vilka definierar regler om vilka datatyper och kommandon som accepteras. Till exempel finns XML-RPC och JSON-RPC som använder sig av fjärranrop, Remote Procedure Calls (RPC), vilket är ett anrop som kan innehålla flera parametrar men som förväntar sig ett enda svar. Det vanligaste protokollet är dock Simple Object Access Protocol (SOAP), som är en designmodell för webbtjänster. Det är ett API-protokoll som använder Extensible Markup Language (XML), vilket är ett textformat likt HTTP [24]. Det är möjligt att bygga ett REST API med SOAP-protokoll men dessa två standarder är vanligvis sedda som konkurrerande specifikationer [23].

2.3.1 Swagger

OpenAPI Specification (OAS) är en teknisk specifikation som definierar en standard för gränssnitt för att beskriva eller visualisera HTTP APIer. Alla HTTP APIer är inte REST APIer men specifikationen är främst för denna typ av mönster [27]. OAS är en community-driven öppen specifikation inom OpenAPI Initiative, vilket är ett Linux Foundation kollaborations-projekt.

Swagger ett gränssnitt bestående av en serie verktyg som implementerar OAS och

finns till för att underlätta interaktion med APIer. Gränssnittet hjälper till att beskriva, producera och visualisera APIer och deras resurser på ett sätt som är enkelt att förstå för utvecklare eller andra testare utan att behöva ta del av källkod, utökad dokumentation eller nätverkstrafik [28]. Swagger UI, som är ett av verktygen, genererar automatiskt en visuell representation av APIerna i webbläsaren vilket gör dem mer lättöverskådliga med ett användarvänligt användargränssnitt. En utvecklare kan enkelt köra och övervaka de API-förfrågningar som skickas och vilka resultat som återsänds. Swagger förenklar möjligheten att ändra eller uppdatera APIerna vilket gör att det passar bra för testning och felsökning, samtidigt som skalbarheten förbättras [29].

2.4 Lina

Lina är Foxways system för resursförvaltning och ärendehantering. Systemet låter kunder hantera de digitala verktyg som Foxway sålt eller leasat till dem. Detta innefattar till exempel att lägga ärenden på verktygen för garanti eller reparation och kommunicera med Foxway om vad för åtgärder som kan eller kommer ske. Lina har dessutom en central roll för flera arbetsprocesser hos Foxways avdelningar samt deras funktionalitet. Bland dessa finns avdelningarna Service, Teknik, Utdelning, Recycle och Logistik. Varje år har Lina cirka 1800 unika användare och hanterar i nuläget över 460 000 aktiva digitala verktyg.

2.4.1 C# och .NET

Lina är byggt på programmeringsspråket C# och ramverket .NET 5. C# är ett objektorienterat, komponentorienterat och typsäkert programmeringsspråk. Språket har rötter i programmeringsspråket C och kan ses som en efterföljare till både C och C++ med influenser av Java. Språket och syntaxen är lätt att känna igen för programmerare bekanta med C, C++, Java och JavaScript. C# utgavs av Microsoft år 2002 för att användas till deras .NET-plattform och var ursprungligen byggt enbart för Windows. Numera har språket öppen källkod och är plattformsoberoende [30]. Applikationer utvecklade med C# kan alltså köras av alla typer av operativsystem.

.NET är en utvecklingsplattform för att bygga olika applikationer och en exekveringsmiljö för att kompilera och exekvera program skrivna i olika språk. Ramverket är kostnadsfritt, har öppen källkod och är ett projekt som hanteras genom .NET Foundation [31]. C# körs ovanpå .NET vilket förenklat innebär att kod skrivet i C# använder klasser och funktioner från .NET. Exekvering av källfiler i C#s resulterar i en exekveringsfil som kräver .NETs klassbibliotek för att kunna köras tillsammans med ett virtuellt exekveringssystem. .NET består av klassbibliotek vars kod är uppdelad i olika moduler som kan nyttjas beroende på vad som ska utföras. Detta gör plattformen flexibel och lättanvänd [32]. Klassbiblioteken är återanvändbara för utvecklingen av olika applikationer. Exekveringssystemet är känt som Common Language Runtime (CLR) vilket är en virtuell komponent av .NET. CLR är Microsofts implementation av Common Intermediate Language (CIL), en internationell stan-

dard. CIL är grunden för att skapa exekverings- och utvecklingsmiljöer där språk och bibliotek samarbetar sömlöst [30].

C# har utvecklats parallellt med .NET och är tänkt att vara det naturliga språket för att utveckla program för ramverket. .NET-utvecklare använder vanligtvis C# som programmeringsspråk men plattformen kan förstå en mängd olika språk som bland annat C++, VB.NET och F#. Det går alltså att utveckla program för .NET utan att använda C#. Det finns utvecklingsmiljöer som kan kompilera flera språk till CIL vilket CLR sedan kan tolka till maskinkod som exekveras [32]. .NET ger tillgång till samma exekveringsmiljö, API och språkkompabiliteter med varje applikation och ger möjlighet att använda plattformsspecifika egenskaper, som operativsystems-APIer [31].

2.4.2 NuGet

Ett nödvändigt verktyg för moderna utvecklingsplattformar är en mekanism som låter utvecklare skapa, dela och utnyttja återanvändbar kod. För att uppnå detta har .NET en pakethanterare kallad NuGet som definierar hur paket skapas, hyses och utnyttjas [33]. NuGet är en Software as a Service (SaaS)-lösning där NuGet Gallery är den centrala datakatalogen för paket-skapare och dess användare [34]. Applikationen var från början en förlängning av utvecklingsmiljön Visual Studio men är idag kostnadsfri med öppen källkod. Ett NuGet-paket är en .zip-fil som består av en .nupkg eller .nupak-fil och innehåller kompilerad kod, andra filer relaterade till koden och paketbeskrivning som inkluderar information som paketets versionsnummer. Utvecklare med kod att dela skapar paket och publicerar dem till nuget.org eller en privat värd. Utvecklare som vill använda delad kod lägger till paket till deras projekt och kan kalla på det API som exponeras av paketet i deras projektkod [31].

Elasticsearch tillhandahåller två NuGet-paket, Elasticsearch.Net och NEST. NEST är den officiella högnivå-klienten och Elasticsearch.Net är en beroendefri lågnivå-klient; NEST använder internt Elasticsearch.Net. Klienterna kopplar förfrågingar och svar 1-till-1 med Elasticsearchs REST API och alla deras klientmetoder har stöd för både synkrona och asynkrona varianter [35].

2.4.3 Visual Studio

Visual Studio är en programutvecklingsmiljö som är utvecklad av Microsoft och kan användas med Windows. Det har omfattande inbyggd funktionalitet designad att användas för utveckling av C++ och .NET. Visual Studio används för att utveckla PC-baserade applikationer för Windows och webbaserade applikationer.

Visual Studio implementerar konceptuella containers kallade solutions och projekt där en solution genereras automatiskt när ett nytt projekt skapas. En solution inkluderar ett eller flera projekt plus filer och metadata som hjälper till att definiera projektet som en helhet [36]. Ett projekt består av alla källkod-filer, datafiler och

annat som behövs för att genomföra en kompilering tillsammans med kompileringsinställningar och andra konfigurationsfiler som kan behöver av olika tjänster eller komponenter som applikationen kommer kommunicera med [37].

2.4.4 Programstruktur

Linas backend består av en solution innehållande 48 projekt. När ett REST API-anrop görs från Linas webbapplikation eller från Swagger går det först via projektet WebApi. Projektet innehåller controller-filer vilka i sin tur kallar på respekitive repository-filer som båda är källkodsfiler lokaliserade i olika tillhörande projekt. För att datan som sänds tillbaka ska presenteras på rätt sätt används Data Transfer Objects (DTO). En DTO är helt enkelt ett objekt som definierar hur datan ska skickas, som en mall [38].

2.5 Databas

Den existerande databasen innehåller all data som används i Lina och hämtas och uppdateras via REST APIer. Den databashanterare som används är Microsoft SQL Server (MSSQL), vilket är ett ledningssystem för relationsdatabaser (RDBMS) utvecklat av Microsoft [39]. För interaktion med databasen används SQL Server Management Studio (SSMS) vilket är en integrerad utvecklingsmiljö (IDE) för databaser [40].

3
Metod

Följande kapitel redovisar för de metoder som används under projektet. Dels den arbetsmetod som följs men även vilken metod som ligger till grund för den testning som producerat jämförbara resultat.

3.1 Arbetsmetod

Projektet kommer i stort följa projektledningsmetoden vattenfallsmodellen. Alltså att arbetet delas in i olika faser och att dessa avklaras en efter en, vilket innebär en enkelriktad utvecklingsprocess [41]. Viss kontinuerlig efterforskning kan dock komma att krävas under projektets gång. Ett agilt arbetssätt är inte nödvändigt då projektet är mer målstyrt än målsökande.

Projektet förbereds genom att planera och sätta upp specifikationer tillsammans med projektägarna, utvecklingsteamet på Foxway. Mål och avgränsningar sätts upp, vad som ska ändras och vad som ska uppnås bestäms och en tidsplan tas fram. Själva arbetet inleds med en introduktion av de existerande interna systemen och en process av att bekanta sig med dessa. För att kunna utföra de förändringar som efterfrågas krävs insikt i nuvarande systems uppbyggnad och sammansättning. Vilka delar som berörs utreds.

När en projektplan satts upp börjar en viktig del av arbetet bestående av en efterforskningsfas. Frågor som ”Vad är Elasticsearch och på vilket sätt kan det användas i detta projekt?” och ”Vilka andra system och verktyg kommer krävas för att få ut den funktionalitet som eftersträvas?” bemöts och besvaras. Vidare ska design och implementation beaktas. Det ska fastställas hur den nya strukturen ska se ut och hur nya delar kopplas samman med gamla. En översiktlig skiss på den nya systemkonstruktionen tas fram och det beslutas om var den nya implementationen ska ske.

Nästa fas består av själva implementationen. En instans av Elasticsearch ska upprättas, relevant data hämtas från databasen och sökningar och hämtningar direkt via den nya instansen ska möjliggöras. Systemutvecklingen kommer utföras på ett sätt som möjliggör testning vid varje steg för att ge en indikation på huruvida implementationsstrategin är möjlig eller inte. Slutligen ska både det tidigare systemet och de nya implementationerna testas separat från varandra för att få fram svarstider och eventuella skillnader i effektivitet. Målet med projektet möts i och med de slutsatser som dras av resultattiderna som fås fram av testningen i arbetets slutskede.

3.2 Testning

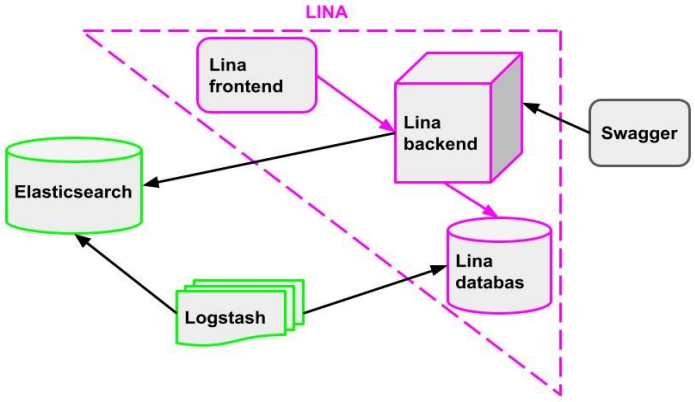
Projektet har resulterat i en prototyp, en omarbetning av nuvarande system, som utnyttjar Elasticsearch som databas och sökmotor parallellt med den redan existerande databasen. Omarbetningen kan göra både fulltext-sökningar och söka efter specifika fält på samma sätt som originalversionen av systemet. All funktionalitet är dock inte exakt likadan så för en del sökningar skiljer sig resultatet mellan implementationerna. Exempelvis söker originalimplementationen efter alla förekomster av ett sökord, oavsett om det är del av ett ord eller hela ordet, medan omarbetningen inte tar med resultat där ett ord har andra tecken eller bokstäver direkt förkommande sökordet. En sökning i omarbetningen efter sökordet ”skola” skulle exempelvis få fram enheter tillhörande ”Göteborgs skola” som resultat men inte ”Göteborgsskolan” eller ”Göteborgs förskola”. Det är en tydlig brist i implementationen men var ett aktivt beslut att utesluta i ett försök att utnyttja Elasticsearchs fulla potential och effektivisera sökresultaten.

Prototypens effektivitet har testats genom att testtider har tagits fram vilka jämförts med motsvarande testtider från implementationen i produktion. Testtiderna har producerats med hjälp av Chrome Devtools vilket är Google Chromes inbyggda webutvecklings-verktyg som nås direkt i webbläsaren. Tiderna representerar den tid varje test tagit och testerna består av enskilda, specifika metodanrop där en API-förfrågan skickats från Linas användargränssnitt och hämtat data från en instans av Elasticsearch uppsatt på en extern server.

4

Systemkonstruktion

Linas frontend skickar förfrågningar till backend som i sin tur skickar efter data från databasen. Med implementationen av Elasticsearch skickas förfrågningar från backend till en instans av Elasticsearch som redan har fått all relevant data från databasen genom Logstash.



Figur 4.1: Hur Elasticsearch, Logstash och Swagger interagerar med Lina

4.1 Installation och konfiguration

Projektet har utförts på operativsystemet Windows och alla installationer är anpassade därefter.

4.1.1 Elasticsearch

Elastcisearch behöver Java Virtual Machine (JVM) för att kunna köras. Tidigare versioner har krävt separat nedladdning av Java Runtime Environment (JRE), som innehåller JVM, alternativt Java Development Kit (JDK), som JRE är en del av. Efter nedladdning behöver en ny systemvariabel, ”JAVA_HOME”, sättas till JVMets sökväg i operativsystemets systeminställningar. Från Elasticsearch 7.0 och framåt inkluderas en variant av OpenJDK som är ett JDK med öppen källkod. Projektet har använt Elasticsearch 7.16.2

Installation av Elasticsearch består av att ladda ner en zip-fil och sedan extrahera den i vad som blir hemkatalogen. I hemkatalogen finns en konfigurationsfil med

filtillägget yml, vilket är ett dataserialiseringsformat. Denna uppdateras med namn för variablerna *cluster.name* och *node.name*. Vidare väljs *Network.host* till den lokala IP-adressen och *http.port* förblir standard 9200. Installationen avslutas med att köra ”elasticsearch.bat”.

4.1.2 Logstash

Även Logstash behöver JVM för att kunna köras. Från Logstash 7.10 och framåt är OpenJDK inkluderat. Installation av Logstash består av att ladda ner en zip-fil och extrahera den i en egen hemkatalog. Inga förändringar har gjorts i konfigurationsfilen för Logstash. För att en anslutning ska kunna upprättas mot databasen krävs en Java Database Connectivity (JDBC)-drivrutin för SQL Server.

4.1.3 Kibana

Installation av Kibana består också av att ladda ner en zip-fil och extrahera den. I den tillhörande konfigurationsfilen uppdateras variabeln *server.host* med samma IP-adress som instansen av Elasticsearch körs på och variabeln *elasticsearch.hosts* med instansens URL. Variabeln *server.port* förblir standard 5061.

4.2 Noder och index

Då projektet har hanterat en relativt liten mängd data så har ett en-nods-kluster använts under utvecklingen. I Elasticsearchs konfigureringsfil behöver variabeln *discovery.type* sättas till single-node för att klustret ska köras i utvecklarläge. Om implementationen skulle skickas ut i produktion hade det idealt gjorts med ett tre-nods-kluster, alternativt ännu fler noder beroende på aktuell datamängd. Detta dels för att få en högre feltolerans då redundans kan uppnås genom möjligheten att skapa replikor åt alla shards och dels att flera instanser ger fler sökbara shards vilket ökar sökprestandan vid större mängder data. I det läget behöver en masternod förbestämmas genom variabeln *cluster.initial_master_nodes*. Totalt har fem index skapats och använts under projektet. De är baserade på existerande tabeller i Linas databas och innehåller data relevanta för de sökfunktioner som har modifierats. Indexen som skapats är ”Financialfirms”, ”Groups”, ”Units”, ”Users” och ”Nested_groups”.

4.3 Från databas till Elasticsearch

Logstash har använts för att skapa nya index i Elasticsearch och föra in data i dem från Linas databas. För varje index har en separat konfigureringsfil utformats. Varje fil består av inmatning, eventuellt filter och utmatning. Inmatningen innehåller sökväg till JDBC-drivrutinen och anslutningsinformation för att nå databasen men även en SQL-sats som definierar vilken data som ska hämtas och skickas vidare. Utmatningen definierar vilken Elsticsearch-instans och index som datan ska skickas till tillsammans med vilket fält som dokumenten ska identifieras efter. Konfi-

gurationsfilen till ”Nested_groups” innehåller också ett filter som består av en ny hämtning av data från databasen vilken sedan sammanfogas med datan från den första inmatningen. Detta för att skapa ett så kallat nästlat index, alltså ett index innehållande dokument som i sin tur också innehåller dokument. I detta fall grupper som innehåller enheter.

4.4 Omarbetning av kod

Projektet har fokuserat på tre olika sökområden: finansbolag, grupper och enheter. Omarbetningen av den befintliga kod som Linas backend består av har i huvudsak skett i de repository-filer som är kopplade till dessa sökområden. Varje repository-fil har en funktion för att sätta upp en klient till en instans av Elasticsearch. Funktionen definierar bland annat instansens URI och vilket index klienten ska söka mot om inget annat anges. De omarbetade filerna har nya implementationer av de metoder som kallas på via GET-förfrågningar men har inte förändrat hanteringen av andra typer av API-förfrågningar. Varje GET-förfrågning förväntar sig svar bestående av olika sorters DTOer.

4.4.1 Finansbolag

Sökområdet finansbolag använder sig av data om olika finansbolag. API-förfrågan försväntar sig svar på formen FinancialFirmDto. Datan i indexet ”Financialfirms” är hämtad från en motsvarande tabell i databasen. Utvecklingsarbetet har uteslutande skett i tillhörande repository-fil. Funktioner som uppdaterats är Get-metoder för alla finansbolag och för specifika bolag baserat på namn eller id.

4.4.2 Grupper

Sökområdet grupper använder sig av data om grupper och deras tillhörande enheter. API-förfrågan förväntar sig svar på formen GroupDto som i sin tur innehåller SimpleUnitDto. Datan i indexet ”Groups” är hämtad från en motsvarande tabell i databasen. Indexet ”Nested_groups” innehåller data från tabellen med grupper kombinerat med en tabell med enheter. Utvecklingsarbetet har skett i repository-filen för grupper men även filen GroupDto. Funktioner som uppdaterats är Get-metoder för alla grupper och för specifika grupper baserat på namn, id och enhet. Ett par hjälpfunktioner har tillkommit för att analysera och omvandla söktext.

4.4.3 Enheter

Sökområdet enheter använder sig av data om enheter och vilket grupp de tillhör. API-förfrågnan förväntar sig svar på formen UnitDto. Datan i indexet ”Units” är hämtad från tabellen med enheter, även indexet ”Users” har använts vilket innehåller data från en tabell med användare. Utvecklingsarbetet har skett i repository-filen för enheter men även filen UnitDto. Funktioner som uppdaterats är Get-metoder för alla enheter och för specifika enheter baserat på id, kundid, aktörsnummer med mera. Ett par hjälpfunktioner har tillkommit för att analysera och omvandla söktext.

5
Resultat

Implementation av Elasticsearch i Lina har varit möjlig utan att behöva göra större förändringar i plattformens nuvarande uppbyggnad och struktur. Projektet har resulterat i en prototyp, en omarbetning av nuvarande system, som utnyttjar Elasticsearch som databas och sökmotor parallellt med den redan existerande databasen. Ett antal sökfunktionaliteter av olika komplexitet och omfattning har med framgång omarbetats.

I tabellerna nedan avser ”IMP” (implementation) den implementation som detta projekt har resulterat i och ”PROD” (produktion) originalimplementationen av systemet. ”Avg” redovisar medeltiden av 30 tagna tester och ”Min/Max” visar den kortaste respektive längsta tiden av dessa test.

5.0.1 Finansbolag

Totalt finns 24 finansbolag. Sökområdet finansbolag har inte funktionen att söka efter specifika bolag utan de enda val som går att ändra är antal sökträffar som ska visas per sida. Följande tester visar 10 respektive 50 sökträffar. Resultaten för testerna redovisas i Tabell 5.1.

Tabell 5.1: Testresultat för finansbolag

Test nr	IMP Avg	IMP Min/Max	PROD Avg	PROD Min/Max
1	83 ms	35 ms/328 ms	30 ms	18 ms/76 ms
2	74 ms	37 ms/188 ms	43 ms	17 ms/108 ms

Implementationernas resultat skiljer sig något men har båda acceptabelt korta svarstider. Då den begränsade funktionen gör att det finns få testmöjlgheter och indexet är litet med få resultat är det inte möjligt att dra några slutsatser utöver att resultaten är godtagbara.

5.0.2 Grupper

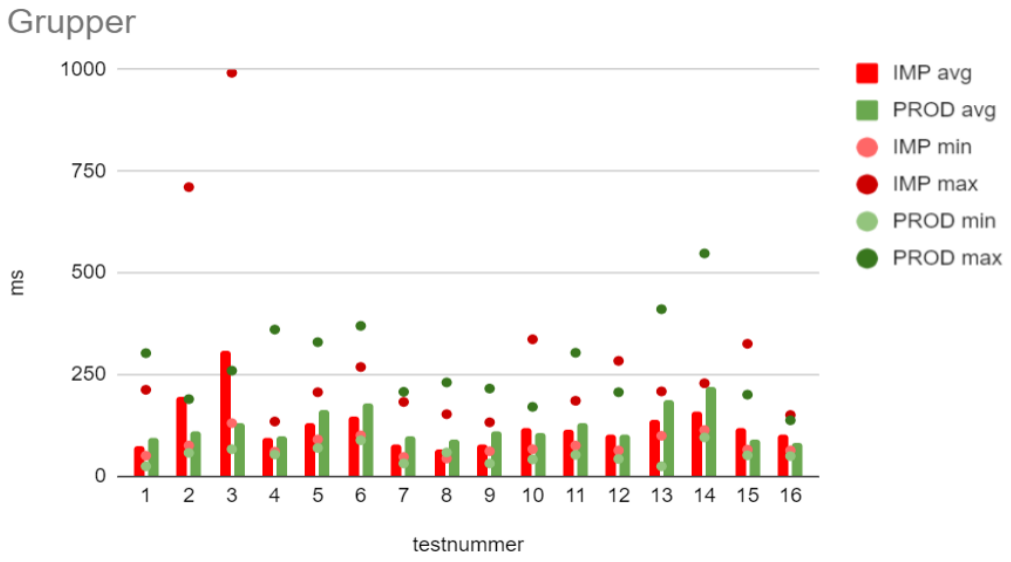
Det finns totalt 275 grupper. Testerna gör sökningar över alla grupper, frisöker på utvalda ord, söker efter specifika gruppnamn och enhetsnamn både enskilt och samtidigt. Alla test har gjorts med flera olika antal visade sökträffar. Resultaten för testerna redovisas i Tabell 5.2.

Tabell 5.2: Testresultat för grupper

Test nr	IMP Avg	IMP Min/Max	PROD Avg	PROD Min/Max
1	76 ms	51 ms/213 ms	94 ms	25 ms/303 ms
2	194 ms	76 ms/711 ms	112 ms	58 ms/190 ms
3	308 ms	131 ms/992 ms	132 ms	67 ms/260 ms
4	93 ms	61 ms/135 ms	97 ms	54 ms/361 ms
5	132 ms	91 ms/207 ms	162 ms	70 ms/330 ms
6	149 ms	101 ms/269 ms	181 ms	89 ms/370 ms
7	79 ms	48 ms/183 ms	100 ms	32 ms/208 ms
8	68 ms	44 ms/153 ms	92 ms	59 ms/231 ms
9	79 ms	62 ms/133 ms	109 ms	32 ms/216 ms
10	120 ms	67 ms/337 ms	106 ms	42 ms/171 ms
11	114 ms	76 ms/186 ms	130 ms	53 ms/304 ms
12	102 ms	64 ms/284 ms	104 ms	43 ms/207 ms
13	138 ms	100 ms/209 ms	188 ms	25 ms/411 ms
14	160 ms	114 ms/229 ms	221 ms	96 ms/548 ms
15	120 ms	66 ms/326 ms	91 ms	52 ms/201 ms
16	101 ms	64 ms/151 ms	83 ms	50 ms/138 ms

Resultatena redovisar att de båda implementationerna följer liknande svarstider över alla tester. Testerna klargör att omarbetningen inte är genomgående effektivare men resultaten är ändå tillfredsställande för ett index i denna storlek och sökningar med så pass korta svarstider. En graf över testresultaten för grupper visas i Figur 5.1. Här representerar staplarna medeltiden för testerna och prickarna testens längsta respektive kortaste tid för de olika implementationerna. Grafen tydliggör att resultatet av omarbetningen i flera fall ger förkortade svarstider och att maxtiden i de flesta fall är kortare än för implementationen i produktion.

Figur 5.1: Graf över resultat av tester för grupper



5.0.3 Enheter

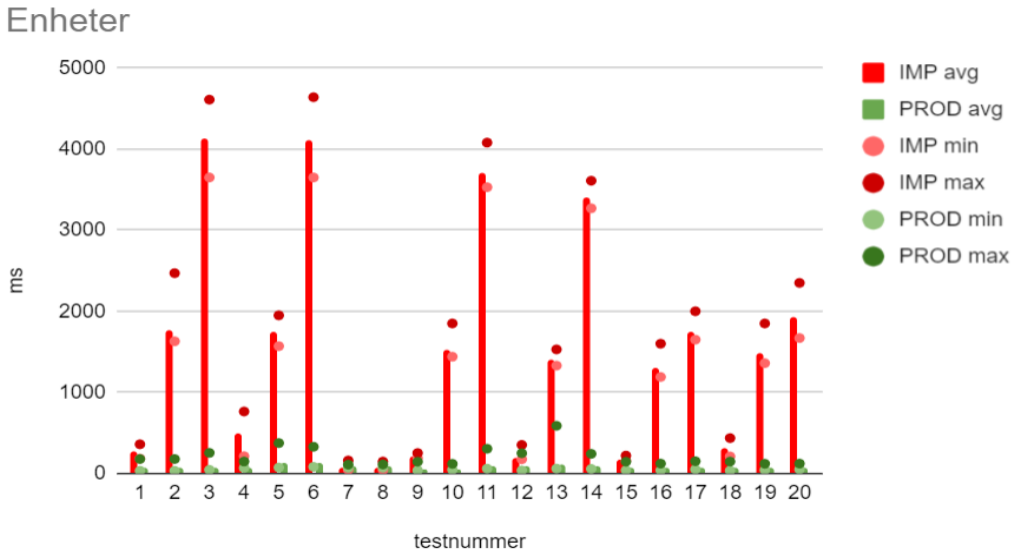
Det finns totalt 3087 enheter. Testerna gör sökningar över alla enheter, frisöker på utvalda ord, söker efter specifika enhetsnamn, gruppnamn och leveransadresser både enskilt och samtidigt. Alla test har gjorts med flera olika antal visade sökträffar. Resultaten för testerna redovisas i Tabell 5.3.

Originalimplementationen håller jämna, mycket korta svarstider medan omarbetningen inte klarar att hålla nere tiderna och i princip genomgående har långsammare resultat. Svarstiderna visualiseras i Figur 5.2 och här syns tydligt att resultaten för vissa test skiljer sig mycket mellan implementationerna medan för andra test är resultaten likvärdiga. De fyra testerna där tiderna skiljer sig som mest, test 3, 6, 11 och 14, har gemensamt att antal visade träffar är satt till 250 stycken. Det gäller även test 17 och 20 men där hade sökningarna endast 137 respektive 129 träffar och visade således inte så många som 250. De test där resultaten för de olika implementationerna är, relativt till sammanhanget, tidsmässigt nära varandra har gemensamt att antalet visade resultat är tio.

Tabell 5.3: Testresultat för enheter

Test nr	IMP Avg	IMP Min/Max	PROD Avg	PROD Min/Max
1	269 ms	183 ms/360 ms	70 ms	33 ms/178 ms
2	1.76 s	1.63 s/2.47 s	70 ms	33 ms/178 ms
3	4.13 s	3.65 s/4.61 s	75 ms	45 ms/254 ms
4	499 ms	213 ms/764 ms	77 ms	68 ms/148 ms
5	1.74 s	1.57 s/1.95 s	140 ms	73 ms/375 ms
6	4.10 s	3.65 s/4.64 s	138 ms	82 ms/329 ms
7	71 ms	44 ms/160 ms	81 ms	71 ms/109 ms
8	64 ms	42 ms/147 ms	84 ms	72 ms/111 ms
9	210 ms	190 ms/252 ms	58 ms	38 ms/145 ms
10	1.52 s	1.44 s/1.85 s	57 ms	46 ms/119 ms
11	3.70 s	3.53 s/4.08 s	93 ms	59 ms/305 ms
12	199 ms	175 ms/352 ms	91 ms	37 ms/250 ms
13	1.41 s	1.33 s/1.53 s	117 ms	60 ms/588 ms
14	3.41 s	3.27 s/3.61 s	97 ms	56 ms/242 ms
15	177 ms	159 ms/220 ms	79 ms	39 ms/145 ms
16	1.30 s	1.19 s/1.60 s	66 ms	38 ms/122 ms
17	1.74 s	1.65 s/2.00 s	73 ms	51 ms/151 ms
18	304 ms	210 ms/436 ms	70 ms	35 ms/147 ms
19	1.49 s	1.36 s/1.85 s	66 ms	43 ms/120 ms
20	1.93 s	1.67 s/2.35 s	70 ms	50 ms/122 ms

Figur 5.2: Graf över resultat av tester för enheter



6
Slutsats

Projektet undersökte möjlighetena att implementera Elasticsearch i Lina och resulterade i en prototyp. Ett antal sökfunktionaliteter omarbetades och tillräckligt korrekt funktion uppnåddes. Projektet byggde enligt avgränsningarna inte om något i användargränssnittet och säkerheten som tagits i beaktning har varit minimal. Resultatet visar dock att målet att effektivisera sökfunktionaliteten och förkorta svartider inte har uppnåtts. Flera av testerna håller jämna steg med originalimplementationen, testerna för både finansbolag och grupper har acceptabla resultat men testerna för enheter har flera resultat med tider långt över den fördefinierade idealtiden och vissa även över den absoluta maxtiden.

6.1 Diskussion

Omarbetningen av finansbolag var relativt okomplicerad då funktionaliteten är begränsad. Även resultaten är begränsade men visar godtagbara tider. Arbetet med grupper var mer komplicerat i och med beroendet mellan de två indexen med grupper och enheter. Detta löstes genom användningen av det nästlade indexet ”Nested_groups” och en extra DTO för datahämtningen. Att kombinera olika index till en mer komplex variant fungerade bra i det här sammanhanget men är inte nödvändigtvis en hållbar lösning för mer invecklade sökområden. Även arbetet med enheter gav ett antal utmaningar. Sökområdet behöver data från flera olika index vilket kräver flera olika sökanrop som efteråt behöver kombineras då sökmotorn inte stödjer sammanfogningar på samma sätt som SQL-förfrågningar gör. En lösning var att utöka själva indexet ”Unit” med relevant data från andra index men även denna metod är begränsad för sökområden med fler beroenden.

Detta är ett exempel på dilemmat av en tids-minnes-kompromiss som är vanligt inom datavetenskap och innebär att ett program byter ökad minnesanvändning mot minskad tidsåtgång. Effektivast skulle vara om alla index kunde innehålla all relevant information för det som eftersöks. Den data som i nuläget tas ut ur databasen fås genom att sammafoga data från flera tabeller. Då skulle det inte krävas flera sökningar, över flera index, men istället skulle det ta väldigt mycket lagringsutrymme. Den typen av lösning hade snabbt blivit för kostsam.

Grafen för grupper i Figur 5.1 visar att testresultaten för implementationerna följer varandra och att de sökningar som har fler visade resultat också tar lite längre tid. Grafen i Figur 5.2 visar att den nya implementationen inte klarar av att hålla

nere söktiderna för sökningar över enheter och det blir väldigt tydligt vid de sökningar som visar många resultat. Även vid de mindre sökningarna förekommer en del variationer, vilket tydligast går att observera i grafen för grupper, men det bör tas i beaktning att testen inte utförts lokalt utan både databasen och instansen av Elasticsearch har varit uppsatta på en extern server. Detta är sannolikt en påverkande faktor för påvisade variationer bland svarstiderna.

Något annat som inte bör förbises vid jämförelse av testerna är att den nya implementationen är resultat av ett fyra månader långt examensarbete medan implementationen i produktion har utvecklats och förvaltas av ett utvecklingsteam. Lina har varit i drift i flera år och förbättras ständigt för att optimera bland annat sökeffektivitet. Därför var målet ett implementera en effektivare lösning väldigt optimistiskt för just detta projekt men med mer positiva resultat hade detta kunnat ligga till grund för vidareutveckling som i sin tur mer realistiskt hade kunnat uppnå alla mål.

Projektet har till största del följt den projektplan som sattes upp i planeringsstadiet. Tidsplanen på två månaders arbete innan årsskiftet och två månader efter har följts. Det tog mer tid att knyta ihop arbetet med alla finjusteringar på slutet, att få Lina och backend att passa ihop, och därför tog det mer tid än planerat att få ut resultaten. Skrivandet av rapporten kom inte igång så tidigt som det var planerat men den avsatta tiden var gott tilltagen så det har ändå funnits marginal till att hinna med att så gott som färdigställa den. Sammanlagt har projektet tagit lite drygt 400 timmar, inklusive rapportskrivning.

Även om planen inte var att implementera ett agilt arbetssätt så jobbar Foxways utvecklingsteam agilt med scrum. Något som varit väldigt givande under projektets gång har varit att följa utvecklingsteamets stand-ups, dagliga morgonmöten, som är en del av deras agila arbetssätt. Varje dag har börjat med frågorna ”Vad har hänt sen senast?”, ”Vad är planen för dagen?” och ”Vad finns det för hinder?” vilket varit ett effektivt och uppskattat sätt att strukturera arbetsdagarna.

6.2 Vidareutveckling

I nuläget är det inte aktuellt att fortsätta undersöka möjligheten till implementation av Elasticsearch i Lina. Ett nytt försök skulle kunna göras att sätta upp index som inte är baserade på de tabeller som finns i databasen, utan som delar upp datan på ett sätt som gör den mer lättanvänd.

Referenser

[1] Foxway. (2022). ”Våra tjänster,” [Online]. Tillgänglig: https://www.foxway.com/sv/vara-tjanster/ (hämtad 2022-03-17).

[2] Foxway. (2022). ”Offentlig sektor,” [Online]. Tillgänglig: https://www.foxway.com/sv/vara-tjanster/offentlig-sektor-se/ (hämtad 2022-03-17).

[3] Foxway. (2022). ”Portaler,” [Online]. Tillgänglig: https://www.foxway.com/sv/login/ (hämtad 2022-03-17).

[4] Elastic. (2022). ”Elastic customer stories of all shapes and sizes,” [Online]. Tillgänglig: https://www.elastic.co/customers/success-stories?usecase=enterprise-search,elastic-observability,security-analytics&industry=All (hämtad 2022-03-17).

[5] Elastic. (2022). ”What is Elasticsearch?” [Online]. Tillgänglig: https://www.elastic.co/what-is/elasticsearch (hämtad 2022-03-17).

[6] Apache Lucene. (2022). ”Apache Lucene Core,” [Online]. Tillgänglig: https://lucene.apache.org/core/ (hämtad 2022-03-21).

[7] G. Reback och D. Berman. (2020). ”An Elasticsearch Tutorial: Getting Started,” [Online]. Tillgänglig: https://logz.io/blog/elasticsearch-tutorial/ (hämtad 2022-03-21).

[8] Elastic. (2022). ”Our story,” [Online]. Tillgänglig: https://www.elastic.co/about/history-of-elasticsearch (hämtad 2022-03-17).

[9] S. Goyal. (2020). ”Elasticsearch and its internals working,” [Online]. Tillgänglig: https://medium.com/geekculture/elasticsearch-internals-4c4c9ec077fa (hämtad 2022-03-25).

[10] K. Tan. (2022). ”Basic Concepts,” [Online]. Tillgänglig: http://www.lucenetutorial.com/basic-concepts.html (hämtad 2022-04-03).

[11] json.org. (2022). ”Introducing JSON,” [Online]. Tillgänglig: https://www.json.org/json-en.html (hämtad 2022-04-12).

[12] baeldung. (2021). ”Introduction to Apache Lucene,” [Online]. Tillgänglig: https://www.baeldung.com/lucene (hämtad 2022-04-04).

[13] F. de Villamil. (2019). ”About Lucene,” [Online]. Tillgänglig: https://fdv.github.io/running-elasticsearch-fun-profit/003-about-lucene/003-about-lucene.html (hämtad 2022-04-04).

[14] Elastic. (2022). ”Scalability and resilience: clusters, nodes, and shards,” [Online]. Tillgänglig: https://www.elastic.co/guide/en/elasticsearch/reference/current/scalability.html (hämtad 2022-04-04).

[15] Elastic. (2022). ”What is the ELK Stack?” [Online]. Tillgänglig: https://www.elastic.co/what-is/elk-stack (hämtad 2022-03-17).

[16] jruby m. fl. (2022). ”JRuby,” [Online]. Tillgänglig: https://github.com/jruby/jruby/wiki (hämtad 2022-04-04).
[17] tutorialpoint.com. (2022). ”Logstash - Introduction,” [Online]. Tillgänglig: https://www.tutorialspoint.com/logstash/logstash_introduction.htm (hämtad 2022-04-04).
[18] Elastic. (2022). ”Logstash introduction,” [Online]. Tillgänglig: https://www.elastic.co/guide/en/logstash/current/introduction.html (hämtad 2022-04-04).
[19] Amazon Web Service. (2022). ”Kibana,” [Online]. Tillgänglig: https://aws.amazon.com/opensearch-service/the-elk-stack/kibana/ (hämtad 2022-04-04).
[20] Elastic. (2022). ”What is Kibana?” [Online]. Tillgänglig: https://www.elastic.co/what-is/kibana (hämtad 2022-04-04).
[21] Elastic. (2022). ”Run Elasticsearch API requests,” [Online]. Tillgänglig: https://www.elastic.co/guide/en/kibana/current/console-kibana.html (hämtad 2022-04-04).
[22] Curl. (2022). ”command line tool and library for transferring data with URLs (since 1998),” [Online]. Tillgänglig: https://curl.se/ (hämtad 2022-04-04).
[23] IBM Cloud Education. (2020). ”Application Programming Interface (API),” [Online]. Tillgänglig: https://www.ibm.com/cloud/learn/api (hämtad 2022-04-11).
[24] M. Wyatt. (2022). ”What is an API? A Digestible Definition with API Examples for Ecommerce Owners,” [Online]. Tillgänglig: https://www.bigcommerce.com/blog/what-is-an-api/#what-is-an-api (hämtad 2022-04-11).
[25] Codecademy Team. (2022). ”What is CRUD?” [Online]. Tillgänglig: https://www.codecademy.com/article/what-is-crud (hämtad 2022-04-11).
[26] Codecademy Team. (2022). ”What is REST?” [Online]. Tillgänglig: https://www.codecademy.com/article/what-is-rest (hämtad 2022-04-11).
[27] T. Tam m. fl. (2022). ”The OpenAPI Specification,” [Online]. Tillgänglig: https://github.com/OAI/OpenAPI-Specification/blob/main/README.md (hämtad 2022-04-26).
[28] Swagger. (2021). ”Swagger UI,” [Online]. Tillgänglig: https://swagger.io/tools/swagger-ui/ (hämtad 2022-04-22).
[29] P. Nguyen. (2021). ”Get Started with Swagger UI for API Testing,” [Online]. Tillgänglig: https://www.blazemeter.com/blog/getting-started-with-swagger-ui (hämtad 2022-04-22).
[30] Microsoft. (2022). ”A tour of the C# language,” [Online]. Tillgänglig: https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/ (hämtad 2022-04-27).
[31] Microsoft. (2022). ”What is .NET? Introduction and overview,” [Online]. Tillgänglig: https://docs.microsoft.com/en-us/dotnet/core/introduction (hämtad 2022-04-27).
[32] CsharpSkolan. (2018). ”Introduktion till .NET,” [Online]. Tillgänglig: https://csharpskolan.se/article/introduktion-till-net/ (hämtad 2022-04-27).
[33] Microsoft. (2022). ”An introduction to NuGet,” [Online]. Tillgänglig: https://docs.microsoft.com/en-us/nuget/what-is-nuget (hämtad 2022-04-27).

[34] nuget. (2022). ”What is NuGet?” [Online]. Tillgänglig: https://www.nuget.org/ (hämtad 2022-04-28).

[35] K. T. Steve Gordon. (2021). ”Elasticsearch-net,” [Online]. Tillgänglig: https://github.com/elastic/elasticsearch-net/blob/main/README.md (hämtad 2022-04-28).

[36] Microsoft. (2013). ”Solutions as Containers,” [Online]. Tillgänglig: https://docs.microsoft.com/en-us/previous-versions/visualstudio/visual-studio-2010/df8st53z(v=vs.100)?redirectedfrom=MSDN (hämtad 2022-04-28).

[37] Microsoft. (2016). ”Solutions and Projects in Visual Studio,” [Online]. Tillgänglig: https://docs.microsoft.com/sv-se/previous-versions/visualstudio/visual-studio-2015/ide/solutions-and-projects-in-visual-studio?view=vs-2015&redirectedfrom=MSDN (hämtad 2022-04-28).

[38] Microsoft. (2020). ”Create Data Transfer Objects (DTOs),” [Online]. Tillgänglig: https://docs.microsoft.com/en-us/aspnet/web-api/overview/data/using-web-api-with-entity-framework/part-5 (hämtad 2022-05-05).

[39] DNSstuff.com. (2020). ”MySQL vs. MSSQL - Performance and Main Differences Between Database and Servers,” [Online]. Tillgänglig: https://www.dnsstuff.com/mysql-vs-mssql-performance (hämtad 2022-03-17).

[40] Microsoft. (2022). ”What is SQL Server Management Studio (SSMS)?” [Online]. Tillgänglig: https://docs.microsoft.com/sv-se/sql/ssms/sql-server-management-studio-ssms?view=sql-server-2017 (hämtad 2022-03-17).

[41] Projektledning.se. (2022). ”Vattenfallsmodellen,” [Online]. Tillgänglig: https://projektledning.se/vattenfallsmodellen/ (hämtad 2022-03-16).

A

Appendix 1

Resultat från den implementation som projektet producerat jämfört med originalimplementationen i produktion.

Finansbolag

Test 1:
Full sökning över alla finansbolag med 10 visade resultat.
Antal totala träffar: 24. Antal visade träffar: 10.
Storlek implementation: 2.3 kB. Storlek Produktion: 2.4 kB.
Resultat implementation (ms): 96, 85, 41, 88, 139, 69, 62, 49, 75, 154, 68, 40, 48, 100, 111, 48, 48, 35, 35, 74, 53, 177, 328, 194, 55, 37, 52, 37, 36, 65
Medelvärde: 83 ms. Minsta värde: 35 ms. Största värde: 328 ms.
Resultat produktion (ms): 76, 29, 23, 38, 21, 28, 32, 18, 22, 22, 35, 27, 38, 26, 24, 27, 35, 26, 37, 27, 47, 18, 32, 23, 34, 19, 29, 27, 32, 19
Medelvärde: 30 ms. Minsta värde: 18 ms. Största värde: 76 ms.

Test 2:
Full sökning över alla finansbolag med 50 visade resultat.
Antal totala träffar: 24. Antal visade träffar: 24.
Storlek implementation: 5.5 kB. Storlek produktion: 5.6 kB.
Resultat implementation (ms): 0 75, 92, 69, 45, 46, 125, 78, 57, 41, 81, 188, 51, 112, 81, 74, 61, 77, 42, 37, 81, 67, 67, 36, 149, 50, 60, 90, 50, 71, 61
Medelvärde: 73,8 = 74. Minsta värde: 37. Största värde: 188.
Resultat produktion (ms): 63, 47, 27, 19, 20, 30, 29, 22, 24, 48, 70, 18, 49, 31, 39, 17, 41, 108, 19, 32, 71, 86, 27, 26, 20, 68, 68, 21, 18, 30, 62, 46
Medelvärde: 43. Minsta värde: 17. Största värde: 108.

Test nr	IMP Avg	IMP Min/Max	PROD Avg	PROD Min/Max
1	83 ms	35 ms/328 ms	30 ms	18 ms/76 ms
2	74 ms	37 ms/188 ms	43 ms	17 ms/108 ms

Grupper

Test 1:
Full sökning över alla grupper med 10 visade resultat.
Antal totala träffar: 275. Antal visade träffar: 10.
Storlek implementation: 14.2 kB. Storlek produktion: 8.8 kB.
Resultat implementation (ms): 74, 70, 125, 51, 67, 55, 68, 73, 87, 61, 213, 54, 52, 54, 91, 60, 63, 85, 59, 60, 88, 115, 79, 76, 118, 115, 62, 60, 61, 56
Medelvärde: 78. Minsta värde: 51. Största värde: 213.
Resultat produktion (ms): 73, 303, 202, 158, 61, 61, 109, 79, 117, 84, 57, 59, 103, 91, 79, 52, 110, 25, 69, 53, 45, 68, 86, 141, 95, 78, 64, 153, 94, 59
Medelvärde: 94. Minsta värde: 25. Största värde: 303.

Test 2:
Full sökning över alla grupper med 100 visade resultat.
Antal totala träffar: 275. Antal visade träffar: 100.
Storlek implementation: 264 kB. Storlek produktion: 157 kB.
Resultat implementation (ms): 143, 380, 164, 419, 391, 86, 289, 173, 96, 76, 96, 128, 125, 111, 150, 115, 153, 345, 163, 125, 86, 125, 246, 711, 239, 114, 130, 125, 101, 208
Medelvärde: 194. Minsta värde: 76. Största värde: 711.
Resultat produktion (ms): 138, 100, 124, 83, 144, 109, 77, 167, 103, 190, 91, 81, 108, 98, 109, 163, 58, 127, 160, 97, 91, 67, 91, 141, 64, 100, 82, 143, 93, 155
Medelvärde: 112. Minsta värde: 58. Största värde: 190.

Test 3:
Full sökning över alla grupper med 250 visade resultat.
Antal totala träffar: 275. Antal visade träffar: 250.
Storlek implementation: 594 kB. Storlek produktion: 351 kB.
Resultat implementation (ms): 216, 252, 771, 963, 204, 172, 285, 208, 228, 183, 204, 175, 126, 131, 151, 134, 219, 167, 183, 157, 230, 850, 316, 213, 487, 406, 992, 185, 188, 253
Medelvärde: 308. Minsta värde: 131. Största värde: 992.
Resultat produktion (ms): 148, 179, 153, 190, 176, 141, 153, 131, 125, 102, 135, 128, 134, 260, 222, 137, 117, 195, 112, 91, 77, 89, 82, 112, 87, 89, 67, 81, 141, 95
Medelvärde: 132. Minsta värde: 67. Största värde: 260.

Test 4:
Frisök, ”kommun”, med 10 visade resultat
Antal totala träffar: 121. Antal visade träffar: 10. Storlek implementation: 22.1 kB.
Storlek produktion: 13.3 kB.
Resultat implementation (ms): 124, 75, 89, 93, 97, 167, 75, 75, 72, 135, 95, 68, 85, 58, 61, 88, 71, 89, 81, 107, 84, 85, 131, 80, 154, 108, 86, 86, 97, 68
Medelvärde: 93. Minsta värde: 61. Största värde: 135.
Resultat produktion (ms): 250, 59, 123, 78, 71, 61, 58, 81, 231, 56, 88, 129, 69, 54, 83, 57, 68, 70, 66, 80, 99, 100, 361, 85, 84, 55, 86, 57, 70, 72
Medelvärde: 97. Minsta värde: 54. Största värde: 361.

Test 5:
Frisök, ”kommun”, med 100 visade resultat.
Antal totala träffar: 121. Antal visade träffar: 100.
Storlek implementation: 311 kB. Storlek produktion: 183 kB.
Resultat implementation (ms): 128, 152, 116, 171, 117, 147, 93, 102, 191, 136, 134, 130, 120, 120, 91, 113, 101, 101, 150, 136, 207, 137, 114, 123, 115, 117, 153, 147, 129, 162
Medelvärde: 132. Minsta värde: 91. Största värde: 207.
Resultat produktion (ms): 330, 113, 125, 103, 76, 94, 124, 166, 135, 203, 190, 105, 82, 70, 134, 229, 303, 99, 158, 79, 225, 281, 255, 80, 256, 293, 226, 119, 74, 131
Medelvärde: 162. Minsta värde: 70. Största värde: 330.

Test 6:
Frisök, ”kommun”, med 250 visade resultat.
Antal totala träffar: 121. Antal visade träffar: 121.
Storlek implementation: 373 kB. Storlek produktion: 220 kB.
Resultat implementation (ms): 136, 135, 269, 132, 134, 120, 195, 120, 109, 128, 167, 101, 166, 109, 136, 162, 218, 138, 129, 186, 156, 244, 140, 154, 105, 142, 140, 143, 118, 134
Medelvärde: 149. Minsta värde: 101. Största värde: 269.
Resultat produktion (ms): 233, 370, 146, 140, 216, 89, 370, 134, 144, 130, 145, 165, 102, 253, 238, 102, 138, 104, 121, 253, 156, 146, 194, 94, 127, 266, 335, 240, 147, 135
Medelvärde: 181. Minsta värde: 89. Största värde: 370.

Test 7:
Frisök, ”Europa”, med 10 visade resultat.
Antal totala träffar: 0. Antal visade träffar: 0. Storlek implementation: 80 B. Storlek produktion: 80 kB.
Resultat implementation (ms): 62, 101, 81, 87, 115, 74, 99, 49, 64, 65, 61, 71, 48, 128, 68, 53, 57, 133, 58, 61, 67, 75, 53, 69, 96, 49, 92, 66, 183, 70
Medelvärde: 79. Minsta värde: 48. Största värde: 183.
Resultat produktion (ms): 96, 70, 75, 53, 149, 50, 81, 131, 117, 208, 141, 110, 87, 71, 105, 55, 130, 32, 73, 139, 98, 88, 161, 124, 150, 85, 84, 62, 133, 51
Medelvärde: 100. Minsta värde: 32. Största värde: 208.

Test 8:
Frisök, ”Europa”, med 250 visade resultat.
Antal totala träffar: 0. Antal visade träffar: 0.
Storlek implementation: 80 B. Storlek produktion: 80 kB.
Resultat implementation (ms): 63, 153, 66, 56, 53, 65, 68, 75, 92, 59, 52, 70, 71, 51, 58, 69, 52, 67, 113, 56, 104, 70, 54, 79, 73, 54, 52, 44, 45, 66
Medelvärde: 68. Minsta värde: 44. Största värde: 153 .
Resultat produktion (ms): 84, 68, 85, 63, 83, 100, 63, 126, 128, 107, 115, 105, 110, 73, 66, 60, 77, 99, 80, 66, 231, 69, 100, 63, 85, 182, 59, 60, 85, 67

Medelvärde: 92. Minsta värde: 59. Största värde: 231.

Test 9:
Sök gruppnamn, ”stad”, med 10 visade resultat.
Antal totala träffar: 19 (21). Antal visade träffar: 10. Storlek implementation: 134 kB. Storlek produktion: 66.7 kB.
Resultat implementation (ms): 74, 71, 122, 61, 58, 63, 68, 67, 67, 64, 60, 89, 164, 133, 63, 103, 62, 62, 71, 69, 69, 63, 71, 66, 73, 73, 58, 79, 75, 71, 68
Medelvärde: 79. Minsta värde: 62. Största värde: 133.
Resultat produktion (ms): 216, 110, 143, 94, 145, 103, 129, 111, 213, 77, 133, 59, 201, 133, 125, 100, 142, 107, 62, 106, 123, 33, 100, 46, 74, 32, 37, 102, 94, 110
Medelvärde: 109. Minsta värde: 32. Största värde: 216.

Test 10:
Sök gruppnamn, ”stad”, med 100 visade resultat.
Antal totala träffar: 19 (21). Antal visade träffar: 19.
Storlek implementation: 163 kB. Storlek produktion: 95.9 kB.
Resultat implementation (ms): 91, 125, 110, 105, 87, 90, 77, 89, 337, 101, 198, 87, 166, 110, 97, 105, 299, 139, 76, 82, 83, 94, 114, 114, 207, 104, 142, 103, 67, 85
Medelvärde: 120. Minsta värde: 67. Största värde: 337.
Resultat produktion (ms): 104, 43, 84, 85, 104, 85, 170, 169, 123, 114, 171, 115, 89, 43, 153, 42, 155, 52, 115, 127, 117, 127, 111, 46, 93, 70, 115, 101, 126, 144
Medelvärde: 106. Minsta värde: 42. Största värde: 171.

Test 11:
Sök gruppnamn, ”stad”, med 250 visade resultat.
Antal totala träffar: 19 (21). Antal visade träffar: 19.
Storlek implementation: 163 kB. Storlek produktion: 95.9 kB.
Resultat implementation (ms): 87, 99, 92, 92, 151, 101, 101, 104, 100, 76, 118, 160, 113, 155, 82, 104, 186, 97, 132, 103, 96, 98, 94, 177, 110, 181, 99, 86, 100, 124
Medelvärde: 114. Minsta värde: 76. Största värde: 186.
Resultat produktion (ms): 176, 188, 304, 204, 129, 114, 112, 143, 75, 56, 98, 213, 92, 79, 158, 52, 98, 68, 101, 270, 157, 65, 127, 170, 95, 101, 113, 97, 73, 163
Medelvärde: 130. Minsta värde: 53. Största värde: 304.

Test 12:
Sök enhetsnamn, ”skola”, med 10 visade resultat.
Antal totala träffar: 74 (156). Antal visade träffar: 10.
Storlek implementation: 68 kB. Storlek produktion: 15.3 kB.
Resultat implementation (ms): 89, 78, 70, 96, 68, 97, 71, 69, 255, 87, 97, 96, 224, 94, 85, 110, 284, 97, 77, 88, 82, 89, 82, 85, 78, 81, 189, 70, 64, 97
Medelvärde: 102. Minsta värde: 64. Största värde: 284.
Resultat produktion (ms): 104, 67, 63, 107, 173, 50, 100, 85, 205, 127, 107, 90, 79, 108, 75, 89, 207, 95, 81, 121, 133, 68, 130, 95, 85, 65, 113, 86, 155, 43
Medelvärde: 104. Minsta värde: 43. Största värde: 207.

Test 13:
Sök enhetsnamn, ”skola”, med 100 visade resultat.
Antal totala träffar: 74 (156). Antal visade träffar: 74 (100).
Storlek implementation: 424 kB. Storlek produktion: 254 kB.
Resultat implementation (ms): 156, 153, 144, 100, 102, 159, 135, 129, 125, 130, 129, 177, 118, 115, 159, 154, 165, 153, 128, 174, 231, 142, 209, 123, 127, 183, 126, 135, 154, 142
Medelvärde: 138. Minsta värde: 100. Största värde: 209.
Resultat produktion (ms): 170, 159, 338, 152, 411, 98, 90, 99, 99, 78, 155, 85, 84, 93, 155, 25, 374, 252, 144, 232, 290, 298, 314, 290, 153, 108, 265, 109, 285, 248
Medelvärde: 188. Minsta värde: 25. Största värde: 411.

Test 14:
Sök enhetsnamn, ”skola”, med 250 Visade resultat.
Antal totala träffar: 74 (156). Antal visade träffar: 74 (156).
Storlek implementation: 424 kB. Storlek produktion: 340 kB.
Resultat implementation (ms): 114, 172, 177, 144, 150, 180, 212, 131, 165, 142, 170, 115, 165, 133, 151, 137, 124, 123, 139, 168, 229, 150, 145, 142, 147, 154, 122, 118, 163, 128
Medelvärde: 160. Minsta värde: 114. Största värde: 229.
Resultat produktion (ms): 192, 159, 96, 102, 215, 110, 548, 127, 128, 181, 126, 99, 330, 296, 142, 105, 196, 440, 286, 146, 385, 279, 228, 458, 351, 101, 275, 118, 183, 236
Medelvärde: 221. Minsta värde: 96. Största värde: 548.

Test 15:
Sök gruppnamn, ”stad”, enhetsnamn, ”förskola”, med 10 Visade resultat.
Antal totala träffar: 5. Antal visade träffar: 5.
Storlek implementation: 133 kB. Storlek produktion: 77.4 kB.
Resultat implementation (ms): 108, 92, 101, 326, 110, 125, 117, 88, 102, 105, 89, 66, 290, 78, 167, 163, 119, 81, 117, 143, 82, 112, 73, 128, 95, 140, 88, 72, 108, 117
Medelvärde: 120. Minsta värde: 66. Största värde: 326.
Resultat produktion (ms): 71, 98, 80, 65, 112, 77, 79, 119, 52, 60, 201, 97, 116, 101, 122, 133, 102, 64, 88, 104, 88, 100, 74, 59, 97, 57, 58, 95, 72, 80
Medelvärde: 91. Minsta värde: 52. Största värde: 201.

Test 16:
Sök gruppnamn, ”stad”, enhetsnamn, ”förskola”, med 250 Visade resultat.
Antal totala träffar: 5. Antal visade träffar: 5.
Storlek implementation: 133 kB. Storlek produktion: 77.4 kB.
Resultat implementation (ms): 83, 64, 119, 85, 91, 76, 131, 99, 85, 96, 141, 88, 99, 77, 95, 114, 119, 151, 101, 78, 80, 94, 91, 138, 116, 89, 101, 89, 116, 112
Medelvärde: 101. Minsta värde: 64. Största värde: 151.
Resultat produktion (ms): 50, 86, 116, 104, 78, 69, 75, 93, 91, 68, 105, 77, 71, 75, 63, 117, 68, 67, 87, 83, 63, 96, 75, 78, 68, 138, 80, 96, 53, 88
Medelvärde: 83. Minsta värde: 50. Största värde: 138.

Test nr	IMP Avg	IMP Min/Max	PROD Avg	PROD Min/Max
1	76 ms	51 ms/213 ms	94 ms	25 ms/303 ms
2	194 ms	76 ms/711 ms	112 ms	58 ms/190 ms
3	308 ms	131 ms/992 ms	132 ms	67 ms/260 ms
4	93 ms	61 ms/135 ms	97 ms	54 ms/361 ms
5	132 ms	91 ms/207 ms	162 ms	70 ms/330 ms
6	149 ms	101 ms/269 ms	181 ms	89 ms/370 ms
7	79 ms	48 ms/183 ms	100 ms	32 ms/208 ms
8	68 ms	44 ms/153 ms	92 ms	59 ms/231 ms
9	79 ms	62 ms/133 ms	109 ms	32 ms/216 ms
10	120 ms	67 ms/337 ms	106 ms	42 ms/171 ms
11	114 ms	76 ms/186 ms	130 ms	53 ms/304 ms
12	102 ms	64 ms/284 ms	104 ms	43 ms/207 ms
13	138 ms	100 ms/209 ms	188 ms	25 ms/411 ms
14	160 ms	114 ms/229 ms	221 ms	96 ms/548 ms
15	120 ms	66 ms/326 ms	91 ms	52 ms/201 ms
16	101 ms	64 ms/151 ms	83 ms	50 ms/138 ms

Enheter

Test 1:
Full sökning över alla enheter med 10 resultat.
Antal totala träffar: 3087. Antal visade träffar: 10.
Storlek implementation: 6.3 kB. Storlek produktion: 6.1 kB.
Resultat implementation (ms): 229, 285, 206, 245, 294, 215, 231, 274, 307, 320, 323, 303, 343, 307, 183, 333, 231, 202, 229, 244, 265, 257, 288, 254, 190, 345, 360, 246, 229, 342
Medel: 269, Min: 183, Max: 360
Resultat produktion (ms): 160, 64, 44, 54, 47, 51, 73, 63, 55, 50, 178, 100, 44, 42, 52, 59, 96, 65, 38, 57, 55, 151, 46, 78, 57, 50, 75, 33, 64, 97 Medelvärde: 70. Minsta värde: 33. Största värde: 178.

Test 2:
Full sökning över alla enheter med 100 visade resultat.
Antal totala träffar: 3087. Antal visade träffar: 100.
Storlek implementation: 74.8 kB. Storlek produktion: 71.4 kB.
Resultat implementation (s): 1.81, 1.77, 1.68, 1.65, 1.84, 1.71, 1.73, 1.80, 1.83, 2.47 1.79, 1.73, 1.63, 1.70, 1.82, 1.67, 1.64, 1.66, 1.65, 1.63, 1.72, 1.72, 1.70, 1.64, 1.75, 1.81, 1.83, 1.78, 1.80, 1.72
Medel: 1.76, Min: 1.63, Max: 2.47
Resultat produktion (ms): 51, 36, 55, 61, 63, 85, 60, 76, 40, 74, 60, 43, 82, 59, 51, 46, 49, 58, 234, 76, 62, 70, 63, 93, 56, 64, 56, 54, 55, 109
Medelvärde: 70. Minsta värde: 33. Största värde: 178.

Test 3:
Full sökning över alla enheter med 250 resultat.
Antal totala träffar: 3087. Antal Visade träffar: 250.
Storlek implementation: 180 kB. Storlek produktion: 174 kB.
Resultat implementation (s): 4.06, 4.27, 3.86, 4.06, 3.99, 3.98, 4.00, 4.06, 4.18, 4.30, 4.25, 4.03, 4.38, 4.18, 4.16, 3.99, 4.50, 4.08, 4.23, 3.86, 4.03, 4.26, 4.24, 4.26, 3.99, 4.03, 4.32, 4.61, 4.17, 3.65
Medel: 4.13, Min: 3.65, Max: 4.61
Resultat produktion (ms): 136, 83, 111, 102, 86, 254, 79, 87, 113, 53, 77, 52, 54, 58, 48, 52, 60, 45, 93, 91, 51, 46, 49, 52, 57, 49, 54, 47, 56, 50
Medelvärde: 75. Minsta värde: 45. Största värde: 254.

Test 4:
Frisök, ”kommun”, med 10 visade resultat.
Antal totala träffar: 1660. Antal visade träffar: 10.
Storlek implementation: 6.9 kB. Storlek produktion: 8.1 kB.
Resultat implementation (ms): 498, 312, 550, 484, 583, 553, 536, 213, 568, 586, 630, 500, 599, 764, 319, 529, 529, 348, 482, 375, 268, 554, 552, 515, 500, 424, 603, 550, 539, 514
Medel: 499, Min: 213, Max: 764
Resultat produktion (ms): 71, 86, 91, 148, 70, 68, 70, 70, 71, 69, 90, 70, 70, 71,

70, 69, 72, 89, 71, 68, 74, 74, 70, 90, 76, 73, 71, 80, 69, 75 Medelvärde: 77. Minsta värde: 68. Största värde: 148.

Test 5:
Frisök, ”kommun”, med 100 visade resultat.
Antal totala träffar: 1660. Antal visade träffar: 100.
Storlek implementation: 75.7 kB. Storlek produktion: 71.7 kB.
Resultat implementation (s): 1.88, 1.77, 1.73, 1.64, 1.75, 1.57, 1.83, 1.93, 1.65, 1.73, 1.59, 1.59, 1.95, 1.74, 1.88, 1.81, 1.90, 1.67, 1.76, 1.77, 1.68, 1.65, 1.66, 1.68, 1.67, 1.72, 1.79, 1.78, 1.72, 1.78
Medel: 1.74, Min: 1.57, Max: 1.95
Resultat produktion (ms): 254, 274, 238, 99, 77, 78, 82, 76, 92, 104, 78, 94, 239, 245, 92, 81, 76, 78, 375, 86, 270, 234, 211, 85, 75, 103, 79, 78, 73, 82
Medelvärde: 140. Minsta värde: 73. Största värde: 375.

Test 6:
Frisök, ”kommun”, med 250 visade resultat.
Antal totala träffar: 1660. Antal visade träffar: 250.
Storlek implementation: 185 kB. Storlek produktion: 177 kB.
Resultat implementation (s): 3.87, 4.15, 4.08, 3.90, 4.02, 3.81, 4.07, 4.01, 3.71, 4.14, 4.29, 4.24, 4.17, 4.53, 4.21, 3.81, 4.09, 3.71, 4.04, 4.31, 4.20, 3.69, 4.64, 4.27, 4.18, 4.10, 4.17, 4.12, 4.14, 4.37
Medel: 4.10, Min: 3.69, Max: 4.64
Resultat produktion (ms): 307, 94, 297, 101, 326, 92, 87, 99, 100, 90, 329, 106, 89, 88, 84, 86, 321, 92, 82, 82, 100, 84, 92, 125, 86, 88, 314, 96, 105, 87
Medelvärde: 138. Minsta värde: 82. Största värde: 329.

Test 7:
Frisök, ”Europa”, med 10 visade resultat.
Antal totala träffar: 0. Antal visade träffar: 0.
Storlek implementation: 79 B. Storlek produktion: 79 B.
Resultat implementation (ms): 55, 80, 58, 71, 45, 52, 62, 78, 74, 67, 96, 88, 65, 107, 59, 87, 160, 44, 52, 58, 64, 57, 58, 92, 50, 86, 60, 55, 88, 61
Medel: 71, Min: 44, Max: 160
Resultat produktion (ms): 77, 77, 90, 81, 75, 83, 74, 73, 86, 78, 109, 77, 76, 78, 76, 77, 78, 94, 93, 105, 75, 86, 74, 71, 93, 76, 79, 77, 72, 78
Medelvärde: 81. Minsta värde: 71. Största värde: 109.

Test 8:
Frisök, ”Europa”, med 250 visade resultat.
Antal totala träffar: 0. Antal visade träffar: 0.
Storlek implementation: 82 B. Storlek produktion: 82 B.
Resultat implementation (ms): 64, 46, 45, 147, 56, 74, 49, 81, 42, 62, 60, 105, 73, 51, 53, 54, 52, 53, 66, 64, 68, 83, 49, 50, 69, 46, 52, 67, 67, 62
Medel: 64, Min: 42, Max: 147
Resultat produktion (ms): 111, 79, 95, 86, 78, 77, 79, 84, 72, 78, 84, 84, 82, 77,

81, 90, 88, 77, 87, 98, 79, 73, 83, 81, 83, 88, 111, 79, 78, 83
Medelvärde: 84. Minsta värde: 72. Största värde: 111.

Test 9:
Sök enhetsnamn, ”skola”, med 10 visade resultat.
Antal totalt träffar: 338 (2074). Antal visade träffar: 10.
Storlek implementation: 8.9 kB. Storlek produktion: 7.7 kB.
Resultat implementation (ms): 198, 222, 209, 210, 201, 204, 192, 195, 207, 210, 206, 217, 220, 215, 250, 252, 207, 206, 190, 204, 198, 206, 203, 211, 206, 199, 241, 215, 199, 209
Medel: 210, Min: 190, Max: 252
Resultat produktion (ms): 38, 44, 53, 42, 71, 115, 51, 50, 43, 46, 51, 51, 54, 61, 39, 48, 145, 105, 42, 57, 47, 42, 44, 63, 48, 120, 45, 41, 48, 42
Medelvärde: 58. Minsta värde: 38. Största värde: 145.

Test 10:
Sök enhetsnamn, ”skola”, med 100 visade resultat.
Antal totala träffar: 338. Antal visade träffar: 100.
Storlek implementation: 78.4 kB. Storlek produktion: 70.7 kB.
Resultat implementation (s): 1.54, 1.45, 1.58, 1.49, 1.50, 1.48, 1.46, 1.53, 1.46, 1.50, 1.44, 1.51, 1.52, 1.46, 1.50, 1.54, 1.48, 1.55, 1.50, 1.57, 1.57, 1.58, 1.49, 1.48, 1.53, 1.56, 1.85, 1.53, 1.50, 1.57
Medel: 1.52, Min: 1.44, Max: 1.85
Resultat produktion (ms): 51, 56, 52, 58, 57, 49, 56, 46, 52, 119, 49, 56, 56, 50, 49, 55, 53, 58, 60, 49, 49, 50, 50, 58, 64, 59, 54, 54, 57, 89
Medelvärde: 57. Minsta värde: 46. Största värde: 119.

Test 11:
Sök enhetsnamn, ”skola”, med 250 visade resultat.
Antal totala träffar: 338. Antal visade träffar: 250.
Storlek implementation: 196 kB. Storlek produktion: 183 kB.
Resultat implementation (s): 3.65, 3.77, 3.57, 3.54, 4.04, 3.66, 3.76, 3.67, 4.08, 3.73, 3.84, 3.77, 3.80, 3.70, 3.72, 3.61, 3.53, 3.55, 3.58, 3.64, 3.58, 3.65, 3.77, 3.74, 3.94, 3.77, 3.60, 3.63, 3.55, 3.68
Medel: 3.70, Min: 3.53, Max: 4.08
Resultat produktion (ms): 79, 63, 62, 155, 305, 85, 92, 87, 66, 113, 88, 60, 122, 67, 63, 110, 63, 97, 76, 104, 59, 83, 59, 97, 79, 151, 63, 84, 77, 73
Medelvärde: 93. Minsta värde: 59. Största värde: 305.

Test 12:
Sök gruppnamn, ”stad”, med 10 visade resultat.
Antal totala träffar: 803 (813). Antal visade träffar: 10.
Storlek implementation: 7.3 kB. Storlek produktion: 7.9 kB.
Resultat implementation (ms): 186, 186, 207, 182, 352, 177, 180, 201, 204, 175, 196, 200, 228, 202, 189, 218, 187, 192, 189, 208, 192, 183, 190, 197, 182, 194, 191, 184, 203, 197

Medel: 199, Min: 175, Max: 352
Resultat produktion (ms): 148, 58, 49, 48, 37, 47, 41, 44, 48, 38, 250, 105, 79, 94, 77, 187, 77, 43, 135, 101, 93, 97, 81, 54, 101, 70, 92, 55, 137, 248
Medelvärde: 91. Minsta värde: 37. Största värde: 250.

Test 13:
Sök gruppnamn, ”stad”, med 100 visade resultat.
Antal totala träffar: 803. Antal visade träffar: 100.
Storlek implementation: 68.5 kB. Storlek produktion: 83.2 kB.
Resultat implementation (s): 1.42, 1.43, 1.41, 1.37, 1.46, 1.49, 1.44, 1.43, 1.40, 1.51, 1.46, 1.42, 1.34, 1.38, 1.41, 1.37, 1.33, 1.40, 1.38, 1.37, 1.42, 1.41, 1.36, 1.41, 1.45, 1.45, 1.40, 1.38, 1.53, 1.38
Medel: 1.41, Min: 1.33, Max: 1.53
Resultat produktion (ms): 92, 92, 93, 89, 88, 60, 146, 79, 588, 110, 96, 129, 94, 72, 93, 103, 156, 128, 138, 82, 92, 117, 93, 108, 132, 71, 103, 66, 129, 76
Medelvärde: 117. Minsta värde: 60. Största värde: 588.

Test 14:
Sök gruppnamn, ”stad”, med 250 visade resultat.
Antal totala träffar: 803. Antal visade träffar: 250.
Storlek implementation: 168 kB. Storlek produktion: 201 kB.
Resultat implementation (s): 3.32, 3.39, 3.43, 3.52, 3.27, 3.42, 3.38, 3.32, 3.31, 3.57, 3.33, 3.35, 3.31, 3.61, 3.32, 3.56, 3.43, 3.54, 3.29, 3.30, 3.40, 3.55, 3.38, 3.45, 3.49, 3.60, 3.44, 3.36, 3.37, 3.32
Medel: 3.41, Min: 3.27, Max: 3.61
Resultat produktion (ms): 148, 71, 68, 111, 77, 70, 105, 56, 88, 106, 76, 88, 77, 217, 135, 193, 122, 78, 242, 75, 76, 93, 72, 113, 81, 77, 123 85, 91, 116
Medelvärde: 97. Minsta värde: 56. Största värde: 242.

Test 15:
Sök leveransadress, ”Förskoleförvaltningen”, med 10 visade resultat.
Antal totala träffar: 137. Antal visade träffar: 10.
Storlek implementation: 5.8 kB. Storlek produktion: 5.8 kB.
Resultat implementation (ms): 187, 198, 170, 166, 171, 189, 170, 181, 181, 220, 172, 179, 167, 162, 183, 175, 159, 167, 172, 181, 177, 173, 180, 178, 181, 179, 177, 167, 176, 171
Medel: 177, Min: 159, Max: 220
Resultat produktion (ms): 55, 76, 109, 61, 61, 52, 58, 77, 69, 65, 99, 105, 136, 69, 95, 84, 63, 65, 84, 76, 69, 75, 120, 145, 39, 55, 47, 75, 73, 100 Medelvärde: 79. Minsta värde: 39. Största värde: 145.

Test 16:
Sök leveransadress, ”Förskoleförvaltningen”, med 100 visade resultat.
Antal totala träffar: 137. Antal visade träffar: 100.
Storlek implementation: 57.6 kB. Storlek produktion: 57.0 kB.
Resultat implementation (s): 1.37, 1.28, 1.24, 1.26, 1.35, 1.60, 1.27, 1.36, 1.29,

1.34, 1.27, 1.26, 1.31, 1.27, 1.26, 1.28, 1.34, 1.28, 1.25, 1.27, 1.29, 1.28, 1.19, 1.23, 1.31, 1.28, 1.34, 1.29, 1.27, 1.23
Medel: 1.30, Min: 1.19, Max: 1.60
Resultat produktion (ms): 49, 38, 77, 64, 63, 46, 60, 72, 54, 109, 53, 52, 56, 103, 75, 58, 65, 53, 39, 65, 50, 53, 66, 67, 72, 53, 86, 122, 108, 56
Medelvärde: 66. Minsta värde: 38. Största värde: 122.

Test 17:
Sök leveransadress, ”Förskoleförvaltningen”, med 250 visade resultat.
Antal totala träffar: 137. Antal visade träffar: 137.
Storlek implementation: 78.8 kB. Storlek produktion: 77.7 kB.
Resultat implementation (s): 1.79, 1.78, 1.98, 1.68, 1.77, 1.71, 1.81, 1.67, 1.74, 1.74, 1.72, 1.73, 1.68, 1.69, 1.66, 1.73, 1.73, 1.76, 1.72, 1.70, 1.66, 1.70, 1.75, 2.00, 1.72, 1.75, 1.68, 1.77, 1.65, 1.81
Medel: 1.74, Min: 1.65, Max: 2.00
Resultat produktion (ms): 56, 63, 56, 105, 56, 145, 74, 66, 99, 57, 51, 56, 54, 56, 61, 83, 59, 53, 85, 64, 53, 68, 60, 67, 77, 91, 93, 67, 151, 53
Medelvärde: 73. Minsta värde: 51. Största värde: 151.

Test 18:
Sök enhetsnamn, ”förskola”, gruppnamn, ”stad”, leveransadress, ”Förskoleförvaltningen”, med 10 visade resultat.
Antal totala träffar: 129. Antal visade träffar: 10.
Storlek implementation: 5.9 kB. Storlek produktion: 5.8 kB.
Resultat implementation (ms): 436, 294, 280, 270, 366, 299, 296, 273, 276, 435, 287, 299, 307, 376, 339, 317, 261, 236, 345, 338, 294, 269, 380, 257, 272, 210, 275, 228, 333, 269
Medelvärde: 304. Minsta värde: 210. Största värde: 436.
Resultat produktion (ms): 74, 46, 115, 71, 44, 73, 46, 103, 64, 124, 61, 68, 72, 63, 57, 63, 57, 121, 46, 42, 57, 45, 147, 60, 64, 92, 64, 60, 35, 59
Medelvärde: 70. Minsta värde: 35. Största värde: 147.

Test 19:
Sök enhetsnamn, ”förskola”, gruppnamn, ”stad”, leveransadress, ”Förskoleförvaltningen”, med 100 visade resultat.
Antal totala träffar: 129. Antal visade träffar: 100.
Storlek implementation: 57.6 kB. Storlek produktion: 57.0 kB.
Resultat implementation (s): 1.43, 1.60, 1.51, 1.85, 1.51, 1.45, 1.57, 1.47, 1.36, 1.51, 1.36, 1.49, 1.43, 1.46, 1.56, 1.49, 1.51, 1.57, 1.60, 1.44, 1.37, 1.40, 1.45, 1.46, 1.56, 1.45, 1.49, 1.42, 1.53, 1.37
Medelvärde: 1.49. Minsta värde: 1.36. Största värde: 1.85.
Resultat produktion (ms): 60, 52, 61, 54, 56, 87, 54, 73, 70, 60, 61, 57, 64, 43, 98, 60, 110, 54, 52, 63, 53, 59, 61, 60, 120, 87, 51, 64, 97, 52
Medelvärde: 66. Minsta värde: 43. Största värde: 120.

Test 20:
Sök enhetsnamn, ”förskola”, gruppnamn, ”stad”, leveransadress, ”Förskoleförvaltningen”, med 250 visade resultat.
Antal totala träffar: 129. Antal visade träffar: 129.
Storlek implementation: 74.3 kB. Storlek produktion: 73.2 kB.
Resultat implementation (ms): 2.04, 1.70, 1.86, 1.99, 1.83, 1.74, 1.81, 1.90, 1.75, 2.00, 1.92, 1.80, 1.80, 1.82, 1.84, 2.10, 2.14, 2.14, 2.35, 2.14, 2.20, 2.08, 2.18, 2.17, 1.69, 1.89, 1.79, 1.80, 1.67, 1.73
Medelvärde: 1.93. Minsta värde: 1.67. Största värde: 2.35.
Resultat produktion (ms): 58, 59, 64, 89, 64, 122, 60, 55, 65, 80, 61, 54, 63, 50, 50, 70, 57, 69, 109, 62, 119, 73, 83, 56, 72, 69, 52, 69, 70, 68
Medelvärde: 70. Minsta värde: 50. Största värde: 122.

Test nr	IMP Avg	IMP Min/Max	PROD Avg	PROD Min/Max
1	269 ms	183 ms/360 ms	70 ms	33 ms/178 ms
2	1.76 s	1.63 s/2.47 s	70 ms	33 ms/178 ms
3	4.13 s	3.65 s/4.61 s	75 ms	45 ms/254 ms
4	499 ms	213 ms/764 ms	77 ms	68 ms/148 ms
5	1.74 s	1.57 s/1.95 s	140 ms	73 ms/375 ms
6	4.10 s	3.65 s/4.64 s	138 ms	82 ms/329 ms
7	71 ms	44 ms/160 ms	81 ms	71 ms/109 ms
8	64 ms	42 ms/147 ms	84 ms	72 ms/111 ms
9	210 ms	190 ms/252 ms	58 ms	38 ms/145 ms
10	1.52 s	1.44 s/1.85 s	57 ms	46 ms/119 ms
11	3.70 s	3.53 s/4.08 s	93 ms	59 ms/305 ms
12	199 ms	175 ms/352 ms	91 ms	37 ms/250 ms
13	1.41 s	1.33 s/1.53 s	117 ms	60 ms/588 ms
14	3.41 s	3.27 s/3.61 s	97 ms	56 ms/242 ms
15	177 ms	159 ms/220 ms	79 ms	39 ms/145 ms
16	1.30 s	1.19 s/1.60 s	66 ms	38 ms/122 ms
17	1.74 s	1.65 s/2.00 s	73 ms	51 ms/151 ms
18	304 ms	210 ms/436 ms	70 ms	35 ms/147 ms
19	1.49 s	1.36 s/1.85 s	66 ms	43 ms/120 ms
20	1.93 s	1.67 s/2.35 s	70 ms	50 ms/122 ms

INSTITUTIONEN FÖR DATA- OCH INFORMATIONSTEKNIK
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige
www.chalmers.se

CHALMERS