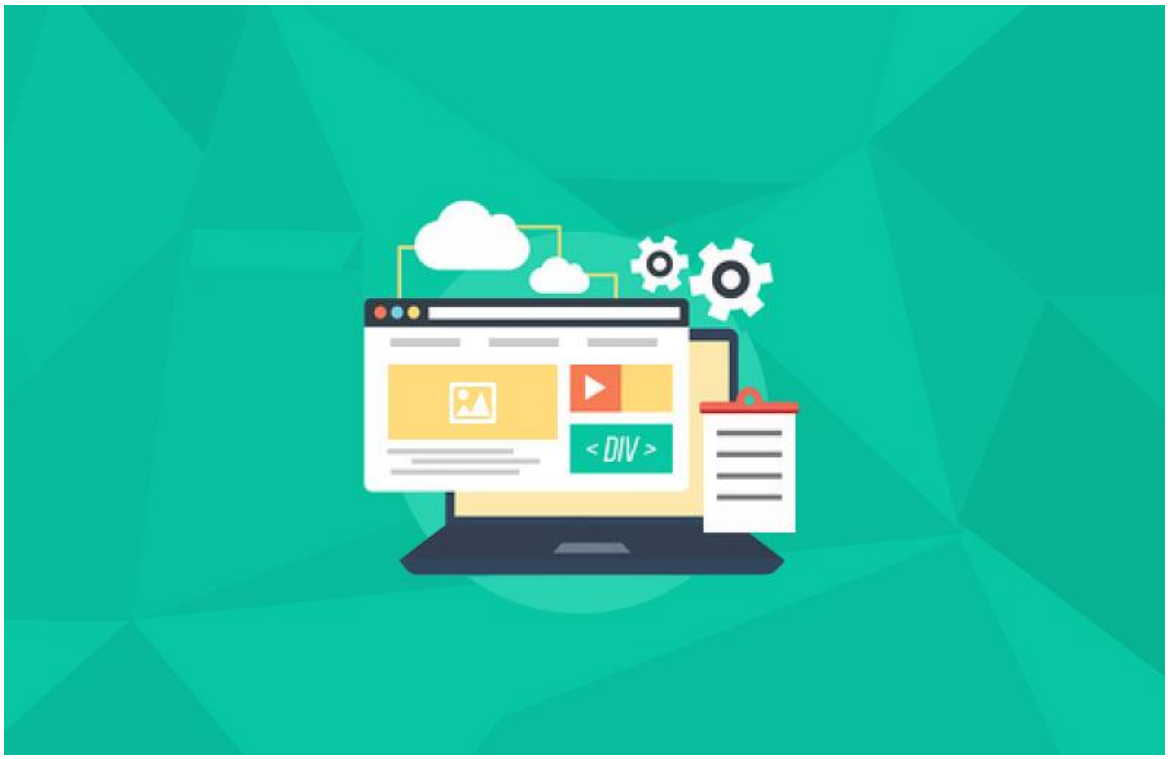


Tillämpning av kundinriktad identitetshantering och behörighetskontroll med elastisk skalning

Examensarbete inom Datateknik och Informationsteknik

Aleksander Pantić
Jonathan Örgård

Institutionen för Data och Informationsteknik
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET
Göteborg, Sverige 2020

Examensarbete 2020

Tillämpning av kundinriktad identitetshantering och behörighetskontroll med elastisk skalning

Aleksander Pantić

Jonathan Örgård

CHALMERS
UNIVERSITY OF TECHNOLOGY

Institutionen för Data och Informationsteknik
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET

Göteborg, Sverige 2020

Tillämpning av kundinriktad identitetshantering och behörighetskontroll med elastisk skalning
Aleksander Pantić
Jonathan Örgård

Handledare: Sakib Sistek, Chalmers Tekniska Högskola
Examinator: Jonas Duregård, Chalmers Tekniska Högskola

Institutionen för Data och Informationsteknik
Chalmers Tekniska Högskola / Göteborgs Universitet
SE-412 96 Göteborg
Telefon: 031-772 1000

Omslag:
Limelight Leads, 4-elements-of-a-good-website-design, oförändrad (CC BY 4.0).

Institutionen för Data- och Informationsteknik
Göteborg 2020

Abstract

Consumers' expectation of a sleepless interaction with services that offer a personified experience has seen a big growth in the last couple of years, this has led to a big need for reliable connections between digital services, devices and identities. The goal of this project was to apply a complete scalable solution for customer identity and access management, also known as CIAM. This report describes how an implementation of this kind of system was done. The resulting system is built with five different Docker containers. A user interface that communicates with a REST API. The REST API then communicates with an SQL database and a Keycloak implementation with an LDAP registry.

The result was a complete solution for customer identity and access management. With support for registering accounts and logging in with either an email address or social login. The project also included a translation module to make it easy to translate the language of the website. The system is scalable and implemented with Kubernetes.

Sammandrag

Konsumenters förväntan av en sömlös interaktion med tjänster som erbjuder en personifierad upplevelse har sett en stor tillväxt under senare år, som i sin tur skapat behov för pålitliga anslutningar mellan digitala tjänster, enheter och identiteter. Målet med detta projekt var att tillämpa en lösning för kundinriktad identitetshantering och behörighetskontroll med fokus på en tilltalande användarupplevelse med elastisk skalning. Denna rapport beskriver hur en implementering av sådant system kan utföras. Det konstruerade systemet är byggt på fem olika Docker-containrar, ett användargränssnitt som via ett REST-API kommunicerar med en SQL-databas samt en öppen identitetslösning med tillhörande LDA- katalog.

Resultatet blev en skalbar lösning för kundinriktad identitetshantering och behörighetskontroll med stöd för registrering och inloggning med sociala medie-konton samt en översättningsmodul som gör det enkelt att översätta texten i webbportalen baserat på valt språk. Systemet har stöd för containerbaserad skalning och har implementerats med Kubernetes.

Nyckelord: CIAM, OAUTH, OpenID Connect, Kubernetes, Docker, SSO

Förord

Vi vill ge ett stort tack till vår handledare Alexander Scott Crayvenn för stöd genom hela projektet samt hans radarpartner Lars-Olof Strid för teknisk rådgivning, med dessa grabbar vid sin sida kan inget gå fel. Vi vill även tacka vår handledare på Chalmers, Sakib Sistek för sin positivitet och för sitt förtroende.

Terminologi och förkortningar

AMER: Även förkortat AMERICAS används oftast av multinationella företag att hänvisa till en geografisk uppdelning som omfattas av Nord-, Central- och Sydamerika.

API: Application Programming Interface. Ett gränssnitt som mjukvaruutvecklare kan använda för att enkelt få åtkomst till data och / eller särskild funktionalitet för deras programvara. Vanligtvis exponerad som ett programvarubibliotek eller HTTP-tjänst.

Backend: En webbplats “baksida” är en kombination av teknik och programmering som driver en webbplats. Det handlar om funktionalitet som agerar bakom kulisserna. I detta fall består baksidan av tre delar användare aldrig ser: en server, en applikation och en databas.

CIAM: Customer Identity and Access Management. Kundinriktad identitetshantering och behörighetskontroll

CRUD: Create-Read-Update-Delete. Databas operationer.

CSS: Cascading Style Sheets. Ett stilarksspråk som beskriver hur dokument skrivna i märkspråk ska presenteras.

Docker: En öppen källkodsplattform för att bygga, använda och hantera behållarapplikationer.

EMEA: (Europe, Middle East, and Africa). Förkortningen är en vanlig metod för multinationella företag att hänvisa till en geografisk uppdelning som omfattas av Europa, Mellanöstern och Afrika.

Frontend: En webbplats “framsida” är det användaren ser i sin webbläsare, gränssnittet för webbplatsen som användaren interagerar med.

HTML: Hypertext Markup Language. Ett märkspråk för dokument som är designade att visas i en webbläsare.

HTTP: Hypertext Transfer Protocol. Protokoll för datakommunikation på internet.

IAM: Identity and Access Management går ut på att definiera och hantera roller och åtkomstbehörigheter för användare.

JSON: JavaScript Object Notation. Filformat för att spara data i läsbar text.

JWT/REF: Nyckeltyper

LDAP: Lightweight Directory Access Protocol är ett protokoll för att upprätthålla och komma åt servicekatalogsinformation.

NoSQL: Alla databaser som inte lagrar data i tabellform. Tex grafer och träd.

OAUTH: Open Authorization, är en öppen standard som används för att ge en säker delegerad åtkomst till en klientapplikation. Oauth använder sig av HTTPS för att kommunicera och använder nycklar för att auktorisera.

OIDC: Open ID Connect är ett autentiseringsprotokoll, baserat på OAuth 2.0-specifikationen. Den använder enkla JSON Web Tokens (JWT), som man kan få med flöden enligt OAuth 2.0-specifikationerna.

RDBMS: Relational Database Management System, databashanteringssystem baserad på den relationell modellen.

Recaptcha: Program för att verifiera att användaren är en människa och inte en dator. Om användaren misstänks vara datorstyrd konfronteras denna med en utmaning som är lätt att lösa för människor men kan vara svår för en dator. Tex bildigenkänning.

REST: Representational state transfer. Mjukvaroarkitektonisk stil för att kommunicera från maskin till maskin.

SMTP: Simple Mail Transfer Protocol. Kommunikationsprotokoll för att skicka mejl.

SOAP: Simple Object Access Protocol. Protokoll för att kommunicera från maskin till maskin.

SSO: Single Sign On är en egenskap för identitets- och åtkomsthantering som gör det möjligt för användare att autentiseras mot flera applikationer genom enbart en inloggning. Nämligen med bara en uppsättning referenser (användarnamn och lösenord).

SQL: Structured Query Language. Ett språk för att hantera, manipulera och hämta data från databashanteringssystem.

URI: Uniform Resource Identifier. En sträng som används för att namnge eller identifiera en resurs.

XHR: XMLHttpRequest. Webbläsar-JavaScript API för att skicka asynkrona HTTP förfrågningar.

XML: Extensible Markup Language. Ett märkspråk för dokument.

Innehåll

1 Introduktion

I detta kapitel presenteras bakgrunden till projektet samt syfte, mål, problemformulering och avgränsningar.

1.1 Bakgrund

Dagens uppkopplade konsumenter förväntar sig en sömlös interaktion med tjänster som erbjuder en personlig uppkopplad upplevelse, samtidigt som deras personliga information skyddas på ett säkert sätt. Ett viktigt steg i resan mot strategiska målsättningar att leverera en integrerad kundupplevelse börjar med rätt formulering och tillämpning av kundinriktad identitetshantering och behörighetskontroll (även känt som Customer Identity and Access Management).

Den digitala tjänsteekonomin har under senare år haft en stor tillväxt i både privat som offentlig sektor och på kort tid skapat ett stort behov för pålitliga anslutningar mellan digitala tjänster, enheter och identiteter. Även till synes traditionella industrier har kommit ombord på denna betydande förändringsresa. En stor andel av dessa adoptörer finner man inom fordonssektorn. Fler uppkopplade bilar lämnar produktionslinjen varje år [1]. Och i kombination med konsumenters önskan att få åtkomst till diverse personifierade tjänster i relation till en bil, banar nya vägar och möjligheter i ett redan kraftigt konkurrensutsatt marknadsområde.

1.2 Problemformulering

Eftersom dagens uppkopplade kunder förväntar sig lättillgängliga och personifierade tjänster via mobil-applikationer och webbportaler, blir deras identitet och behörighet en central fråga för att ge en enhetlig användarupplevelse. Detta lägger grunden till arbetets huvudfråga som undersöker hur man implementerar identitets- och behörighetsfederation som en elastiskt skalbar lösning för att hantera åtkomst och behörighet till tjänster och innehåll som kunder tar del av.

1.3 Syfte

Syftet med detta examensarbete är att tillämpa ett system för kundinriktad identitetshantering och behörighetskontroll med fokus på en tilltalande användarupplevelse med federerad identitet och dynamisk elastisk-skalning. Lösningen utgår från en identifierad växande behovsbild inom bilindustrin, men kan med lätthet anpassas till annan sektor. Den avsedda praktiska tillämpningen kommer ge en säker, sömlös kundupplevelse och via en federerad identitet kunna hantera digitala tjänster från fler aktörer som ingår i federationen. Hur kan man då på ett säkert sätt registrera och autentisera användare utifrån befintliga standarder?

1.4 Mål

Detta arbete har för avsikt att:

- Med hjälp av kända standarder och säkerhetsprinciper i åtanke utforma och implementera komponenter för autentiserings- och behörighetskontroll.

- Tillämpa tredjeparts autentisering anpassad för en kundsektor inom EMEA och AMERICAS.

- Utarbeta och realisera en dynamiskt skalbar lösning.

1.5 Omfång och avgränsningar

Projektet kommer innefattas av ett registreringssystem, ett inloggningssystem, en översättningsmodul, federeringsserver samt databaser. Funktionalitet och tjänster som vidare använder sig av ett användarkonto kommer inte täckas av detta examensarbete.

2 Teori

2.1 Identitetshantering och behörighetskontroll

Flertalet kända webbapplikationer använder öppna API:er vilket gör det möjligt för en tredjepart att få tillgång till deras tjänster. Öppna API:er är värdefulla då de tillåter tjänster att samarbeta i viktiga processer. Dessa öppna API:er används i många fall för personifierade tjänster. Därför är det viktigt att säkerställa användarens identitet och behörighet vilket kan vara komplext [2].

I början av internet eran var delning av information mellan tjänster enkelt uppbyggd. Användaren behövde endast ange sina inloggningsuppgifter för att nyttja tjänsterna. Detta tillät tredjeparts tjänster att hantera information om användaren utan några vidare begränsningar [3]. Det fanns inga garantier att organisationen bakom tjänsten sparade användarens lösenord på ett säkert sätt eller att tjänsten inte utnyttjade användarens personuppgifter till mer än vad som var nödvändigt.

Ett ytterligare problem för slutanvändaren var avsaknaden att på enkelt sätt ta tillbaka sitt medgivande till en tjänst som tidigare fått tillåtelse att använda ens uppgifter eller konto [3]. Användaren kunde dessutom inte kontrollera hur mycket information en tjänst fått tillgång till. Antingen delades allting eller inget.

En standard för auktorisering som tillåter användaren ge begränsad behörighet till ens information som även lätt kan återkallas blev snabbt efterfrågat. Facebook var ett av några få företag som implementerat egna auktoriseringssystem för att möta efterfrågan [2].

Man insåg snabbt komplexiteten och begränsningarna med växande popularitet och användning av integrerade tjänster som använde olika auktoriseringsprotokoll. OAuth skapades som det första steg mot ett standardprotokoll för auktorisering utan behovet av att dela ens lösenord [2].

2.1.1 OAuth

För enkelhets skull kan OAuth ses som ett nyckelkort till ett hotellrum. En gäst kan endast få tillgång till sitt eget rum med ett nyckelkort. Gästen får nyckelkortet genom en auktoriseringsprocess vid hotelldisken. Efter auktoriseringen får gästen nyckelkortet som kan användas för att ge tillträde till förbestämda utrymmen på hotellet [4].

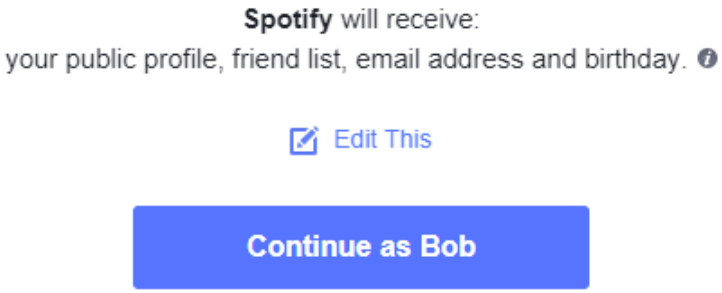
OAuth skapades för att ersätta det gammalmodiga inloggnings baserade auktoriseringssystemet, som nämndes tidigare i kapitel 2.1. OAuth är varken ett API eller en tjänst, det är en öppen standard för auktorisering. OAuth ger tillgång till data eller funktioner till en tredjepartstjänst på användarens vägnar utan att denne behöver ange sitt lösenord till tredjepartstjänsten.

OAuth 1.0 och OAuth 2.0 är två versioner av OAuth. Versionerna skiljer sig i hög grad och är ej kompatibla med varandra. Den mest använda standarden av de två och den som används

främst i detta projekt är OAuth 2.0. Det finns mycket som skiljer OAuth 1.0 mot OAuth 2.0, men två primära skillnader sticker ut. Första skillnaden är att OAuth 1.0 var designat för server-client webbapplikationer. Denna typ av webbapplikationer har blivit en mindre del av internet, nu när ensidiga applikationer blivit allt större på internet. Den andra skillnaden är att OAuth 1.0 kräver signerade nycklar och signaturer, medan i OAuth 2.0 används barriärnycklar som inte kräver en signatur [4].

OAuth har support för webbsidor, konsoler/TV, mobilapplikationer och server till serverapplikationer [5]. OAuth används för social inloggning men även datalagring, IoT enheter och mikrotjänster [4].

Ett exempel på användning av OAuth är när en användare med ett konto på en social media-plattform vill tillåta Spotify att få tillgång till användarens lista av vänner på den sociala plattformen. Användaren kommer då till en auktoriserings sidan (se figur 1). Där kan användaren se vilken tjänst som efterfrågar informationen samt vilka tillåtelser som tjänsten begär från användaren.



Figur 1. Bild på användarens auktoriseringssida med hjälp av OAuth

2.1.2 OAUTH terminologi

Innan vi ser närmare på vad som händer i ett typiskt OAuth flöde kan det vara bra att bekanta sig med terminologin bakom OAuth.

Resursägare	Ägaren av ens identitet, data och handlingar som behandlas [6].
Klient	Applikationen som resursägaren använder. Applikationen ber om data på begäran av resursägaren till resursservern [6].
Resursserver	API:et eller tjänsten som klienten vill använda på resursägarens villkor [6].
Auktoriseringsserver	Applikationen som känner till resursägaren och vart resursägaren redan har skapat konton. auktoriseringsservern får samtycke från en användare att tillåta en applikation att få tillgång till API:et. [7]

Scope	Det granulära tillstånd som applikationen vill ha, såsom tillgång till data eller att få tillstånd att utföra vissa handlingar. [7]
Verifikationskod	En kod som ges ut efter dess att klienten har gett tillstånd till omfattningen som applikationen vill ha. [6]
Klient ID/ klientnyckel	En identifierare som används för att identifiera klienten mot auktoriseringsservern. [7]
Klient hemlighet	Nyckel som används för att auktorisera klienten mot auktoriseringsservern. [8]
Auktoriseringskod	En kortlivad temporär kod som klienten ger till auktoriseringsservern i utbyte mot en tillgångsnyckel. [7]
Tillgångsnyckel	Nyckel som klienten kommer använda för att kunna kommunicera med resursservern. Detta är likt en nyckel som ger klienten tillåtelse att utföra uppgifter på resursservern eller förfråga data på användarens begäran. [7]
Uppdateringsnyckel	Används för att få en ny tillgångsnyckel om den gamla har utgått. [7]
Dirigerings URI	Ett URI som auktoriseringsservern kommer dirigera resursägaren till efter att resursägaren gett tillåtelse till klienten. [8]
Respons typ	Typen av respons som klienten förväntar sig få tillbaka. Den vanligaste respons typen är så kallat ’code’, då förväntar sig klienten att svaret innehåller en auktoriseringskod. [8]

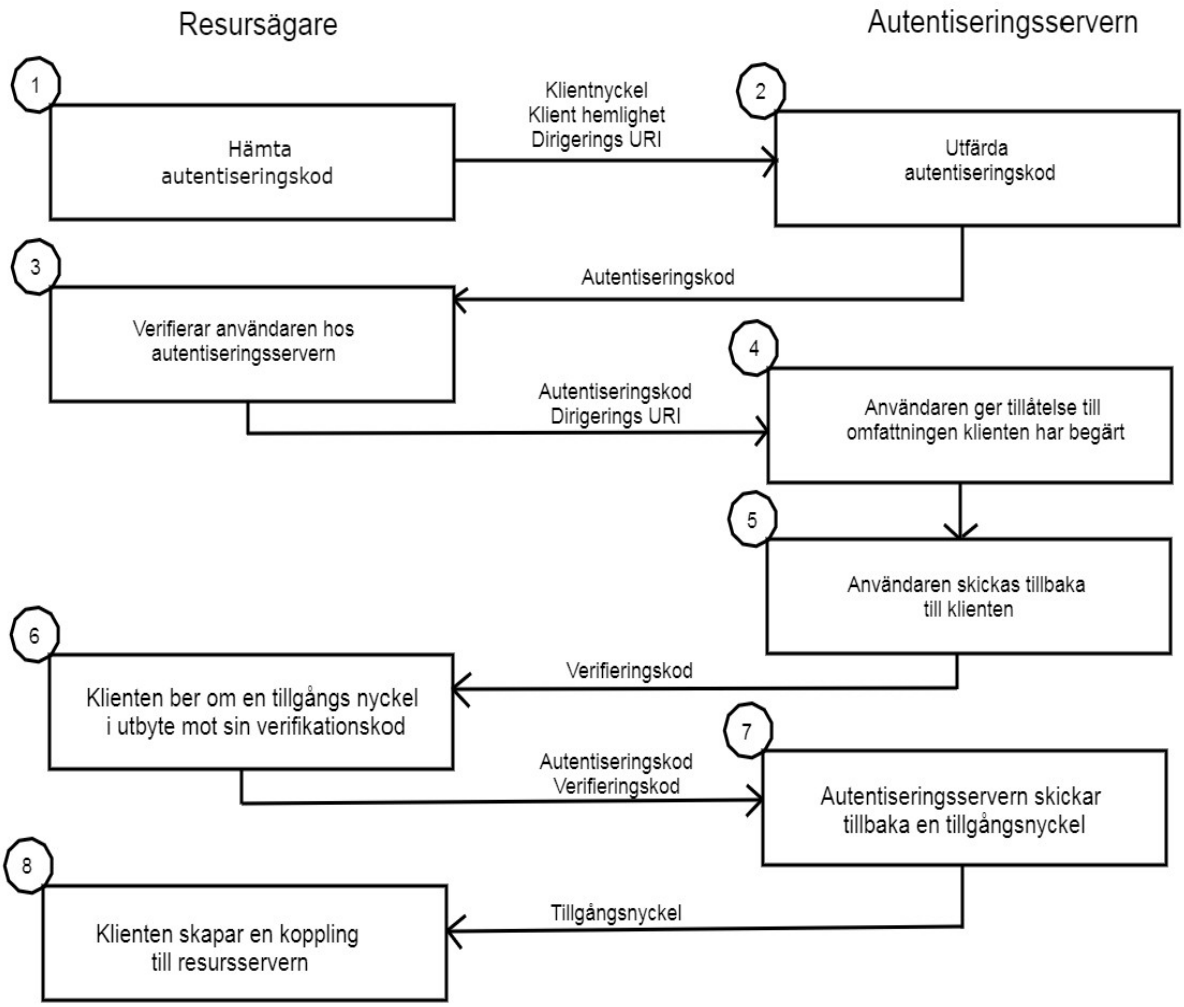
2.1.3 OAUTH flöden

2.1.3.1 Oauth 1.0 flöde

Oauth 1.0 flödet kan beskrivas med följande steg (se figur 2):

1. Användaren som då är resursägaren vill att en klient som kan vara en applikation får tillgång till information om resursägaren. Klienten ber om en auktoriseringskod som bara kan användas för att få en tillgångsnyckel. En dirigerings-URI, klientnyckel samt klienthemlighet skickas med i förfrågan.
2. Auktoriseringsservern skickar tillbaka en auktoriseringskod till klienten.
3. Auktoriseringsservern verifierar användaren och om nödvändigt ber användaren att logga in. Auktoriseringskoden och dirigerings-URI skickas även med i förfrågan.
4. Auktoriseringsserveren kommer då att begära tillstånd av användaren där det tydligt står vilken omfattning som klienten har begärt. Användaren får acceptera eller förneka villkoren.

5. Auktoriseringsserveren dirigerar webbläsaren tillbaka till klienten med hjälp av dirigerings-URI:en som skickades, auktoriseringsservern skickar också med en verifikationskod.
6. Klienten skickar en förfrågan om att få en tillgångsnyckel i utbyte mot sin verifikationskod.
7. Auktoriseringsservern verifierar koden och skickar tillbaka en tillgångsnyckel.
8. Klienten kan nu använda denna tillgångsnyckel för att skicka förfrågningar till resursservern för att få tillgång till användarens information.



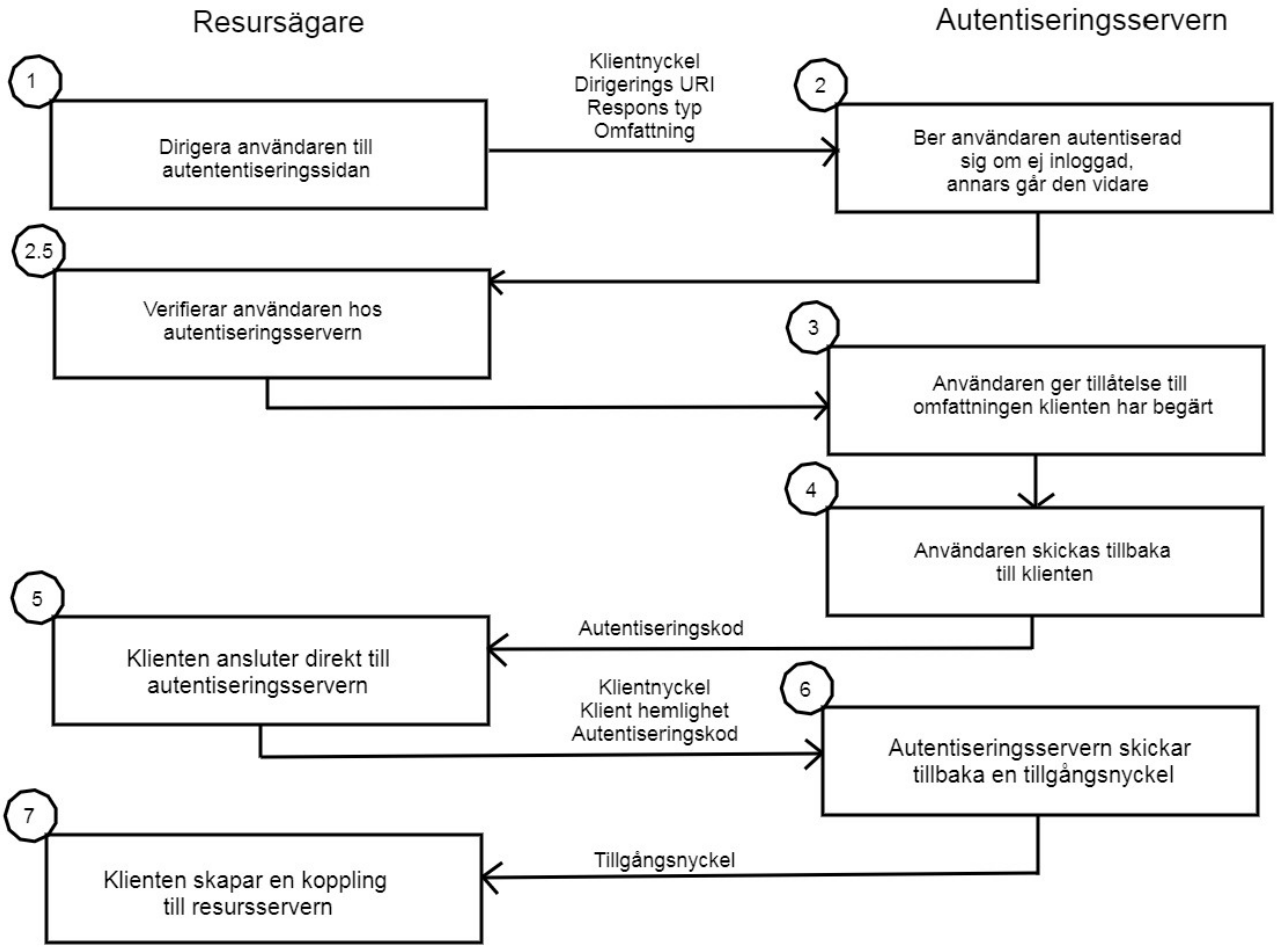
Figur 2. OAuth 1.0 flöde

2.1.3.2 OAuth 2.0 flöde

OAuth 2.0 flödet kan beskrivas med följande steg (se figur 3):

1. Användaren som då är resursägaren vill att en klient som då kan vara en applikation får tillgång till information om resursägaren. Klienten dirigerar då användarens webbläsare till auktoriseringsservern, i förfrågningen finns ett klientid, dirigerings-URI, responstypen och vilken omfattning som efterfrågas.
2. Auktoriseringsservern verifierar användaren och ber användaren att logga in om ej redan auktoriserad.

3. Auktoriseringsservern kommer då att begära en tillåtelse där det tydligt står vilken omfattning som klienten har begärt. Där får användaren acceptera eller förneka villkoren.
4. Auktoriseringsservern dirigerar webbläsaren tillbaka till klienten med hjälp av dirigerings-URI:n som skickades innan, en auktoriseringskod skickas med.
5. Klienten förbinder sig med auktoriseringsservern direkt dvs utan webbläsaren som resursägaren använder och skickar då på ett säkert sätt en klientnyckel**,** klienthemlighet samt auktoriseringskoden.
6. Auktoriseringsservern verifierar datan och skickar tillbaka en tillgångsnyckel.
7. Klienten kan nu använda tillgångsnyckel för att skicka förfrågningar till resursservern och därmed få tillgång till användarens information.



Figur 3. OAuth 2.0 flöde

2.1.4 OIDC

OpenID Connect (OIDC) är ett autentiseringsprotokoll. OIDC är ett tunt lager ovanpå OAuth 2.0 [9]. OAuth 2.0 har som syfte att ge tillgång till resurser och delning av resurser mellan tjänster, OIDC utökar detta med autentisering för användare. OIDC har som uppgift att ge användaren tillgång till flera webbsidor med endast ett par av inloggningsuppgifter. Vid

inloggning på en webbsida, kommer användaren till en OpenID inloggningssida där användaren loggar och sedan hänvisas tillbaka till webbsidan som inloggad.

OIDC flödet kan beskrivas i fyra olika steg:

1. En användare vill logga in på en tjänst med ett konto från en social plattform, tjänsten skickar då en auktoriseringsförfrågan till den sociala plattformen.
2. Den sociala plattformen autentiserar användarens inloggningsuppgifter eller ber användaren att logga in om ingen inloggningssession hittas. Sedan ber den sociala plattformen om användarens tillåtelse för de tillåtelser tjänsten vill erhålla, till exempel tillåtelse att se användarens mejladress.
3. När användaren har autentiserat sig och gått med på tjänstens förfrågan, skickar den sociala plattformen en tillgångsnyckel och eventuellt en ID-nyckel, tillbaka till tjänsten.
4. Tjänsten kan nu använda ID-nyckel eller tillgångs nyckel för att använda den sociala plattformens API.

2.2 SSO och federerad identitet

2.2.1 SSO (Single sign-on)

SSO kan enkelt ses som ett hus. Huset använder en ytterdörr som skyddar rummen istället för olika nycklar till alla rum i huset [10]. SSO är en del av IAM och tillåter användare att autentiseras på ett säkert sätt till olika applikationer/tjänster genom att endast logga in en gång med ett par inloggningsuppgifter. System utan SSO har kontouppgifter lagrade i sina egna databaser, detta gör att användaren måste logga in på varje applikation/tjänst som besökts med olika inloggningsuppgifter [11].
Ett förtroende mellan tjänster är grunden till att autentisering med SSO är möjligt [11]. Tjänsterna litar på varandra när användare loggar in att autentiseringen har skett på ett säkert. Därför behövs det inte att logga in igen när användaren fått tillgång till en av de andra tjänsterna [11]. SSO använder sig av tokens som skapar en identitet för en användare istället för inloggningsuppgifter, även om användaren fortfarande skriver in sina inloggningsuppgifter för att kunna logga in [12].

Följande steg används med SSO-autentisering:

1. Applikationen/tjänsten kontrollerar om användaren redan är autentiserad med SSO, om det stämmer får användaren tillgång till applikationen/tjänsten.

2. Om användaren inte har autentiserats tidigare hänvisas användaren till SSO-lösningen för att logga in. Där skriver användaren in sina inloggningsuppgifter för att autentiseras.

3. SSO-lösningen begär autentisering från identitets/autentiserings systemet som applikationen/tjänsten använder. Den verifierar autentiseringen och notifierar SSO-lösningen. SSO-lösningen skickar då autentiserings datan till applikationen/tjänsten och dirigerar användaren tillbaka till applikationen/tjänsten.

4. Om användaren försöker få tillgång till en annan applikation/tjänst efter autentisering med samma SSO-lösning så kontrolleras användaren. Eftersom användaren redan är autentiserad så kontrollerar den nya applikationen/tjänsten användarens identitet utan att användaren behöver skriva in sina inloggningsuppgifter igen.

2.2.2 Federation och federerad identitet

En federation innebär att minst två domäner med egna system som litar på varandra beslutar sig för att följa en viss standard för att kommunicera med varandra [13].

Federerad identitet kopplar en användares identitet mellan olika domäner i en federation [12]. Detta gör att användaren kan autentisera sig mot en domän och sedan komma åt resurser i den andra domänen. Ett exempel på detta är social inloggning då användaren vill logga in med sitt sociala plattformskonto från domän 1 för att komma åt resurser på domän 2.

2.2.3 Skillnad mellan federerad identitet och SSO

Federerad identitet och SSO är lätt att blanda ihop eller tro att dem är synonymer.
SSO tillåter användaren att med en inloggning få tillgång till flera olika system inom en organisation. Federerad identitet tillåter istället användare att få tillgång till olika system i olika organisationer. Federerad identitet skapar SSO för användare, men organisationer som använder SSO använder nödvändigtvis inte federerad identitet [14]. Federerad identitet är dock helt beroende av SSO-teknologi för att autentisera användare mellan olika domäner.

2.3 REST

REST-baserade tjänster används för att kommunicera från maskin till maskin via internet. Två vanliga API konsumtionstjänster är REST och SOAP.

REST är ett API-konsumtionsramverk som utifrån en given URI och HTTP-metod utför en funktion och sedan skickar ett svar som tex kan innehålla efterfrågad data eller en bekräftelse på att operationen som funktionen gjorde utfördes korrekt [15]. Exempel:

URI	HTTP Metod	Funktion	Svar
/användare	GET	Hämtar alla användare från databasen	Lista på alla användare
/användare	POST	Skapar en ny användare i databasen	Bekräftelse på om operationen utfördes korrekt
/användare/{AnvändarID}	GET	Hämtar den efterfrågade användare från databasen	Den efterfrågade användaren
/användare/{AnvändarID}	DELETE	Tar bort den efterfrågade	Bekräftelse på om operationen utfördes

		användaren från databasen	korrekt
/användare/{AnvändarID}	PATCH	Uppdaterar den specifika användaren i databasen	Bekräftelse på om operationen utfördes korrekt

SOAP är en API-konsumtionsprotokoll som utifrån ett XML-dokument utför en funktion och sedan skickar ett svar som tex kan innehålla efterfrågad data eller en bekräftelse på att operationen som funktionen gjorde utfördes korrekt [16]. Exempel:

SOAP förfrågnings dokument body del	**Funktion**	**Svar**
<soap:Body xmlns:m="http://www.exemple.se/användare<br">> <m:GetUsers> </m:GetUsers> </soap:Body>	Hämtar alla användare från databasen	Lista på alla användare
<soap:Body xmlns:m="http://www.exemple.se/användare<br">> <m:CreateUser> ...User info... </m:CreateUser> </soap:Body>	Skapar en ny användare i databasen	Bekräftelse på om operationen utfördes korrekt
<soap:Body xmlns:m="http://www.exemple.se/användare<br">> <m:GetUser> <m:UserID>{UserID}</m:UserID> </m:GetUser> </soap:Body>	Hämtar den efterfrågade användare från databasen	Den efterfrågade användaren
<soap:Body xmlns:m="http://www.exemple.se/användare<br">> <m:DeleteUser> <m:UserID>{UserID}</m:UserID> </m:DeleteUser> </soap:Body>	Tar bort den efterfrågade användaren från databasen	Bekräftelse på om operationen utfördes korrekt
<soap:Body xmlns:m="http://www.exemple.se/användare<br">> <m:UpdateUser> ...User info... </m:UpdateUser>	Uppdaterar den specifika användaren i databasen	Bekräftelse på om operationen utfördes korrekt

</soap:Body>		

SOAP är ett protokoll medan REST är en mjukvaroarkitektonisk stil. Detta innebär att även om båda ofta kan användas till samma saker så finns det för- och nackdelar med båda.

REST fördelar:

- REST är oftast lättare att förstå än SOAP på grund av det enklare URI och HTTP-metodgränssnitt som REST använder sig av jämfört med SOAPs XML gränssnitt.
- REST är snabbare än SOAP eftersom det simplare REST gränssnittet innebär att mindre data behöver överföras.

SOAP fördelar:

- Eftersom SOAP är ett protokoll finns det mer fördefinierade stöd för protokoll som kan användas, medan i REST är det upp till programmeraren själv att skapa protokollen. Ett exempel på detta är WS-ReliableMessaging protokollet [17] i SOAP som ser till att ett meddelande kommer fram.
- SOAP kan definiera exakt hur XML-förfrågningsdokument ska se ut och inte acceptera dokument som ser annorlunda ut. I REST är det upp till programmeraren att kontrollera så att all data som kommer in är korrekt formaterad.

2.4 Relationsdatabaser

För lagring och hantering av data har relationsdatabaser med SQL varit den främsta teknologin senaste årtiondena [18]. En databas är en uppsättning av data lagrat inuti en dator. Strukturen på databaser är att data på ett simpelt sätt blir tillgänglig att hämta, uppdatera och använda. En relationsdatabas är en databas som är uppdelat i olika tabeller (se figur 5). En tabell kan innehålla hundra, tusen eller miljontals rader av data. Varje rad innehåller kolumner av data. En kolumn kan tex vara av typen string, char eller integer [19].

Students Table

Student	ID*
John Smith	084
Jane Bloggs	100
John Smith	182
Mark Antony	219

Activities Table

ID*	Activity*	Cost
084	Swimming	$17
084	Tennis	$36
100	Squash	$40
100	Swimming	$17
182	Tennis	$36
219	Golf	$47
219	Swimming	$15
219	Squash	$40

Figur 5. Relation mellan två databastabeller

För att kommunikation med datatabeller i databasen används SQL. SQL gör det simpelt att med sökfrågor hämta eller lagra data på databasen.
Exempel på en sökfråga skulle kunna vara “SELECT ID FROM Students WHERE Student = Jane Bloggs”, detta skulle då returnera ID numret 100 från Students-tabellen (se figur 5). Med värdet från exemplet innan går det att i Activities-tabellen vaska fram att Jane Bloggs tränar squash och swimming (se figur 5).

2.5 Containerbaserad utveckling

Containerbaserad utveckling gör det lättare att skapa, använda och köra applikationer genom så kallade containrar [20]. En container är ett fristående och exekverbart paket av mjukvara som inkluderar allt som behövs för att köra en applikation: programkod, bibliotek, inställningar och systemverktyg. Detta göra att applikationen kan exekveras på andra system utan behovet av att sätta upp en miljö i förväg. En container är likt en virtuell maskin men till skillnad från en virtuell maskin så skapar en container inte ett helt eget operativsystem. En container använder sig av operativsystem-kärnan från systemet den kör på och gör att endast komponenter som inte redan finns på datorn skickas med i applikationscontainern [21]. Detta leder till en bättre prestanda och mindre storlek på applikationen. Containerbaserad utveckling gör att utvecklare kan fokusera på utveckling av applikationen utan att behöva tänka på systemet som exekverar applikationen.

2.6 Molnskalning

Molnskalning har som syfte att smidigt öka eller minska resurser till en IT lösning [22]. En av de största fördelarna med molntjänster är skalning, eftersom mängden resurser som allokeras kan variera beroende på behovet. Detta leder till att resurserna som används blir kostnadseffektivare. Horisontell och vertikal skalning är de två vanligaste typerna [23]. Horisontell skalning fungerar genom att mer maskiner läggs in i molnarkitekturen men fortfarande arbete som en enhet. Vertikal skalning innebär däremot att maskinerna som redan finns får mer kraft genom att bland annat öka CPU, RAM och lagrings resurserna.

Det finns tre huvudsakliga sätt att skala en molnarkitektur [24]:

Manuell skalning: Med manuell skalning krävs det att en individ manuellt ändrar tillgångarna till resurser i molntjänsten. Detta gör att minut snabba svängningar vid användning av applikationen inte går att kontrollera, vilket är kostnadsineffektiv. Risken med mänskliga misstag som att glömma minska resurser kan leda till onödiga kostnader och koldioxidutsläpp.

Tidsplanerad skalning: Med tidsplanerad skalning ökas resurserna den tiden på dygnet som användningen brukar vara större, samt minskas resurserna på tider som användning brukar vara lägre.

Automatisk skalning: Resurser skalas baserat på förutsatta regler. Till exempel så kan resurserna ökas när CPU eller RAM användning går över en viss procentuell gräns av resursens kapacitet. Autoskalning gör det möjligt att alltid ha tillräckligt med resurser tillgängligt samt att inte slösa resurser när användningen är lägre.

3 Genomförande

3.1 Verktyg

3.1.1 Docker

Docker är en öppen plattform för utveckling och exekvering av applikationer [25]. Docker gör det möjligt att köra mjukvara i containrar. Containrar är processer som är isolerade från resten av systemet, vilket ökar säkerheten eftersom mjukvaran inte kan komma åt resten av systemet. Docker använder sig av en distinkt systemavbildning vilket leder till att endast de beroenden som behövs används, vilket gör att systemresurser inte slösas bort i onödan. Till skillnad från VM (virtuella maskiner) som använder sig av virtuell hårdvara så delar containrar värd systemets hårdvara och operativsystemskärna vilket leder till mindre overhead.

3.1.2 Kubernetes

Kubernetes är ett orkestreringsverktyg som hanterar kluster av Docker-containrar [26]. Kubernetes kan automatiskt skala och hantera Docker containrar. Detta gör det möjligt att ha ett kluster av maskiner/noder med en huvudmaskin/nod som styr de andra maskinerna/noderna. Huvudmaskinen/noden kan dynamiskt starta eller stänga av Docker-containrar på de olika maskinerna/noderna beroende på behov [27].

Kubernetes använder sig av kapslar (eng. pod) som är de minsta enheterna som används vid skalning. En kapsel representerar en körbar enhet av arbete. En kapsel representerar en eller flera containrar och skall kontrolleras som en enhetlig applikation.

En replikerings kontroller används för att hantera flera kapslar. Med replikeringskontrollern kan Kubernetes skala kloner av en kapsel genom att öka och sänka antalet kopior av en kapsel.

En gruppering är en uppsättning av en eller flera kopior av kapslar. Grupperingar är gjorda för att hantera livscykel av kloner av en kapsel. Beroende på behov så ändras antalet kapslar i en gruppering för att fördela belastningen.

3.1.3 MSSQL

MSSQL (Microsoft SQL Server) är ett relationsdatabashanteringssystem [28]. Ett relationsdatabashanteringssystem används för att utföra operationer på en relationsdatabas. Tex för att skapa, läsa, uppdatera eller ta bort relationstabeller [29].

3.1.4 OpenLDAP

OpenLDAP (Open Lightweight Directory Access Protocol) är en öppen programvara för att implementera LDAP [30]. LDAP är ett protokoll för att upprätthålla och komma åt servicekatalogsinformation [31].

3.1.5 Keycloak

Keycloak är en IAM lösning som är en öppen programvara och används i moderna applikationer och tjänster [32]. Keycloak gör det möjligt att säkra applikationer och tjänster med relativt lite källkod.

En användare kan autentisera sig med Keycloak istället för flera olika individuella applikationer. När användaren har loggat in en gång behöver användaren inte logga in igen för att få tillgång till andra applikationer som använder sig av samma Keycloak-lösning. Samma principer då användaren loggar ut. Keycloak erbjuder även systemsäkerhet, social inloggning, support för mobilappar och integration för andra lösningar. Keycloak stödjer lagring av användardata i LDAP.

3.1.6 Node JS

Node.js är öppen programvara och plattformsoberoende som tillåter exekveringsmiljö i Javascript [33]. Node.js används vid utveckling av serversidor och nätverksapplikationer. Node.js körs på V8 JavaScript motorn som är kärnan för Google Chrome och är väldigt effektivt vid analysering och körning av javascript.

I Node.js medföljer NPM (Node Package Manager) som är ett pakethanteringssystem för JavaScript [34]. NPM tillåter delning av Javascript-paket så att andra användare kan använda dem i sina egna paket. Det finns tusentals av paket uppladdade från olika utvecklare världen över. NPM är ett grundläggande verktyg och är i många projekt oundvikligt.

3.1.7 React

React är ett JavaScript-biblioteket med öppen källkod. React utvecklades av Facebook och underhålls idag av Facebook, samt andra individuella utvecklare och företag [35]. React används för att bygga användargränssnitt.

React är komponentbaserat vilket leder till att komponenter programmeras och sedan sätts samman av React. Det finns många färdiga komponentbibliotek till React vilket gör det snabbt och enkelt att bygga ett användargränssnitt. Varje komponent tar hand om sitt eget tillstånd och när data inom en komponent uppdateras så laddar React om den komponenten vilket gör det simpelt att programmera ett responsivt användargränssnitt [35].

3.1.8 Git och GitHub

Git är ett versionshanteringsprogram med öppen källkod. Git används för versionshantering av kod vilket gör det lättare att samarbeta inom ett programmeringsprojekt [36]. Git tillåter ett icke linjärt arbetsflöde vilket innebär att man kan grena av och få flera parallella versioner av samma projekt samtidigt. Vilket gör det möjligt att jobba på flera olika funktionaliteter samtidigt. Från sin nya gren går det sedan att sammanfoga med huvudgrenen igen eller någon annan sidogren.

GitHub användes även för att spara projektet online för att snabbt och enkelt kunna dela med sig av uppdaterad källkod [37].

3.1.9 Visual Studio Code

För utvecklingen av projektet och källkoden användes Visual Studio Code IDE:n. Visual Studio Code användes på grund av användarvänligheten samt att flera bra verktyg kan användas med extensioner som finns tillgängliga att installera [38].

3.2 Implementering

3.2.1 Scrum

Scrum är ett agilt ramverk för att utveckla, leverera och underhålla komplexa produkter [39]. Scrum går ut på att dela upp arbetet i mål som kan bli avklarade inom en tidsbegränsad iteration [40]. Efter varje iteration är det tänkt att det man gjort under iterationen ska kunna vara en levererbar produkt. Dessa iterationer kallas sprintar och varar 1–4 veckor. Detta gör att man lättare kan anpassa och ändra målen med tiden eftersom man inför varje ny sprint kan sätta upp nya mål och på så sätt vara mer agil i sitt projekt. Efter varje sprint hålls en sprintåterblick där man demonstrerar vad man har gjort under sprinten och en sprint-retrospektiv för att reflektera över det som gick bra och det som gick mindre bra under sprinten, så att man kan göra ännu bättre nästa sprint.

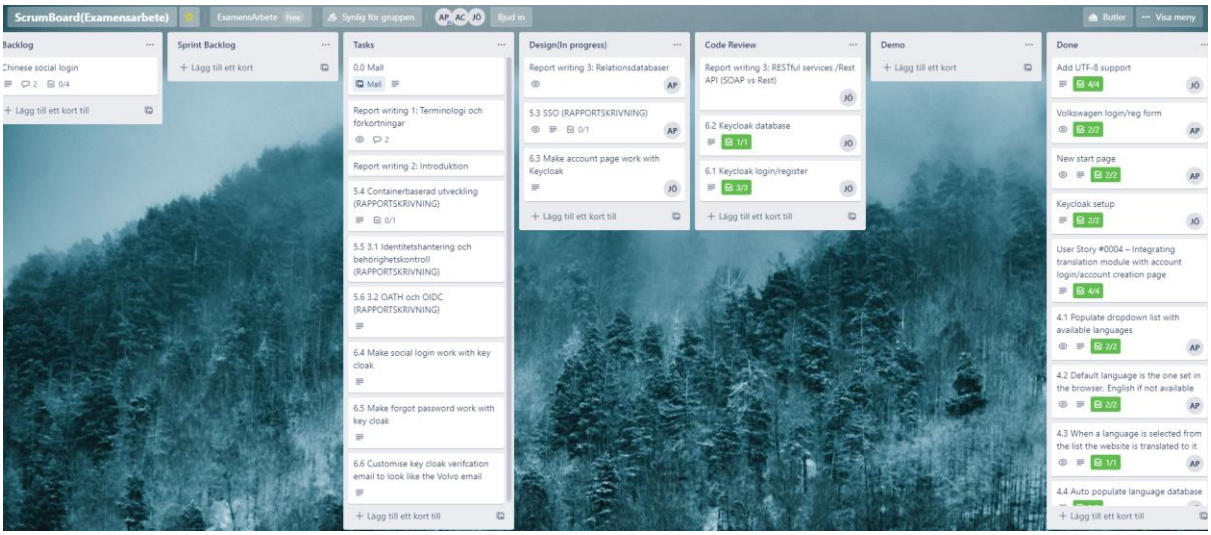
Utvecklingen av projektet utfördes med influenser av det agila ramverket Scrum. Dock applicerades inte alla teorier av Scrum då gruppens storlek var för liten. I och med applicering av Scrum används två veckors långa sprintar där teamet i början av varje sprint tillsammans med handledare bestämde vad som skulle utföras under de kommande två veckorna. Arbetsuppgifterna var utformade i form av användarscenarier och de användarscenarion som tillsammans beslutades att göras lades in i en sprint-backlog. Handledaren kunde ses som en produktägare ur ett Scrum perspektiv.

Varje dag hade utvecklarna korta möten där det diskuterades vad som ska göras och vad som har gjorts. Mot slutet av varje sprint anordnades en sprintåterblick med handledare och utvecklarna där de reflekterade över det som gjorts under veckan och vad som gick bra och vad som kunde ha utförts bättre.

Ett digital Scrum-bräde från webbsidan Trello (se figur 6) användes som hjälpmedel och var synlig för alla parter. Där listades användarscenarion, som sedan delades in i aktiviteter vilket skapade en transparens och tydlighet på vad som hade utförts och vad som var kvar att utföra. Utöver prioritetsordningen listades även alla krav för att möta “Definition of Done” för att ett användarscenario eller uppgift skulle räknas som klar.

I början av varje sprint bröt teamet ner användarscenarion i uppgifter för att lättare förstå vad som skulle krävas för att uppnå kraven för användarscenariot samt lättare kunna fördela arbetsuppgifterna. När en teammedlem börjat arbeta på en arbetsuppgift satte denna sin stämpel på uppgiften för att indikera att den tog ansvar för uppgiften samt flyttade den till “In Progress” kolumnen på Scum-brädet för att visa att uppgiften hade påbörjats. När kraven hade uppfyllts och arbetstagaren kände sig klar med arbetsuppgiften så flyttades uppgiften till “Code Review” kolumnen där den andra teammedlemmen kontrollerade att alla krav uppfyllts, att allt fungerade felfritt samt att uppgiften var kodmässigt bra gjord. Efter att den andre teammedlemmen kontrollerat allt detta flyttades

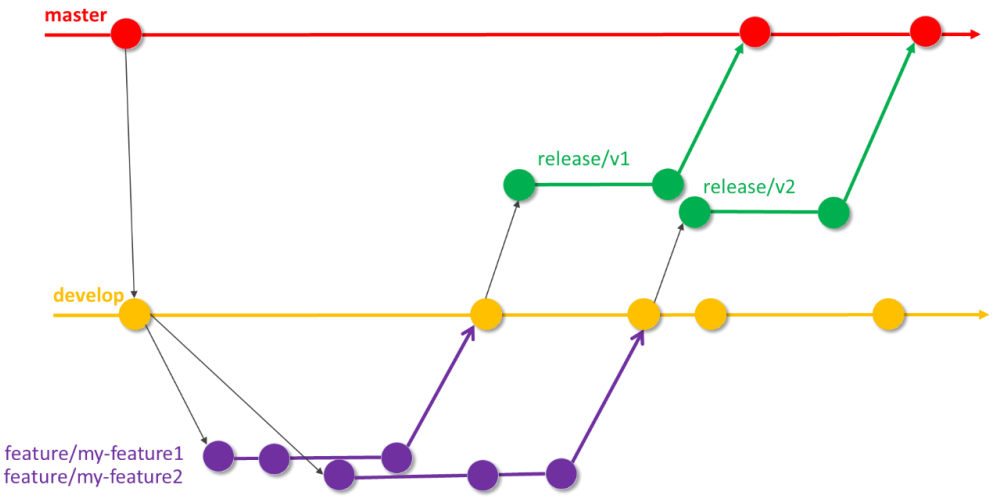
uppgiften till “Demo” kolumnen för att visa att den var redo att visas för handledaren nästa sprintåterblick. Efter godkännande från handledare flyttades sedan uppgiften till “Done” kolumnen på Scrum-brädet.



Figur 6. Bild från Scrum-Brädet.

3.2.2 Gitflow

Projektets källkod delades med Github och strategin för delningen var Gitflow (se figur 7). Gitflow användes under projektet för att öka verkningsgraden av Git. Gitflow går ut på att en mästergrenen med projektets kod uppdateras i slutet av varje sprint samt en utvecklingsgren som uppdateras kontinuerligt under sprinten [41]. Implementation av koden utfördes med funktionsgrenar från utvecklingsgrenen och när implementationen var klar och testad utfördes en så kallad “pull” förfrågan till utvecklingsgren. Efter inspektion av koden och godkännande av den andra personen i gruppen implementerades koden till utvecklingsgrenen. Utvecklingsgrenen implementeras i slutet av sprinten till mästergrenen efter godkännande från handledaren. Gitflow gjorde det flexibelt att ta bort och utföra ändringar på källkoden.



Figur 7. Processen (GitFlow) kring hur gruppmedlemmarna har delat upp koderna genom projektets gång.

4 Användarscenarion

Under projektet utgick vi ifrån användarscenarier som delades upp under projektets sprintar och dessa sprintar varade i 2 veckor vardera. Användarscenarier ses från användarens perspektiv och listar vad en användare skall kunna åstadkomma för att ett användarscenario skall kunna anses som klar i form av acceptanskrav.

När ett användarscenario anses vara färdigutvecklad flaggas den för kodgranskning. Då går en teammedlem igenom alla acceptanskrav och kontrollera att programmet uppfyller alla dessa krav, så kallad acceptanstestning. Teammedlemmen går även igenom den förändrade koden för att se att den är välskriven. Om alla acceptanskrav är uppfyllda och koden är välskriven anses användarscenariot vara klart, om ej så lämnar teammedlemmen feedback på användarscenariot och den som arbetat på den får förbättra koden.

4.1 Användarscenario 1 – Kontoregistrering (EU/NA)

Som konsument vill jag kunna registrera ett personligt konto (med min mejladress som användarnamn eller via Google, Facebook, Twitter, LinkedIn),
Så att jag kan ansluta och hantera personliga tjänster.

Acceptanskrav:

- Registreringssidan skall ha följande obligatoriska fält:
    - Förnamn
    - Efternamn
    - Mejladress (vid registrering med mejl, inte för socialt konto)
    - Lösenord (vid registrering med mejl, inte för socialt konto)
- Användaren kan bara lämna in en förfrågan genom att fylla i alla de nödvändiga fälten (vid registrering med mejl, inte för socialt kontot)
- Mejladressen som har getts måste nå kriterierna för mejl strukturen (vid registrering med mejl, inte för socialt kontot)
- Användaren kan skapa ett konto med följande sociala konton:
    - Google
    - Facebook
    - Twitter
    - LinkedIn
- Användaren måste bekräfta att de har läst och accepterat de allmänna villkoren för att kunna skicka en begäran för registrering
- En listruta för språk styr språkinställningarna (lista på språk ges separat)
    - Registreringsformulärets språk är initialt inställt för att återspegla webbläsarens språk
    - Andra språk kan väljas med hjälp av listrutan
    - Engelska används som språk. Om det valda språket inte finns
- Användaren kommer att få ett bekräftelsemeddelande efter framgångsrik registrering för att verifiera och aktivera konto
- Information från formuläret lagras i en registreringsdatabas
- Lösenords kriterium:
    - Minst 8 karaktärer
    - Minst 1 versal
    - Minst 1 gemen
    - Minst 1 siffra
    - Minst 1 specialtecken
- En Google Recaptcha skall se till att använder inte är en robot

4.2 Användarscenario 2 – Kontoinloggning (EU/NA)

Som konsument vill jag kunna logga in på mitt personliga konto (med min mejladress som användarnamn eller via Google, Facebook, Twitter, LinkedIn),
Så att jag kan ansluta till och hantera mina personliga tjänster.

Acceptanskrav:

- Användaren kan logga in med mejl om mejladressen har blivit verifierad, eller med följande sociala konton:
    - Google
    - Facebook
    - Twitter
    - LinkedIn
- En listruta för språk styr språkinställningarna (funktionalitet kommer i nästa sprint)
    - Registreringsformulärets språk är initialt inställt för att återspegla webbläsarens språk
    - Andra språk kan väljas med hjälp av listrutan
    - Engelska används som språk. Om det valda språket inte finns
- Framgångsrik inloggning leder användaren till sitt konto
- En Google Recaptcha skall se till att använder inte är en robot

4.3 Användarscenario 3 – Användarkonto aktiveringsmejl och aktivering

Som konsument som lyckats registrera ett nytt konto baserad på min mejladress,
vill jag få ett mejl för att bekräfta min registrering och aktivera mitt konto,
så att jag kan verifiera min identitet innan kontot aktiveras.

Acceptanskriterier:

Mejl mottagaren ska kunna interagera med vårt varumärke genom att:

- Klick på varumärkets logga öppnar varumärkets hemsida
- Klick på Facebook ikonen öppnar varumärkets Facebook-sida
- Klick på Twitter ikonen öppnar varumärkets Twitter-sida
- Klick på Instagram ikonen öppnar varumärkets Instagram-sida
- Klick på YouTube ikonen öppnar varumärkets YouTube-sida

- Aktiveringslänken är giltig i 14 dagar från och med registrering.
- Eftersom svar på mejl som är utskickade från företaget inte tas emot, så kan mottagaren komma i kontakt med företaget genom att klicka på en aktiv länk, som tar mottagaren till företagets “Kontakta oss”-sida.
- Mottagaren av mejlet har lätt tillgänglig till företagets kund integritetspolicy genom att klicka på en aktiv länk.
- Mottagaren har möjligheten att ta reda på mer om företaget och kontot genom att klicka på en aktiv länk som öppnar företagets support sida.
- Klick på konto aktiveringslänken efter 14 dagar kommer leda till en sida som säger: “Din aktiveringslänk har utgått. Registrera ditt konto igen.”.
- Vid klick på aktiveringslänken, (innan den har utgått), så blir användaren skickad till en sida som informerar användaren att aktiveringen av kontot var lyckat.

4.4 Användarscenario 4 – Användarkonto aktiveringsmejl påminnelse

Som konsument som lyckats skicka in en ny kontoregistrering baserad på min mejladress,

Vill jag få en mejlpåminnelse 7 dagar efter att kontot har registrerats (om ej redan verifierad),
Så att jag blir påmind att göra klart min registrering genom att aktivera mitt användarkonto.

Acceptanskriterier:
Mejl mottagaren ska kunna interagera med vårt varumärke genom att:

- Klick på varumärkets logga öppnar varumärkets hemsida.
- Klick på Facebook ikonen öppnar varumärkets Facebook-sida.
- Klick på Twitter ikonen öppnar varumärkets Twitter-sida.
- Klick på Instagram ikonen öppnar varumärkets Instagram-sida.
- Klick på YouTube ikonen öppnar varumärkets YouTube-sida.

- Aktiveringslänken är giltig 7 dagar från och med att mejlet har sänts ut.
- Eftersom svar på mejl som är utskickade från företaget inte tas emot, så kan mottagaren komma i kontakt med företaget genom att klicka på en aktiv länk, som tar mottagaren till företagets “Kontakta oss”-sida.
- Mottagaren av mejl har lätt tillgänglig tillgång till företagets kund integritetspolicy genom att klicka på en aktiv länk.
- Klick på konto aktiveringslänken efter 7 dagar kommer leda till en sida som säger: “Din aktiveringslänk har utgått. Registrera ditt konto igen.”.
- Vid klick på aktiveringslänken, (innan den har utgått), så blir användaren skickad till en sida som informerar användaren att aktiveringen av kontot var lyckat.

4.5 Användarscenario 5 – Översättningsmodul lagra ord

Som en utvecklare som vill översätta text,
Vill jag kunna välja en bastext från en listruta och lagra översättningar av denna bastext på olika språk till en databas,
Så att det underlättar för mig som utvecklare att skapa översättningar till webbapplikationen.

Motivering:
Skapa en sida så att utvecklaren enkelt kan lagra översättningar.

Acceptanskrav:

- Skall kunna lägga in ett nytt språk i databasen
- Skall kunna lägga in bastexter som är på engelska
- Skall kunna ändra och ta bort bastexterna
- Skall kunna skapa översättningar bastexterna till andra språk
- Skall kunna ändra och ta bort bastexterna

4.6 Användarscenario 6 – Integrera översättningsmodulen med login/registreringssidan

Som en konsument som vill skapa eller logga in på ett konto,
Vill jag kunna välja ett språk från en listruta som översätter språket på sidan.
Så att jag som konsument känner mig tryggare att använda mitt konto med ett språk av mitt val.

Motivering:
Skapa en konsumentupplevelse som adapteras till konsumentens valda språk.

Acceptanskrav:

- Om du väljer listrutan listas alla tillgängliga språk / översättning
- Varje språkalternativ skrivs på det lokala språket

- Språk är initialt inställt för att återspegla webbläsarens språk
- När du väljer ett annat språk i listrutan hämtas översättningstexten från översättningsmodulen och översätter sidan

4.7 Användarscenario 7 – Skalning av systemet

Som utvecklare av applikationen,
Vill jag kunna skala systemet genom att köra systemet på Kubernetes,
Så att system för de resurser den behöver och inte mer.

Motivering:
Integrera systemet med Kubernetes

Acceptanskrav:
- Skall kunna köra systemet på Kubernetes
- Skall kunna skala med Kubernetes autoskalningsfunktion
- Ett enkelt test för att se om skalningen funkar

4.8 Användarscenario 8 – Återställning av lösenord

Som en konsument som har glömt mitt lösenord,
Vill jag kunna återställa mitt lösenord med hjälp av min mejladress,
Så att jag kan komma åt mitt konto igen.

Motivering:
Skapa ett sätt för användare att återställa sitt lösenord via sin mejladress.

Acceptanskrav:
- Användaren ska kunna klicka på en “glömt lösenord”-knapp i inloggningsformuläret som tar den till ett glömt lösenordsformulär.
- Glömt lösenord formuläret skall ha följande obligatoriska fält:
    - Mejladress
- Användaren kan bara lämna in en förfrågan genom att fylla i alla nödvändiga fält
- Mejladressen som har getts måste nå kriterierna för mejlstrukturen
- Användaren kommer att få ett bekräftelsemeddelande efter framgångsrik registrering för att verifiera och aktivera konto
- En Google Recaptcha skall se till att använder inte är en robot

4.9 Användarscenario 9 – Konto borttagning

Som konsument som inte vill använda applikationen längre,
Vill jag kunna ta bort mitt konto,
Så att jag inte finns kvar i systemet.

Acceptanskrav:
- Användaren kan enkelt ta bort sitt konto om registrerad med mejl
- Användaren kan enkelt ta bort sitt konto om registrerad med ett av följande sociala konton
    - Google
    - Facebook
    - Twitter
    - LinkedIn

4.10 Användarscenario 10 – Keycloak implementation

Som en utvecklare,
Vill jag kunna hantera konton med Keycloak,
Så att jag har en säker autentiseringsmetod.

Acceptanskrav:

- Keycloak ska använda en LDAP katalog för lagring
- Implementera registrering/logga in med följande sociala konton
    - Google
    - Facebook
    - Twitter
    - LinkedIn
- Registrera/logga in med mejl
- Exekvera i en egen Docker-container
- Kunna byta lösenord

4.11 Användarscenario 11 – Modulärt med Docker

Som utvecklare,
Vill jag ha ett system som är modulärt med Docker,
Så att systemet kan exekvera lätt på olika maskiner.

Acceptanskrav:

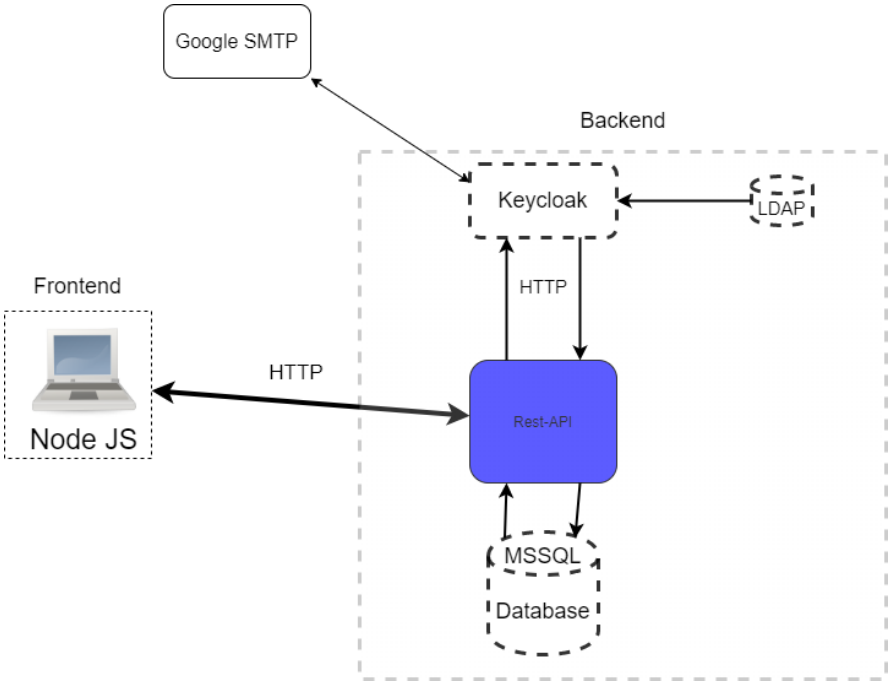
- Följande delar skall exekveras i Docker-containrar
    - Användargränssnittet
    - Rest API
    - MSSQL databas
    - LDAP
    - Keycloak
- Containrarna skall kunna kommunicera med varandra via HTTP

5 Arkitekturbeskrivning

5.1 Överblick

Systemet kan delas upp i två delar, frontend och backend. Frontend är webbsidan med ett användargränssnitt medan backend innehåller delarna som användaren inte kan se. Frontend exekverar i en Docker-container i Node JS medan backend har fyra olika Docker-containrar. De fyra Docker-containrarna är ett REST-API, Keycloak, OpenLDAP samt en MSSQL-databas.

Frontend som användaren ser kommunicerar med Keycloak och MSSQL-databasen genom ett REST-API. Kommunikationen sker med HTTP-förfrågningar från webbsidan till ändpunkter på REST-API:et. MSSQL-databasen innehåller data för översättningsmodulen. Keycloak används som federationserver och hanterar konto registrering och inloggning. Keycloak använder sig utav en LDAP-katalog för att spara användardata. Se figur 8.



Figur 8. Överblick på systemet

5.2 Användargränssnittet (webbsidan)

Användargränssnittet exekveras i en Docker-container som är byggt på Node JS. React används för kodningen av användargränssnittet tillsammans med flertalet JavaScript-bibliotek. Användargränssnittet skickar HTTP-förfrågningar till REST-API:et för att olika operationer skall utföras på backend sidan, beroende på vad användaren väljer att göra.

5.3 Rest API

Rest API:et exekveras i en Docker-container och är byggd med C# ramverket .NET Core. Rest-API:et lyssnar efter förfrågningar från webbapplikationen för att kommunicera med

databasen och Keycloak. Rest API:et utför sedan databasoperationer på databasen eller ber Keycloak utföra operationer beroende på vad webbsidan har efterfrågat. Efter att operationen utförts skickas ett svar tillbaka till webbsidan som innehåller data eller status på hur operationen gick beroende på vilken typ av operation som har utförts.

5.4 MSSQL

Microsoft SQL 2019 databasen exekveras i en Docker-container. MSSQL-databasen innehåller tre olika tabeller som används till översättningsmodulen.

dbo.Language: Innehåller information om varje språk. I tabellen lagras förkortningar av språket, språkets namn på engelska, språkets inhemska namn och i vilket land språket används (se figur 9).

ID	NameShort	NameShortEnglish	NameLocal	NameEnglish	CountryShort	Country	Createdate	Modifydate
1	BG	BUL	Balgariya	Bulgarian	BGR	Bulgaria	2020-05-13 23:09:22.9766667	2020-05-13 23:09:22.9766667
2	BE	BEL	Belaruskaja mova	Belarusian	BLR	Belarus	2020-05-13 23:09:22.9766667	2020-05-13 23:09:22.9766667
3	BS	BOS	Bosanski	Bosnian	BIH	Bosnia and Herzegovina	2020-05-13 23:09:22.9766667	2020-05-13 23:09:22.9766667
4	CS	CZE	Čeština	Czech	CZE	Czechia	2020-05-13 23:09:22.9766667	2020-05-13 23:09:22.9766667
5	DA	DAN	Dansk	Danish	DNK	Denmark	2020-05-13 23:09:22.9766667	2020-05-13 23:09:22.9766667

Figur 9. Exempel på språk.

dbo.Text: I denna tabell lagras all bastext som varje översättning utgår ifrån (se figur 10).

ID	Content	Comments	CreateDate	ModifyDate
1	Volvo ID	NULL	2020-05-13 23:09:22.9800000	2020-05-13 23:09:22.9800000
2	Log in	NULL	2020-05-13 23:09:22.9800000	2020-05-13 23:09:22.9800000
3	Register	NULL	2020-05-13 23:09:22.9800000	2020-05-13 23:09:22.9800000
4	Your email address	NULL	2020-05-13 23:09:22.9800000	2020-05-13 23:09:22.9800000
5	Password	NULL	2020-05-13 23:09:22.9800000	2020-05-13 23:09:22.9800000

Figur 10. Exempel på bastexter.

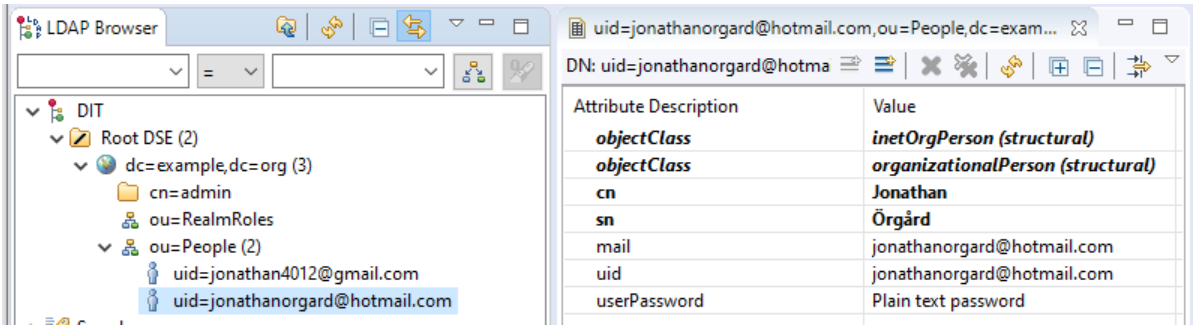
dbo.Language_Text: Tabellen lagrar översättningarna till bastexterna. I tabellen kopplas varje text till ett språk med LanguageID som finns i dbo.Language tabellen samt en bastext med TextID som finns i dbo.Text tabellen (se figur 11).

LanguageID	TextID	LanguageContent	Createdate	Modifydate
34	14	Välj språk	2020-05-13 23:09:22.9833333	2020-05-13 23:09:22.9833333
34	15	Volkswagen ID	2020-05-13 23:09:22.9833333	2020-05-13 23:09:22.9833333
37	1	Volvo ID	2020-05-13 23:09:22.9833333	2020-05-13 23:09:22.9833333
37	2	登录	2020-05-13 23:09:22.9833333	2020-05-13 23:09:22.9833333
37	3	寄存器	2020-05-13 23:09:22.9833333	2020-05-13 23:09:22.9833333

Figur 11. Exempel på översättningar.

5.5 OpenLDAP implementation

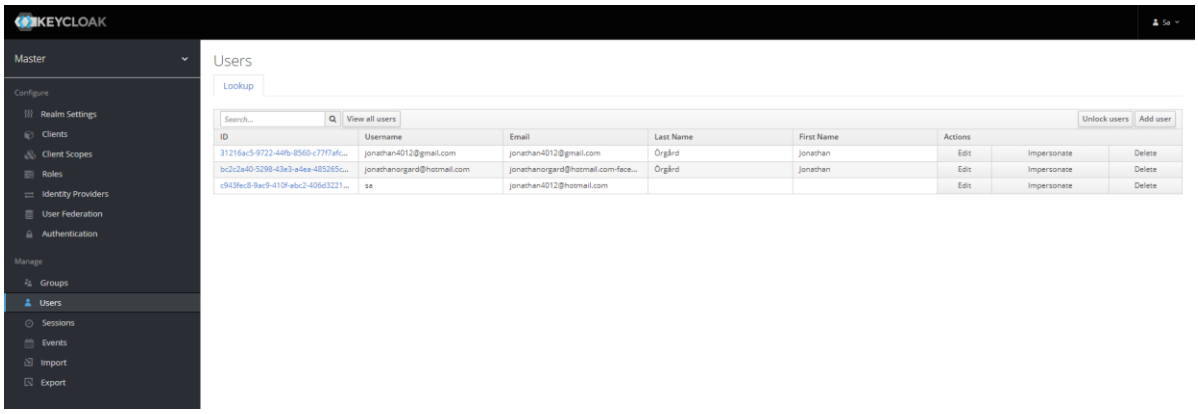
OpenLDAP katalogservern exekverar i en Docker-container och innehåller en katalog med användarinformation. Användarinformationen innehåller användarens uid (användarnamn), mejladress samt hashade lösenord (se figur 12).



Figur 12. Exempel på användarinformation.

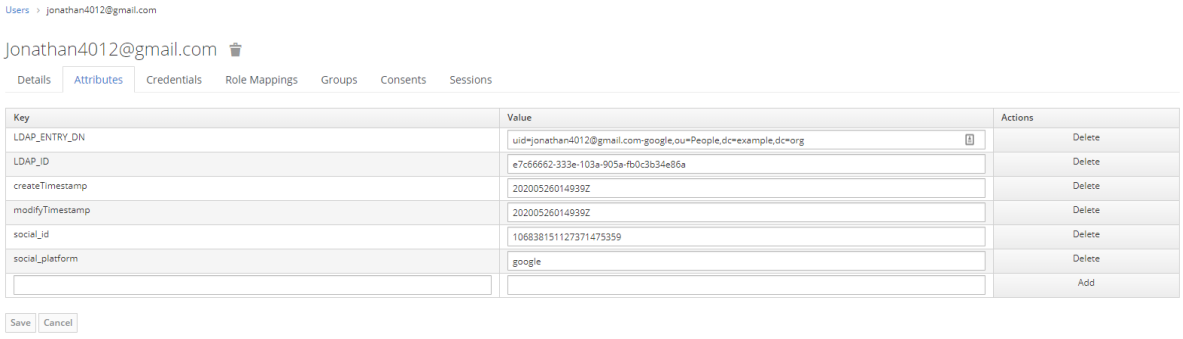
5.6 Keycloak implementation

Keycloak federeringservern exekverar i en Docker-container och har ett inbyggt API som lyssnar efter förfrågningar från REST-API:et för att utföra operationer samt en administratörskonsol som låter en administratör redigera användares uppgifter och ändra Keycloak inställningar (se figur 13).



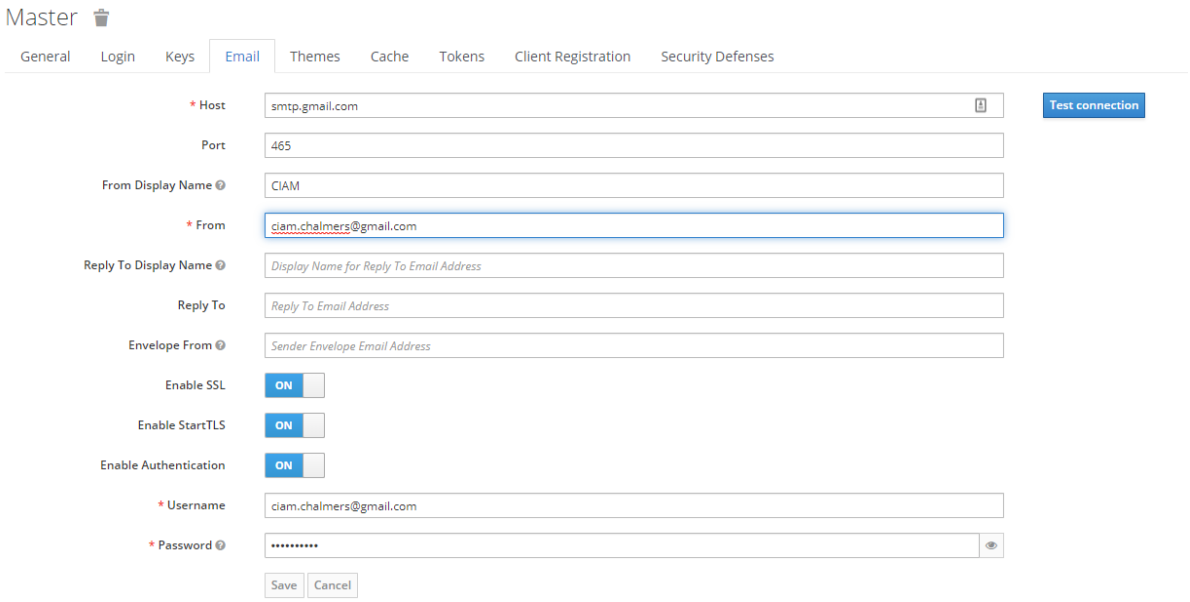
Figur 13. Lista på alla användare i administratörskonsolen.

Användardata för autentisering som nämns i kapitel 4.5 sparas och läses från en LDAP-katalog. Resten av användardatan sparas i KeyCloaks egna SQL-databas. Även denna data går att uppdatera via API eller administratörskonsolen (se figur 14).



Figur 14. Användar attribut i administratörskonsol.

För att verifiera en användares mejladress samt för att återställa en användares lösenord skickar Keycloak ett mejl till användaren. Detta sker med hjälp av en Google SMTP-server (se figur 15).

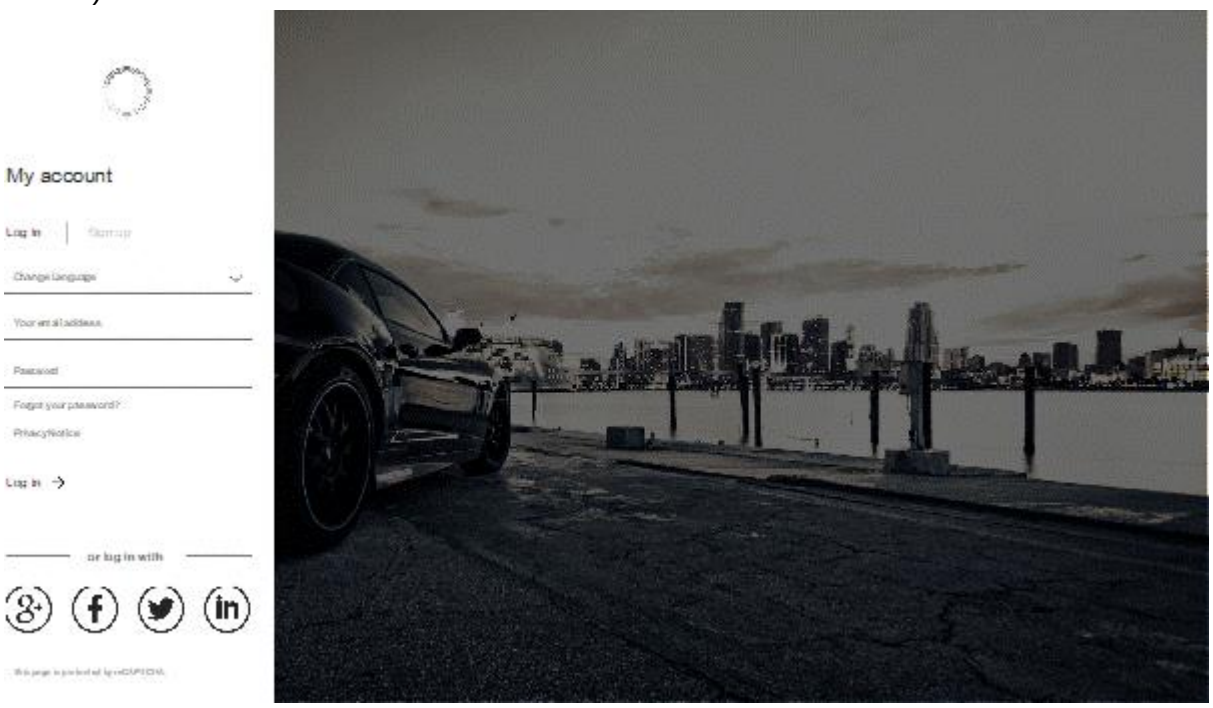


Figur 15. Google SMTP koppling.

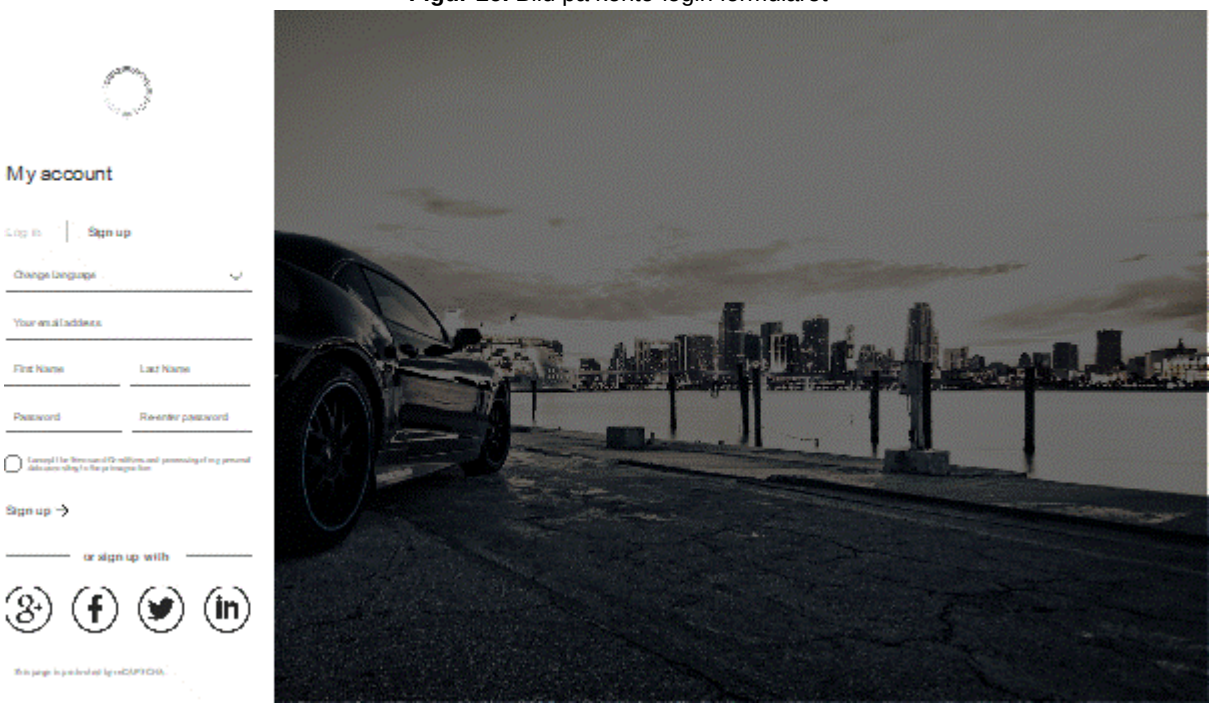
6 Design

6.1 Användargränssnittet

Gränssnittet togs fram via en iterativ attrappprocess med fokus på minimalism, enkelhet och en ändamålsenlig design som påminner om en bekant mobilanvändarupplevelse (se figur 16 och 17).



Figur 16. Bild på konto-login formuläret



Figur 17. Bild på registreringssidan

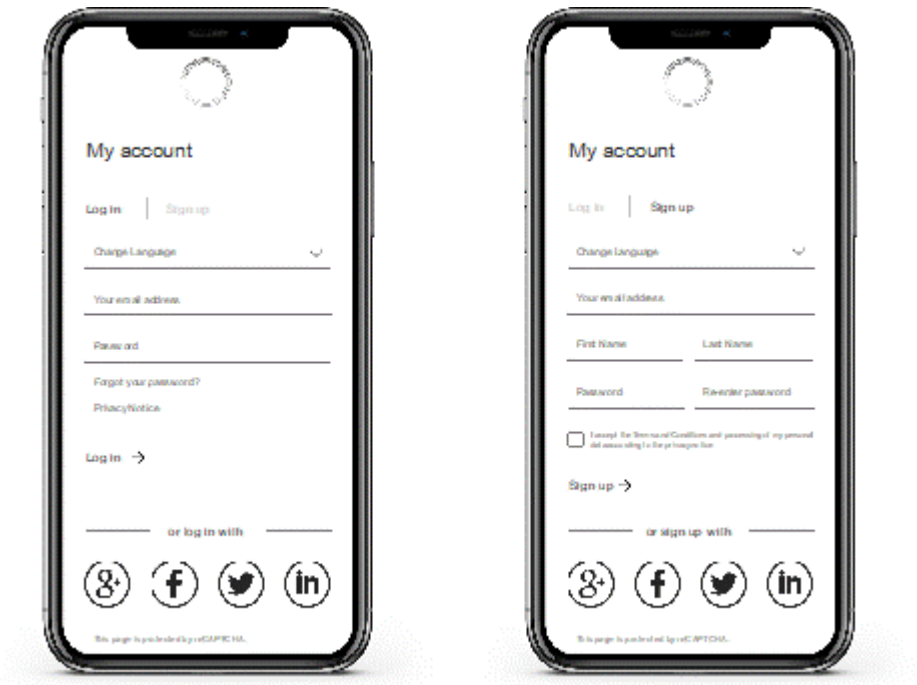
Varje enskild sida är skapad med HTML och CSS. Användargränssnittet är responsivt anpassad för att kunna användas på datorer och mobiler. Responsiviteten skapades genom att den procentuella storleken av HTML-elementen kunde bestämmas beroende på webbläsarens skärmstorlek. Även mediaförfrågningar (eng. media queries) användes vilket tillät att olika CSS-stilregler kunde sättas beroende på webbläsarens skärmbredd och skärmlängd. Detta gjorde det möjligt att anpassa vad som skulle visas och hur det såg ut beroende på ett visst intervall av skärmstorlekar.

I användargränssnittet kan användaren ändra språket som visas med hjälp av översättningsmodulen. Detta gör att texten på användargränssnittet byts till det valda språket.

6.2 Beskrivning av registreringsprocessen

6.2.1 Registrering med e-postadress

När användaren väljer att registrera sig på webbsidan med en mejladress skriver användaren in sin mejladress, förnamn, efternamn och samt ett lösenord som stämmer överens med lösenordskriterierna. Användaren behöver även bocka i att denne godkänner användarvillkoren, se figur 18.



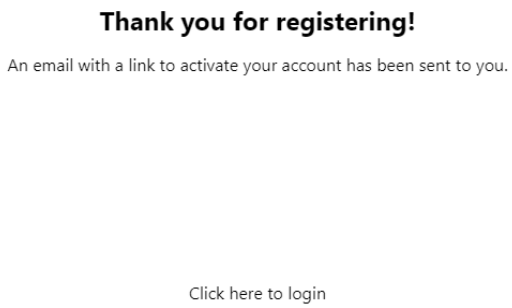
Figur 18. Loggin- och registreringsformuläret i mobilläge.

När användaren klickar på registrerings knappen säkerställer en Google Recaptcha att användaren inte är en robot. Kontouppgifterna paketeras in i ett JSON-objekt och skickas som en HTTP POST-förfrågan till en ändpunkt i REST-API:et som lyssnar efter förfrågningar (se figur 19).

```
[HttpPost]
[Route("SaveUser")]
0 references
public async Task<IActionResult> SaveUser([FromBody] UserItem model)
{
    return Ok(await _userService.SaveUser(model));
}
```

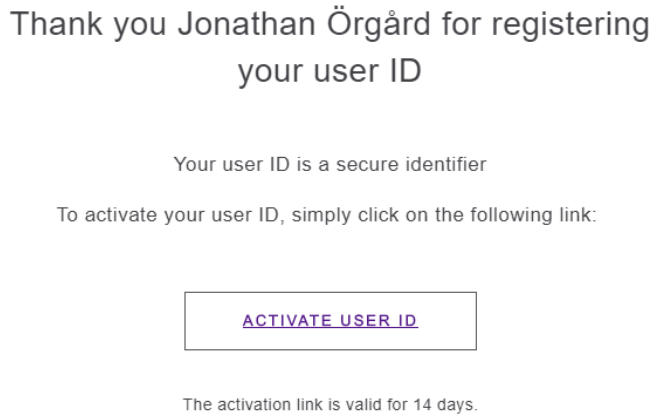
Figur 19. Ändpunkten för i REST-API:et för att spara ny användare.

REST-API:et skickar då en HTTP GET förfrågan för att få en tillgångsnyckel från Keycloak som tillåter REST-API:et att kommunicera med Keycloak. Samtidigt skapas en slumpmässig verifikationskod som kommer att användas vid verifikation av mejladressen. När tillgångs nyckeln har erhållits skickas den tillsammans med användarens användaruppgifter och verifikationskoden till Keycloak som en HTTP POST-förfrågan. Keycloak lagrar då användarens autentiseringsdata i dess LDAP-katalog samt resten av användardata (tex tidsstämpel för när kontot skapades) i sin egen databas. Om användaren lagras i Keycloak utan fel skickas ett verifikations mejl till användarens angivna mejladress. Ett popup fönster visas då på användargränssnittet som tackar användaren för registrering (se figur 20). Om registreringen misslyckas visas ett felmeddelande istället.

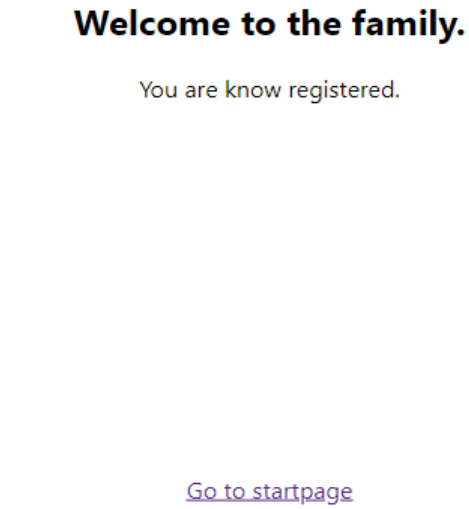


Figur 20. Lyckad registrering.

I verifikationsmejlet finns en länk som användaren är tvungen att klicka på för att verifiera sin mejladress och aktivera sitt konto för att sedan kunna logga in på webbsidan (se figur 21). Länken innehåller mejladressen och den genererade verifikationskoden för mejladressen. När länken klickats öppnas en webbsida som genast skickar en HTTP GET-förfrågan till REST-API:et med mejladressen och verifikationskoden. I REST-API:et skickas då en HTTP GET-förfrågan för att erhålla verifikationskoden som tidigare var inlagd i Keycloak för den specifika användaren/mejladressen. Om verifikationskoden i Keycloak matchar med verifikationskoden från mejlet skickas en HTTP POST-förfrågan till Keycloak som aktiverar användarkontot. Ett respons meddelande från Keycloak skickas tillbaka till användargränssnittet som beskriver om verifikationen är godkänd (se figur 22).

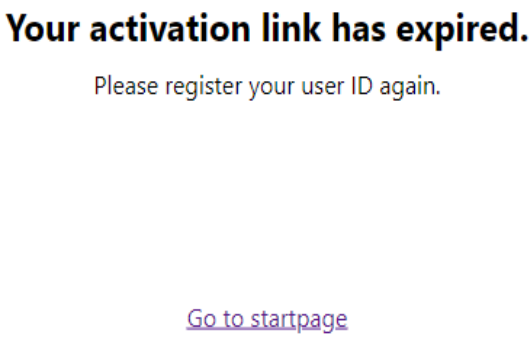


Figur 21. Verifikationsmejll.



Figur 22. Godkänd verifikation av mejladress.

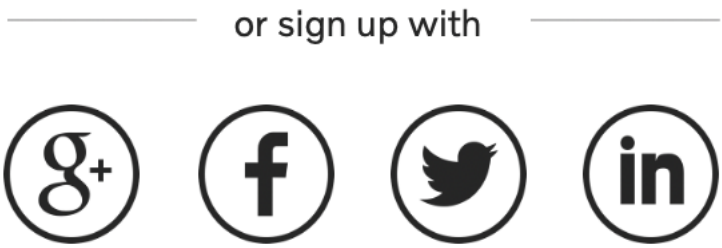
Om användaren inte verifierat sin mejl inom 7 dagar skickas ett påminnelsemejl och efter 14 dagar utan verifikation raderas användaren från Keycloak. Om användaren klickar på länken i verifikationsmejlet efter att det gått ut visas ett felmeddelande (se figur 23).



Figur 23. Misslyckad aktivering.

6.2.2 Registrering med social inloggning

När användaren väljer att ansluta sitt sociala plattformskonto för att registrera sig på webbsidan så klickar användaren på ikonen för den sociala plattformen som den vill ansluta (se figur 24). Twitter, Google, Facebook och LinkedIn stödjs som sociala plattforms inloggningsalternativ. Detta då de är populära tredjeparts autentisering för kundsektorn inom EMEA och AMERICAS. Samtliga plattformar använder sig av Oauth för registrering fast med lite olika implementationer.



Figur 24. Klickbara ikoner för registrering med sociala plattformar.

Twitter använder sig av Oauth 1.0a protokollet vid inloggning medan Google, Facebook och LinkedIn använder Oauth 2.0 protokollet. När användaren klickat på ikonen för dess valda sociala plattform så påbörjas Oauth flödet. För beskrivning av dessa flöden se kapitel 2.1.3. När Oauth-flödet hade utförts skickas en HTTP GET-förfrågan med tillgångsnyckelns som erhålls från Oauth-flödet till den sociala plattformens API för att få tillgång till användarens användardata.

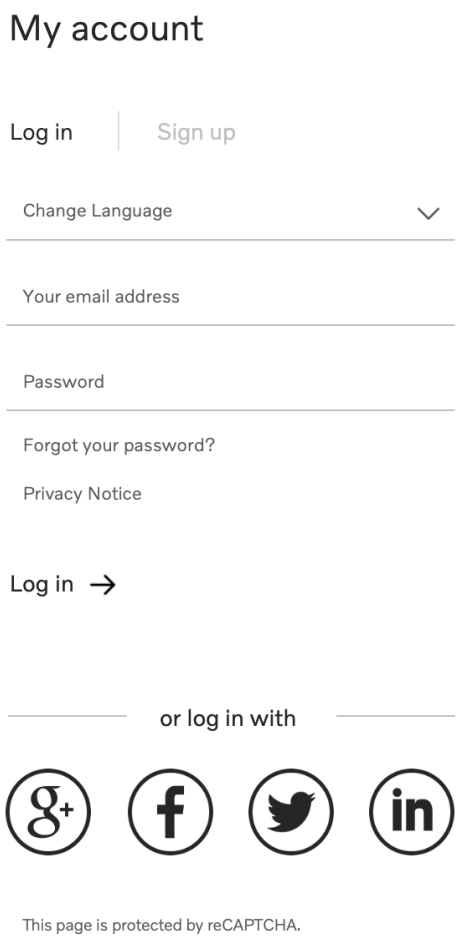
Användardata från den sociala plattformen samt vilken sociala plattform som använts paketeras in i ett JSON-objekt och skickas som en HTTP POST-förfrågan med en ändpunkt i REST-API:et. Rest-API:et ber först om en tillgångsnyckel från Keycloak som tillåter REST-API:et att kommunicera med Keycloak. När tillgångsnyckeln har erhållits skickas den tillsammans med användarens användaruppgifter i en HTTP POST-förfrågan till Keycloak som lagrar användaren i dess LDAP-katalog samt annan användardata i sin egen databas.

Inget verifikationsmejl används vid registrering med sociala plattformar eftersom användaren redan förväntats ha verifierat sin mejl på sin sociala plattform.

6.3 Beskrivning av inloggningsprocessen

6.3.1 Inloggning med e-postadress

När användaren väljer att logga på webbsidan med en egen mejladress så skriver användaren mejladressen och lösenordet i inloggningsformuläret (se figur 25).



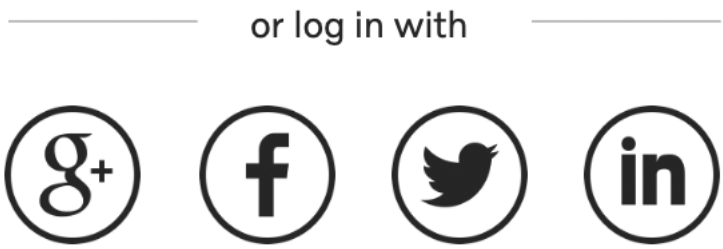
Figur 25. Inloggningsformuläret.

När användaren sedan klickar på inloggningsknappen säkerhetsställer en Google Recaptcha att användaren inte är en robot. Kontouppgifterna paketeras sedan in i ett JSON-objekt och skickas som ett HTTP POST-förfrågan till en ändpunkt i REST-API:et.

Användaruppgifterna skickas då till Keycloak som en HTTP POST-förfrågan. Keycloak kontrollerar om användaruppgifterna matchar en användare. Om en användare hittas kontrollerar Keycloak att användarkontot är aktiverat. Om användaren skriver in fel användaruppgifter eller om användarkontot inte är aktiverat skickas ett felmeddelande tillbaka till användargränssnittet som visar ett felmeddelande och inloggningen misslyckas. Om användaren lyckas logga in sparas en tillgångsnyckel och en uppdateringsnyckel som kakor i användarens webbläsare. Se kapitel 6.3.3 för hur dessa nycklar används.

6.3.2 Inloggning med social inloggning

När användaren väljer att logga in med sitt sociala plattformskonto på webbsidan så klickar användaren på ikonen för den sociala plattformen som den vill logga in med (se figur 26). Handskakningen med den sociala plattformen fungerar precis som vid registrering med ett socialt plattformskonto. Se kapitel 6.2.2.



Figur 26. Klickbara ikoner för inloggning med de stödda sociala plattformarna.

Men om användaren redan har ett socialt plattformskonto kopplat till webbsidan och bara vill logga in så behövs inget nytt konto skapas. Utan användardatan som erhållits från den sociala plattformen används istället för att identifiera kontot på Keycloak och vid inloggning så kontrolleras användarens uppgifter från den sociala plattformen för att se om något har ändrats. Tex om användaren har bytt mejladress på sitt sociala plattformskonto så uppdaterades även mejladressen kopplat till Keycloak kontot.

6.3.3 Inloggning med sessionskakor

Om användaren redan har loggat in i en tidigare session kan det finnas en tillgångsnyckel eller en uppdateringsnyckel lagrad som en kaka i användarens webbläsare (se figur 27). En tillgångsnyckel används för att autentisera sig som användaren mot Keycloak medan en updateringsnyckel används för att få en ny tillgångsnyckel. Detta leder till att dessa kan användas för att automatiskt logga in användaren. Efter inloggning är en tillgångsnyckel giltig i 7 dagar och en updateringsnyckel i 14 dagar. Vid utloggning raderas tillgångs- och uppdateringsnycklarna från webbläsarens kakor så att användaren inte loggas in igen.

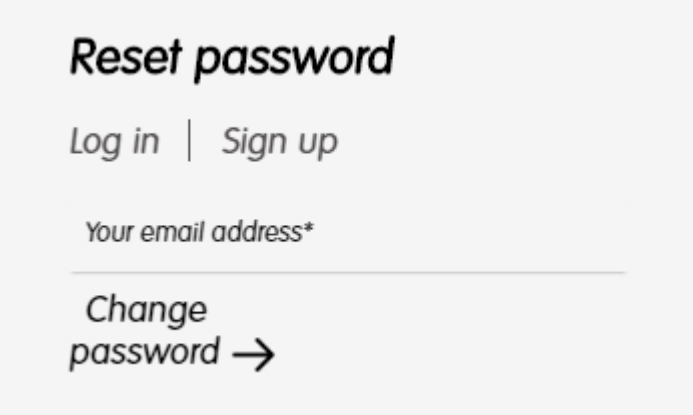
Name	Value	Domain	P...	Expires / Max-Age	Size
refresh_token	eyJhbGciOiJIUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICI2ZDJkNjk3ZC00Mjk5LTQ2N2ItOTMzZi1iNGJhODdjZWVjNTEifQ.ey...	localhost	/	2020-06-12T09:24:18.000Z	612
access_token	eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICI0Y0lzMi1ZVjdQWWh1M1N0MDdiR2M0NVhtQUlIWTg4SzFUckJF...	localhost	/	2020-06-05T09:24:18.000Z	1373

Figur 27. Tillgångs- och uppdateringsnycklar som kakor i webbläsaren.

Om någon av dessa finns vid anslutning till webbsidan så skickas en HTTP GET-förfrågan till REST-API:et med användarens nyckel. Nyckeln skickas senare vidare till Keycloak med en HTTP GET-förfrågan för att autentisera användaren. Om nyckeln är giltigt svarar Keycloak med nya tillgångs- och uppdateringsnycklar som rest API:et sedan skickar vidare som svar till webbläsaren och användaren loggas in. Däremot om autentiseringen misslyckats raderas nycklarna från webbläsarens kakor.

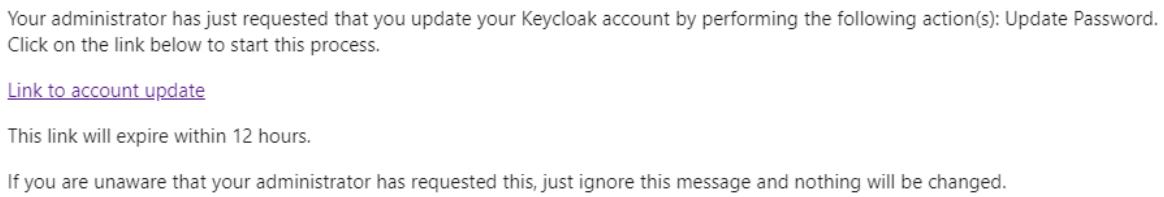
6.4 Återställning av lösenord

Om användaren glömt sitt lösenord så kan lösenordet återställas. Användaren kan göra detta genom att klicka på “Forgot password?”-knappen i inloggningsformuläret. Användaren hamnar då till ett lösenordsåterställningsformulär. Där får användaren skriva in sin mejladress och klickar sedan på “Change password”-knappen. Se figur 28.



Figur 28. Lösenordsåterställnings formuläret

När användaren klickar på återställlösenordsknappen skickas en HTTP GET-förfrågan med mejladressen till REST-API:et. Mejladressen skickas sedan vidare till Keycloak med en HTTP POST-förfrågan att sätta “UPDATE PASSWORD” flaggan på användarkontot kopplat till mejladressen. När denna flagga har satts så skickar Keycloak ett mejl till användaren som ber användaren att uppdatera sitt lösenord. Se figur 29.



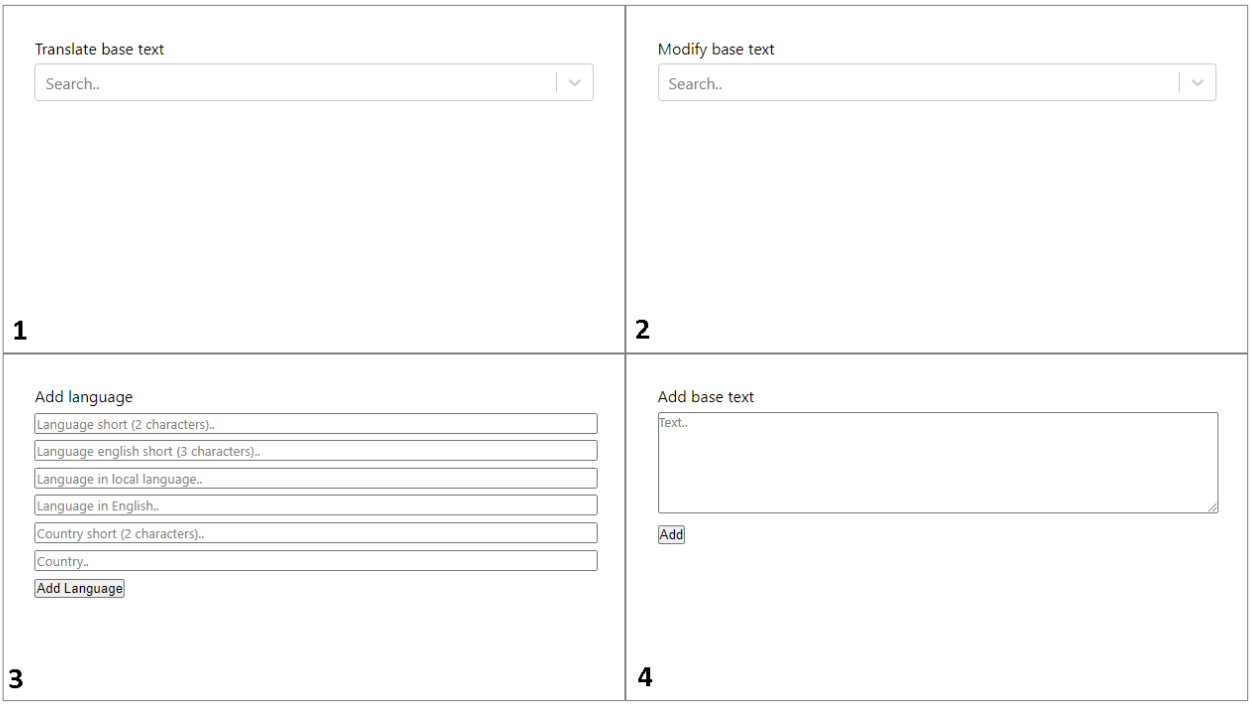
Figur 29. Lösenordsåterställnings mejlet.

Lösenordsåterställningslänken går ut efter 12 timmar och om användaren inte använder länken ändras inte lösenordet.

6.5 Översättningsmodulen

6.5.1 Lägga till översättning

För att lägga till ord för översättning finns ett användargränssnitt, se figur 30.



Figur 30. Översättnings användargränssnitt.

Alla översättningar refererar till en bastext. För att lägga till en bastext skriver användaren in en bastext i kolumn 4 i figur 30 och klickar sedan på “Add”-knappen. Den inlagda bastexten lagras i databastabellen dbo.Text.

Om användaren vill justera en bastext väljer användaren bastexten i kolumn 2 i figur 30 och utför sedan ändringarna. När ändringarna på bastexten är klara klickar användaren på “Save”-knappen. Den valda bastexten uppdateras då i databastabellen dbo.Text.

Vill användaren lägga till ett språk fyller användaren i värdena i kolumn 3 på figur 30 och klickar sedan på “Add Language”-knappen. Detta sparar språket i dbo.Language tabellen.

Om användaren vill lägga till eller redigera en översättning av en bastext används kolumn 1 i figur 30. Användaren väljer först bastexten som ska översättas eller redigeras. Användaren kan lägga till en ny översättning genom att klicka på “New translation”-knappen. Användaren skriver sedan in den nya översättningen och väljer sedan språket för den nya översättningen i språklistan. Om användaren däremot vill redigera en översättning klickar den på “Edit translation”-knappen efter valet av bastext. Användaren redigerar då översättningen och klickar sedan på “Save”-knappen när ändringen är redo att sparas. Vill användaren ta bort en översättning klickar den på “Delete translation”-knappen istället. Alla översättningar lagras i tabellen dbo.Language_Text.

6.5.2 Översättningsmodul frontend

För att användaren skall kunna byta mellan språken på webbsidan används JavaScript-biblioteket i18n [42]. i18n underlättar processen att byta språk som visas på webbsidan. Med i18n kunde man lägga in översättningar och även byta språket som visades dynamiskt. Om användaren väljer ett språk hämtas språket tillsammans med tillhörande text från MSSQL-databasen och lagras i i18n-biblioteket. När användaren sedan byter språk så letar i18n efter översättning på texten som tidigare har angetts. Om ingen matchning hittas används bastexten från programmeringskoden. Om språket däremot finns i databasen ändras texten på användargränssnittet (se figur31).

```
<Translation>{(t) => (t('Sign up'))}</Translation>
```

Figur 31. Exempel på i18n med bastext i HTML.

När användaren ansluter till webbsidan finns användarens senast valda språk i en webbläsarkaka. Om användaren inte tidigare har använt webbsidan eller om ingen kaka finns tillgänglig blir språket på webbsidan densamma som webbläsarens. Om språket inte har en översättning i databasen används engelska. När användaren byter språk på webbsidan så kontrollerades det om texten har skrivits i HTML-koden (se figur 31) och har en översättning på det specifika språket. Om ingen översättning finns i databasen så används texten som är skrivet i HTML-koden, vilket är den engelska texten.

6.6 Skalning med Kubernetes

För att implementera skalning användes Kubernetes. Projektets Docker-containrar exekverade med hjälp av Kubernetes som fanns inbyggt i Docker Desktop. Kubernetes tillåter användaren bland annat att manuellt skala systemet genom ökning och minskning av antalets kapslar som en gruppering använder.

6.6.1 Horisontell autoskalning

Kubernetes automatiska horisontella skalning testades på Node JS Docker-containern det vill säga användargränssnittet av projektets system. Gränsvärdena var att minst en kapsel (eng. pod) skulle finnas, samt att största antalet kapslar skulle vara fyra, där belastningen på varje kapsel skulle vara cirka 60% av en enskilds kapsels kapacitet.

För lokal testning simulerades ökningen av användare med HTTP-belastningsverktyget Hey [43]. Under testningen hade fyra av fyra möjliga kapslar skapats där vardera hade en resursbelastning på 57%, se figur 31. 5 minuter efter att testningen hade slutförts började antalet kapslar att minskas och efter ett tag användes bara en kapsel.

```
NAME       REFERENCE              TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
frontend   Deployment/frontend    57%/60%   1         4         4          4h7m
```

Figur 31. Kapslarnas (eng. pods) under en period av belastnings testet.

7 Resultat

Alla användarscenarion som nämns i kapitel 4, har implementerats och utförts. Vilket gjorde att resultatet blev en fungerande webbapplikation med ett grafiskt användargränssnitt, REST-API, databasstruktur, federeringsserver samt en LDAP-katalog. Webbapplikationen är modulanpassad då den är uppdelad i fem separata delar och varje modul är Docker anpassad.

Systemet uppfyller nedanstående funktionella krav:

- En användare kan registrera på webbapplikationen med sin mejladress
- Registrera på webbapplikationen med tredjeparts autentisering anpassad för en kundsektor inom EMEA och AMERICAS. Detta genom att använda sitt redan existerande sociala plattformskonto från Twitter, Facebook, Google eller LinkedIn.
- Ta bort sitt användarkonto
- Återställa sitt användarlösenord
- Byta språk på hela webbapplikationen.
- Användargränssnitt för att lägga till eller redigera översättningar på webbapplikationen

Systemet uppfyller nedanstående icke-funktionella krav:

- Systemet kan skala upp moduler dynamiskt beroende på behov med hjälp av Kubernetes
- Varje modul är lätt anpassad till andra system då de använder sig av REST API förfrågningar för att kommunicera
- Lätt kört på olika system då alla moduler är Docker-baserad.

8 Diskussion

8.1 Genomförande

Scrum som förklaras under kapitel 3.2.1 användes som arbetsmetodik. Scrum passar väldigt bra för mjukvaruprojekt och tillät oss flexibilitet i vårt planerade. Planeringen blev mer flexibel då vi kunde planera två veckor i taget och snabbt ändra planer inför kommande veckor. Med mer traditionell arbetsmetodik hade vi behövt planera 16 veckor framåt och förhålla oss till det. Projektet delades upp i flera olika sprintar där varje sprint skulle göra klart olika användarscenario. Varje användarscenario sågs som ett delmål som skulle uppfyllas och leda oss till en klar produkt. Om vi märkte att vi hade tidsbrist och inte skulle klara ett av användarscenarierna fullt ut kunde vi minska på komplexiteten men samtidigt behålla samma mål. Om vi däremot hade tid över innan sprinten var över kunde vi utöka användarscenarion och därmed målet ytterligare.

Att jobba fullstack, det vill säga med både frontend och backend, gjorde att vi fick lära oss hela processen bakom utvecklingen av en webbapplikation. Naturligt blev det då också en lite längre inlärningskurva då vi fick lära oss mycket nytt, samt jobba med flera olika ramverk och programspråk.

Implementation av två olika användargränssnittdesigner kan i efterhand ses som onödigt. Anledningen till två olika användargränssnitt var att den ena var tänkt för den kinesiska marknaden där det används andra typer av sociala konton. Men krångligheter med att införskaffa tillstånd för utveckling med kinesiska sociala konton gjorde att tanken gavs upp. Lyckligtvis tog det inte alldeles för lång tid att implementera den andra designen av användargränssnittet då den var baserad på den första.

8.2 Resultatet

Mycket nytt behövdes lära för att utvecklingen av projektet skulle kunna gå framåt vilket gjorde att arbetet tog längre tid än planerat. Detta ledde till tidsbrist för implementation av molnskalning, samt att faktumet att molnskalning kostar pengar var en bidragande faktor.

När autoskalningen testades lokalt uppstod det många problem och krascher på grund av bristande hårdvara på den exekverande datorn. Detta gjorde att testning på skalningen utföras på en ytterst liten nivå. Detta gjorde också att endast en del av vårt system kunde skalas åtgången och inte testa skalning av hela systemet samtidigt.

Vårt system är modulärt och uppbyggt i fem olika containrar. Detta gör att implementationen av skalningen underlättas och blir effektivare. Modulariteten gör det även enklare att ändra på en av containrarna utan att de andra delarna av systemet påverkas.

Funktionen för att kunna använda sociala konton för registrering och inloggning är en fördel gentemot andra implementationer som finns i branschen. Detta anser vi förbättra konsumentupplevelsen nämnvärt.

Vår översättningsmodul gör det även enkelt för kunder att byta språk på hemsidan. Detta gör att olika instanser med olika språk av webbsida inte behöver skapas och underhållas separat. Samt att konsumenten kan känna sig tryggare genom att använda webbapplikationen med ett språk de känner sig mest bekväma med. Detta bidrar till en mer tilltalande användarupplevelse. Översättningsmodulen gör att webbapplikationen är användbart oberoende på region i världen.

8.3 Miljöaspekt

Molntjänster som agerar värdar för hemsidor har negativ miljöpåverkan och är energikrävande och bidragande till koldioxidutsläpp. Eftersom applikationen är modul- och containeriserad så går det effektivt att skala upp de moduler som har stor efterfrågan och sedan skala ner när efterfrågan är mindre. Detta leder till att resurser inte slösas på att starta kloner av hela projektet och att flera instanser av moduler inte behöver vara igång i onödan. Detta kan bidra till en mindre påfrestning på servrarna och skapa en effektivare energiförbrukning.

8.4 Vidareutveckling av webbapplikation

Webbapplikation är en väldigt bra grund för vidareutveckling inom många områden. Då webbapplikationen är fokuserad på autentisering så är det väldigt öppet med vad applikation kan användas till. Den kan till exempel utökas så att användaren ser information om sin bil, boka service, besiktning eller tvättning efter att användaren har autentiserats.

Eftersom applikationen är skapad med containrar och skalning av dessa i åtanke så kan applikationen vidareutvecklas till molnskalning. I dagsläget skalas applikationen endast på en lokal maskin, men grunderna för molnskalning finns klara. Alla IP adresser för koppling mellan de olika containrarna är skrivna som systemvariabler och kan dynamiskt sättas i Kubernetes.

De sociala plattformarna som stöds för social inloggning i dagsläget är främst riktade mot AMER/EMEA regionerna. Det är inte alla regioner i världen som använder sig av dessa sociala plattformar så man skulle kunna lägga till stöd för andra sociala plattformar i andra regioner.

I dagsläget skapas ett nytt konto för varje social plattform som en användare loggar in med. Detta är inte optimalt då en användare kanske vill kunna logga in till samma konto på webbapplikationen med hjälp av olika sociala plattformar eller via vanlig inloggning. Detta kan förbättras genom att be användaren koppla sitt sociala plattformskonto till ett vanligt konto vid inloggning, om användare inte har ett vanligt konto redan kan man be användaren att denne skapar ett sådant.

9 Källhänvisning

[1] Statista," Connected Cars - Statistics & Facts", 2020. [Online] Tillgänglig: https://www.statista.com/topics/1918/connected-cars/, hämtad 2020-06-15.

[2] RYAN PAUL, "OAuth and OAuth WRAP: defeating the password anti-pattern", 2010. [Online] Tillgänglig: https://arstechnica.com/information-technology/2010/01/oauth-and-oauth-wrap-defeating-the-password-anti-pattern/, hämtad 2020-06-01.

[3] Aaron Parecki, "OAuth 2.0 Simplified, Background", 2020. [Online] Tillgänglig: https://www.oauth.com/oauth2-servers/background/, hämtad 2020-05-28.

[4] Matt Raible, "What the Heck is OAuth?", 2017. [Online] Tillgänglig: https://developer.okta.com/blog/2017/06/21/what-the-heck-is-oauth, hämtad 2020-06-01.

[5] Kassandra Perch, "Everything You Wanted to Know About OAuth 2", 2016. [Online] Tillgänglig: https://auth0.com/blog/everything-you-wanted-to-know-about-oauth-2-but-were-too-afraid-to-ask, hämtad 2020-06-01.

[6] IETF, "RFC 5849, The OAuth 1.0 Protocol", 2010. [Online] Tillgänglig: https://tools.ietf.org/html/rfc5849, hämtad 2020-05-28.

[7] IETF, "RFC 6819, OAuth 2.0 Threat Model and Security Considerations", 2013. [Online] Tillgänglig: https://tools.ietf.org/html/rfc5819, hämtad 2020-05-28.

[8] IETF, "RFC 6749, "The OAuth 2.0 Authorization Framework", 2012. [Online] Tillgänglig: https://tools.ietf.org/html/rfc6749, hämtad 2020-05-28.

[9] Auth0, "OpenID Connect", 2020. [Online] Tillgänglig: https://auth0.com/docs/protocols/oidc, hämtad 2020-06-01.

[10] PingIdentity, "What is Single Sign On (SSO)?", 2019. [Online] Tillgänglig: https://docs.pingidentity.com/bundle/sob1564002312903/page/yqi1564002285683.html, hämtad 2020-06-01.

[11] Onelogin, "What is single sign-on?", 2020. [Online] Tillgänglig: https://www.onelogin.com/learn/how-single-sign-on-works, hämtad 2020-06-01.

[12] TechTarget, "federated identity management", 2020. [Online] Tillgänglig: https://searchsecurity.techtarget.com/definition/federated-identity-management, hämtad 2020-05-29.

[13] Techopedia, "Federation", 2020. [Online] Tillgänglig: https://www.techopedia.com/definition/2500/federation, hämtad 2020-05-29.

[14] Andy Zindel, "Federated Identity Management vs. SSO", 2020. Tillgänglig: https://www.centrify.com/blog/federated-identity-management-vs-sso/, hämtad 2020-06-01.

[15] W3, “Web Services Architecture”, 2004. [Online] Tillgänglig: https://www.w3.org/TR/2004/NOTE-ws-arch-20040211/, hämtad 2020-05-28.

[16] IETF, “SOAP: Simple Object Access Protocol”, 1999. [Online] Tillgänglig: https://tools.ietf.org/html/draft-box-http-soap-00, hämtad 2020-05-28.

[17] Oasis, “Web Services Reliable Messaging”, 2007. [Online] Tillgänglig: http://docs.oasis-open.org/ws-rx/wsrm/200702/wsrm-1.1-spec-os-01.pdf, hämtad 2020-05-28.

[18] Schreiner, G.A., Duarte, D. & dos Santos Mello, R. “Bringing SQL databases to key-based NoSQL databases: a canonical approach”. Computing 102, 221–246 (2020). doi: https://link-springer-com.proxy.lib.chalmers.se/article/10.1007%2Fs00607-019-00736-1

[19] Information technology — Database languages — SQL — Part 1: Framework, ISO/IEC 9075-1:2016

[20] Docker, “What is a Container?”, 2020. [Online] Tillgänglig: https://www.docker.com/resources/what-container, hämtad 2020-05-29.

[21] IBM, “Kubernetes vs. Docker: It's Not an Either/Or Question”, 2018. [Online] Tillgänglig: https://www.youtube.com/watch?v=2vMEQ5zs1ko, hämtad 2020-06-01.

[22] IronOrbit, “Scalability In Cloud Computing”, 2018. [Online] Tillgänglig: https://www.ironorbit.com/blog/scalability-in-cloud-computing/, hämtad 2020-06-01.

[23] Irene Maida, “CLOUD COMPUTING SCALABILITY: WHAT IS IT AND WHY IT’S IMPORTANT”, 2019. [Online] Tillgänglig: https://www.criticalcase.com/blog/cloud-computing-scalability-what-is-it-and-why-its-important.htm, hämtad 2020-06-01.

[24] Cloudvheckr, “Cloud vs Data Center: What is Scalability in Cloud Computing?”, 2020. [Online] Tillgänglig https://cloudcheckr.com/cloud-cost-management/cloud-vs-data-center-what-is-scalability-in-cloud-computing/, hämtad 2020-06-01.

[25] Docker, “Docker docs”, 2020. [Online] Tillgänglig: https://docs.docker.com/, hämtad 2020-05-28.

[26] Kubernetes, “Production-Grade Container Orchestration”, 2020. [Online] Tillgänglig: https://kubernetes.io/, hämtad 2020-05-28.

[27] Justin Ellingwood, “An Introduction to Kubernetes”, 2018. [Online] Tillgänglig: https://www.digitalocean.com/community/tutorials/an-introduction-to-kubernetes, hämtad 2020-06-02.

[28] Microsoft, “SQL-Server-2019”, 2019. [Online] Tllänglig: https://www.microsoft.com/en-us/sql-server/sql-server-2019, hämtad 2020-05-28.

[29] Oracle, “What a Relational Database Is”, 2020. [Online] Tillgänglig: https://www.oracle.com/database/what-is-a-relational-database/, hämtad 2020-05-28.

[30] The OpenLDAP Project, “The OpenLDAP Project Overview”, 2020. [Online] Tillgänglig: https://www.openldap.org/project/, hämtad 2020-05-28.

[31] IETF, “Lightweight Directory Access Protocol (LDAP): The Protocol”, 2006. [Online] Tillgänglig: https://tools.ietf.org/rfc/rfc4511.txt, hämtad 2020-05-28.

[32] JBoss, “Keycloak - About”, 2020. [Online] Tillgänglig: https://www.keycloak.org/about, hämtad 2020-05-28.

[33] OpenJS Foundation, “Run JavaScript Everywhere.”, 2020. [Online] Tillgänglig: https://nodejs.dev/, hämtad 2020-05-28.

[34] NpmJS, “About npm”, 2020. [Online] Tillgänglig: https://docs.npmjs.com/about-npm/, hämtad 2020-06-02.

[35] Facebook, “React A JavaScript library for building user interfaces”, 2020. [Online] Tillgänglig: https://reactjs.org/, hämtad 2020-05-28.

[36] Git, “Git”, 2020. [Online] Tänglig: https://git-scm.com/, hämtad 2020-05-28.

[37] GitHub, Inc, “GitHub”, 2020. [Online] Tillgänglig: https://github.com/, hämtad 2020-05-28.

[38] Microsoft, “Visual Studio Code - Code editing. Redefined.”, 2020. [Online] Tillgänglig: https://code.visualstudio.com/, hämtad 2020-05-28.

[39] Scrum, “WHAT IS SCRUM?”, 2020. [Online] Tillgänglig: https://www.scrum.org/resources/what-is-scrum, hämtad 2020-05-29.

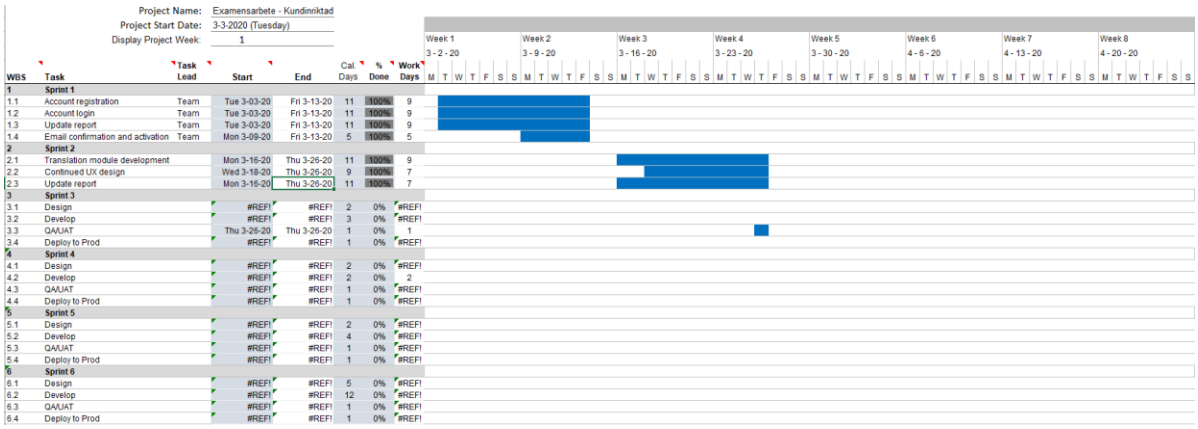
[40] Scrum guides, “Scrum Guide”, 2017. [Online] Tillgänglig: https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf, hämtad 2020-05-30.

[41] Atlassian, “Gitflow Workflow”, 2020. [Online] Tillgänglig: https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow, hämtad 2020-05-28.

[42] i18, “i18nex documentation”, 2020. [Online] Tillgänglig: https://www.i18next.com/, hämtad 2020-05-29.

[43] Rakyll, “HTTP load generator”, 2020. [Online] Tillgänglig: https://github.com/rakyll/hey, hämtad 2020-05-29.

10 Bilagor



Project Name:	Examensarbete - Kundinriktad
Project Start Date:	3-3-2020 (Tuesday)
Display Project Week:	1

WBS	Task	Task Lead	Start	End	Cal. Days	% Done	Work Days
1	**Sprint 1**						
1.1	Account registration	Team	Tue 3-03-20	Fri 3-13-20	11	100%	9
1.2	Account login	Team	Tue 3-03-20	Fri 3-13-20	11	100%	9
1.3	Update report	Team	Tue 3-03-20	Fri 3-13-20	11	100%	9
1.4	Email confirmation and activation	Team	Mon 3-09-20	Fri 3-13-20	5	100%	5
2	**Sprint 2**						
2.1	Translation module development		Mon 3-16-20	Thu 3-26-20	11	100%	9
2.2	Continued UX design		Wed 3-18-20	Thu 3-26-20	9	100%	7
2.3	Update report		Mon 3-16-20	Thu 3-26-20	11	100%	7
3	**Sprint 3**						
3.1	Design		#REF!	#REF!	2	0%	#REF!
3.2	Develop		#REF!	#REF!	3	0%	#REF!
3.3	QA/UAT		Thu 3-26-20	Thu 3-26-20	1	0%	1
3.4	Deploy to Prod		#REF!	#REF!	1	0%	#REF!
4	**Sprint 4**						
4.1	Design		#REF!	#REF!	2	0%	#REF!
4.2	Develop		#REF!	#REF!	2	0%	2
4.3	QA/UAT		#REF!	#REF!	1	0%	#REF!
4.4	Deploy to Prod		#REF!	#REF!	1	0%	#REF!
5	**Sprint 5**						
5.1	Design		#REF!	#REF!	2	0%	#REF!
5.2	Develop		#REF!	#REF!	4	0%	#REF!
5.3	QA/UAT		#REF!	#REF!	1	0%	#REF!
5.4	Deploy to Prod		#REF!	#REF!	1	0%	#REF!
6	**Sprint 6**						
6.1	Design		#REF!	#REF!	5	0%	#REF!
6.2	Develop		#REF!	#REF!	12	0%	#REF!
6.3	QA/UAT		#REF!	#REF!	1	0%	#REF!
6.4	Deploy to Prod		#REF!	#REF!	1	0%	#REF!

Gant schema på ungefärlig plan av arbetet.

language support
Balgariya
Belaruskaja mova
Bosanski
Čeština
Dansk
Deutsch
Eesti keel
Elliniká
English
Español
Français
Hrvatski
Íslenska
Italiano
ivrit
Latviešu valoda
Lëtzebuergesch
Lietuvių kalba
Limba moldovenească
Magyar nyelv
Makedonski
Nederlands
Nihongo
Norsk
Polski
Português
Русский
Românește
Shqip
Slovak
Slovenščina
Srpski
Suomen kieli
Svenska
Türkçe
Українська мова
Österreichisches Deutsch
한국어
中文

Språklista för översättnings modulen

English	Svenska
Log in	Logga in
Sign up	Skapa konto
Change Language	Välj spark
Your email address	Din e-postadress
Password	Lösenord
Forgot your password	Glömt lösenord
Or log in with	Eller logga in med
First Name	Förnamn
Last Name	Efternamn
Re-enter password	Ange lösenord igen
I accept the Terms and Conditions and processing of my personal data according to the Information notice	Jag accepterar villkoren och att Volvo Car behandlar mina uppgifter enligt Informationsmeddelandet.
Or sign up with	Eller skapa konto via

Översättning från engelska till svenska för texter som lagrades i översättnings databasen.