



UiT The Arctic University of Norway

Faculty of Science and Technology
Department of Computer Science

Integration of HelseID with Third-Party Role Assignments Data

Thomas L. Fagermyr

INF-3981 Master's Thesis in Computer Science - August 2021

Abstract

The Norwegian healthcare sector is vast, with a substantial number of organizations employing plenty of people, and thereof, 7,300 are customers of NHN. Maintaining and keeping up-to-date information about access rights for these customers is a difficult, time-consuming, and manual task, especially since organizations often change personnel without notifying the system administrators. The task is worsened since the access rights are often tied to an account using username and password as login credentials, which are easy to forget in the midst of many other login credentials.

This thesis proposes a proof of concept system for solving these problems. The proof of concept system utilizes HelseID, a unified login solution designed for health care personnel, offering authentication through known IDPs. Further, Altinn, a trusted third-party state enterprise, offers an API for service owners to extract information about organizational roles. The system acquires role assignments data from Altinn and delegates it to users when they are authenticated. The main goal is to automatically and dynamically keep information about access rights up-to-date, alleviate the task from system administrators, and make login simpler for the end-user by using a login solution they are familiar with.

Acknowledgements

I would like to thank my supervisors, Håvard Dagenborg Johansen, Espen Mæland Wilhelmsen, and Øyvind Guttvik Årnes, for being my supervisors, for all the helpful discussions, and for the great feedback during this thesis. Additionally, thanks to Norsk Helsenett for this opportunity.

Furthermore, I would like to thank my fellow study friends for great discussions and invaluable input throughout the years. Especially thanks to Eirik Haugen, who carried me through the semester when I fell ill. I could never have done it without you.

Finally, I would like to thank my friends and family for always being there and supporting me throughout the study. Special thanks to my Mother for always pushing me to do my best. Further, I would like to express my appreciation for my best friends, Frida Dalheim and Benjamin Bakli Aglen, for all the long talks and support. You have been paramount to my success, and I could not have accomplished this without you. Thank you.

Contents

Abstract	i
Acknowledgements	iii
List of Figures	vii
List of Tables	ix
List of Abbreviations	xi
1 Introduction	1
1.1 Norsk Helsenett	2
1.2 Problem definition	3
1.3 HelseID	4
1.4 Methodology	5
1.5 Interpretation, Scope, and Limitations	6
1.6 Terminology	6
1.7 Outline	7
2 Background	9
2.1 OAuth2	9
2.1.1 Roles	10
2.1.2 Grant Types	11
2.2 OpenID Connect	12
2.2.1 ID token	13
2.3 IdentityServer4	15
2.4 Summary	15
3 The NHN Customer Portal	17
3.1 Obtaining the data	18
3.2 Statistics of the customer portal	20
3.3 Key insights	24
4 Requirements Specification	27

4.1	System Functional Overview	27
4.2	System Model	28
4.2.1	The client	29
4.2.2	API	29
4.2.3	Back-end	29
4.3	Non-functional requirements	30
4.3.1	Security and Privacy	30
4.3.2	Reliability and Availability	30
4.3.3	Fault-tolerance	31
4.3.4	Dependency	31
4.3.5	Interoperability and Extensibility	31
4.3.6	Usability	32
4.3.7	Scalability and Performance	32
4.4	Summary	32
5	Design and Implementation	33
5.1	Determining the API	33
5.2	The client	35
5.3	API	39
5.4	Back-end	43
5.5	Summary	44
6	Evaluation and Results	47
6.1	Methodology and Methods	47
6.2	Experiments	48
6.2.1	Role delegation experiments	48
6.2.2	API Benchmark	51
6.3	Summary	52
7	Conclusion	53
7.1	Achievements	53
7.2	Concluding remarks	54
7.3	Future Work	54
A	JSON Response from Altinn API	57
A.1	Subject 24065500317	57
A.1.1	With roleDefinitionId as access controller (4)	57
A.1.2	Without role specification	58
A.2	Subject 28065501580	60
A.2.1	With roleDefinitionId as access controller (4)	60
A.2.2	Without role specification	61
	References	63

List of Figures

1.1	What HelseID can be used by. Figure altered from NHN. . . .	4
2.1	OAuth2 roles in the system. Figure altered from Microsoft. .	11
2.2	ID token payload data from an end-user in the proof of concept system.	14
3.1	A typical log entry in Splunk. Personal information has been redacted.	18
3.2	Six lookup requests within two seconds for a singular username. Personal information has been redacted.	19
3.3	All username attempts within a given time period	20
3.4	In-depth statistics of 82,296 valid usernames	21
3.5	In-depth statistics of 26,925 invalid usernames	23
4.1	Abstract architecture of the system model	29
5.1	Presentation of the MVC design pattern.	35
5.2	Code snippet for configuration of the HTTP client to support the API. API key redacted.	36
5.3	Test client login page displaying IDPs	37
5.4	Displayed reportees for subject 28065501580 in proof of concept system.	38
5.5	Request information with description about the reportee call. Figure from Altinn.	40
5.6	Reportee API call in the Altinn test environment using Postman. .	41
5.7	Reportee API response from the Altinn test environment using Postman.	42
5.8	ER-Diagram of model relations	44
6.1	List of reportees for subject 24065500317 in Altinn.	49
6.2	List of reportees for subject 24065500317 in proof of concept system.	50
6.3	API request for unregistered subject 15037104229	50

6.4	Response to the unregistered subject in the proof of concept system.	51
6.5	Latency benchmark of reportee API call done in Postman. . .	52

List of Tables

3.1	Displays the number of distinct users separated by user type	21
-----	--	----

List of Abbreviations

API Application Programming Interface

BRREG Brønnøysundregisteret

CGM CompuGroup Medical

HTTP HyperText Transfer Protocol

HTTPS Hypertext Transfer Protocol Secure

IDP Identity Provider

JSON JavaScript Object Notation

JWT JSON Web Token

MVC Model-View-Controller

NHN Norsk Helsenett

OIDC OpenID Connect

PID Personal Identifier

REGEX Regular Expression

REST Representational State Transfer

SQL Structured Query Language

SSN Social Security Number

UI User Interface

URI Uniform Resource Identifier

URL Universal Resource Location



Introduction

The healthcare sector is populated with large and complex organizations employing many people with various roles. Maintenance of access rights to computing infrastructure within these organizations is often a manual and error-prone task done by system administrators. It either requires the customer to inform about organizational changes or the system administrators to request this information at will, which can be tedious and may not coincide with all changes. As the number of healthcare principals requiring access rights increases, the task of requesting organizational information becomes substantially larger for the system administrators, to a point where it is not feasible to request information. In addition, access rights are often tied to an account consisting of username and password as credentials. These credentials are often forgotten among users if they are not used frequently; thus, aggravating the task of maintaining access rights by making system administrators spend their resources on managing the credentials.

Trusted state services, like the Norwegian national Altinn service and Brønnøysundregisteret (BRREG) registry platform, provide authoritative information on the structure of organizations, including the identity of people with key roles, like company owners and board members. This information identifies the root of trust for an organization and should reflect the rights assigned to objects within an organization's digital infrastructure. With the ongoing digitalization of trusted state enterprises, new official registries are becoming available online as web services. In addition, many trusted state enterprises already offer their own unique APIs for verifying and extracting information

dynamically about a given user or organization, like the ones described in Section 5. In this thesis, we propose using such services to maintain up-to-date information to manage internal rights.

The Norwegian national registry, BRREG, is a registry platform that consists of a large variety of registries. The concerned registries for this thesis are the registries containing information about Norwegian businesses, citizens, and government agencies. BRREG focuses on creating trusted, secure, organized, and high-quality sources of information for the Norwegian community. Furthermore, they aim to simplify the dialog and relation with the public for citizens and businesses.

Altinn [4] is a Norwegian web portal created to establish a good digital dialog for businesses, private individuals, and government agencies. Altinn started as a collaboration between Skatteetaten, Statistisk sentralbyrå and BRREG. Altinn was established to provide an alternative channel for reporting economic data. The growth of Altinn has steadily increased since it was established, and its collaboration has expanded to 63 unique service owners as of August 2021¹, which includes Norsk Helsenett. The current platform is used by over 90% of the Norwegian population and almost 100% of the businesses².

1.1 Norsk Helsenett

NHN is a Norwegian state enterprise that connects the Norwegian healthcare sector by establishing a secure digital arena for all of Norway's health sector actors. As a result, the Norwegian health sector can utilize this arena to communicate and exchange sensitive information safely and efficiently. NHN offers this service, along with other services, through a membership subscription.

NHN has customers all over the country and facilitates their services to every organ of the Norwegian healthcare sector. The customers range from hospitals to dental offices to psychologists and doctors' offices. All of whom have various needs and requirements for the services.

The services NHN offers to its customers can be seen in NHN's customer portal. It allows its customers to manage their business and membership. Further, it provides customers with various self-services such as e-mail accounts, placing orders on a variety of services, managing their bandwidth subscriptions, view invoices, and the home office solution, which allows health care personnel

1. <https://www.altinndigital.no/om-altinn/om-altinn-samarbeidet/>

2. <https://docs.altinn.studio/teknologi/altinnstudio/about/>

to access their work computer remotely for work applications, among other services.

1.2 Problem definition

NHN lacks up-to-date information about many of their customers because they change their administrative personnel without notifying NHN about the changes. NHN has more than 7300 customers, resulting in requesting this information from the customers a tedious and difficult task. In addition, contacting various customers can prove extremely complicated considering the information is already out of date.

NHN is often informed about organizational changes when an organization employee needs to use the customer portal and is unaware of their login information; thus, the customer contacts NHN's customer service for assistance. For example, if an employee from the health care principal contacts NHN's customer support to obtain the login information, they cannot receive it without proving they are whom they claim to be. Furthermore, if NHN's contact information for that customer is outdated, the support center can only supply the customer with its login information if a request is sent by e-mail from someone registered under that organization in BRREG.

NHN's support center creates and delegates various customer accounts, including administrative, e-mail, and home office accounts. Furthermore, the administrative accounts can use self-services in the customer portal to create e-mail, and home office accounts for their own organization. Username and password are always delivered by two separate mediums: username over e-mail and password on SMS. This mechanism is in place to ensure that any individual with malicious intent cannot gather the login information.

This thesis proposes a proof of concept system that aims to solve the problem of outdated rights access information and missing or forgotten login information for NHN's customers. The goal is to solve this problem by integrating HelseID [1] with third-party role assignments data from BRREG's APIs [7, 9] or Altinn's service owner API [3]. HelseID allows health care personnel to log in using their Social Security Number (SSN) through a well-known IDP such as BankID [6] or Buypass. With these APIs, the system will automatically assign the appropriate role to any users given they have registered affiliations, thus alleviating NHN from creating and delegating users.

Ideally, the system should work across other platforms and be a plug-and-play system to allow the users to navigate NHN's various platforms with ease. The

goal is to make it easier for the customers to log on to NHN's Customer Portal, decrease unnecessary workload for the support center, and create better and more up-to-date information for NHN about its customers.

1.3 HelseID

HelseID[1] is a unified login system targeted at the healthcare sector in Norway. Its purpose is to provide a secure and simple login for health care personnel. It eliminates the necessity for usernames, as it utilizes the SSN for identification through known IDPs. Various applications can use the service; for instance, it is used on NHN's registry platform (<https://register.nhn.no>).

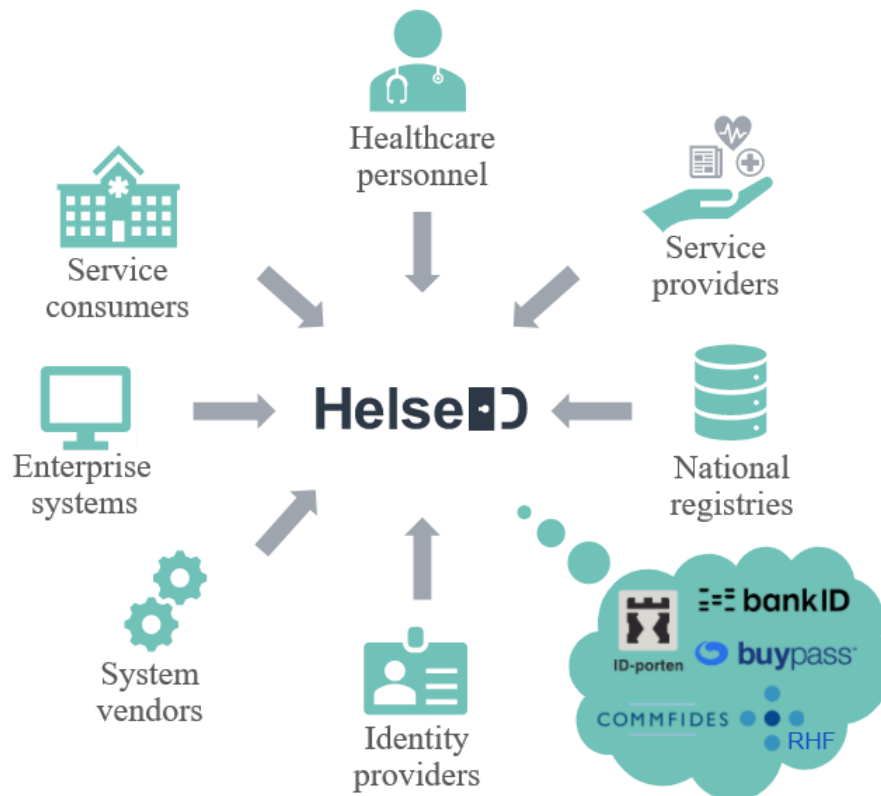


Figure 1.1: What HelseID can be used by. Figure altered from NHN.³

HelseID is an authorization server that utilizes IdentityServer4 as its core

³. <https://www.nhn.no/samhandlingsplattform/helseid/hva-er-helseid>

component, and it is based on the security protocols OAuth2 and OpenID Connect. HelseID provides services such:

Authentication as a Service offers centralized login logic and workflow for applications.

Single Sign-on / Sing-out allowing the user to have one set of login credentials that can be used to access multiple applications.

Access Control for APIs by issuing access tokens for the client.

Federation Gateway allows for external IDPs. HelseID is a federation gateway that allows for internal and external IDPs. Internal providers can be regional health trusts and municipalities. External providers are the likes of Buypass, BankID, and Commifides.

HelseID utilizes the SSN of the user as a known claim. This presents the ability to use this identifier with an API to get information regarding the subject from a third-party application such as the API from Brønnøysundregisteret or Altinn.

1.4 Methodology

According to the final report of the ACM Task Force on the Core of Computer Science [10], the computing discipline is divided into the following three major paradigms:

- *Theory* is rooted in mathematics. Its approach is to define problems, propose theorems, and determine whether the relationships are true to interpret the results.
- *Abstraction* is rooted in the experimental scientific method. Its approach is to investigate a phenomenon by forming a hypothesis, constructing models, and make predictions. Then experiment and collect data from interpreting results.
- *Design* is rooted in engineering. Its approach consists of three parts; To construct a system or device to solve a defined problem by stating the requirements and specifications. Then, design and implement the system or device. Finally, testing and evaluation of the system or device are performed according to the requirements and specifications.

Out of the three paradigms, the design paradigm is the most suitable for this thesis which describes a proof of concept for using HelseID with third-party assignments data for role delegation.

1.5 Interpretation, Scope, and Limitations

This thesis aims to develop a system that helps keep organizational access rights and information up-to-date for NHN about its customers and simplify the login system for end-users. The main focus will be to prove that the system can automatically obtain the information without the interference of a system administrator. Furthermore, the system dynamically collects the information when the user logs into the login solution, similar to ID-porten, familiar to the end-user, making it simple to use. In addition, the number of requests sent to the customer service will lessen due to the system being automated.

The thesis will resolve the problems that are stated by designing and implementing a proof of concept system. The system will include login through known IDPs familiar to the user and a simple User Interface (UI) to present the affiliations and access rights. In addition, it will have a backend component in the form of a simple Structured Query Language (SQL) database to store user identities. This is to reduce the number of requests sent to the API and limit dependency on the API.

The system will be evaluated according to the requirement specification stated in Chapter 4. The goal is to prove that the system adheres to its specification and limitations.

The limitations of the system will mainly be dependant on the API. When using an API from a third party, there will always be some limitations regarding what a developer can and cannot do. For instance, gaining access to read all the necessary data can be a challenge, as discussed in Section 5.1

There are features beyond the scope of this thesis. The future work section describes such potential features. The evaluation of this thesis will focus on testing how the client and API handle various people with different affiliations.

1.6 Terminology

Terminology important for this thesis includes:

Claim: Assertions that one subject (e.g., a user or an Authorization Server) makes about itself or another subject⁴.

Client: An application that requests access to a protected resource on behalf of and with the authorization of the resource owner.

Component: An entity with its own functionality that is part of a larger system.

End-user: An end-user is a person that uses the system, may be referred to as user and resource owner.

Reportee: A legal entity that a user can represent and act on behalf of.

Scope: Scopes are a group of claims.

1.7 Outline

The remainder of this thesis is structured as follows:

Chapter 2 presents relevant information about the technical background for the thesis.

Chapter 3 analyzes the usage of the customer portal to show the need for a new login solution and automated role delegation.

Chapter 4 presents the requirement specifications for the proof of concept system, including a system model, along with functional and non-functional requirements.

Chapter 5 describes the design and implementation of the proof of concept system.

Chapter 6 presents an evaluation of the system and results.

Chapter 7 presents future work and concludes the thesis.

4. <https://curity.io/resources/learn/scopes-vs-claims/>

/2

Background

This chapter describes the necessary technical background to understand the proof of concept system.

2.1 OAuth2

OAuth2[14] is an authentication protocol, which addresses the problems of the traditional client-server authentication model. The client uses the resource owner's credentials to authenticate with the resource server to access a protected resource using the traditional model. This means that for a third-party application to gain access to a protected resource, the resource owner will have to share its credentials with the aforementioned third-party application. This presents a range of problems.

Due to the resource owner having shared its credentials, the owner cannot control or restrict access to its protected resource. The application which has the credentials gains access to the entire protected resource. The resource owner cannot set a finite access duration or limit which resources the third-party application can access. The third-party application must store the resource owner's credentials for further use in the future, and servers must authenticate passwords. In any case, where the third-party application faces a data breach, the resource owner's credentials will be revealed. This renders the protected resource as being exposed.

The purpose of OAuth2 is to address all these issues. The protocol achieves this by creating an authorization layer and separating the client and resource owner into different roles. The protected resource is administered by the resource owner and is hosted by the resource server. When the client requests access to the protected resource, it is issued an access token rather than the resource owner's credentials. An access token is a string that represents an authorization granted for the client, and it is used to access protected resources. It contains information such as scopes, the lifetime of the token, and other access attributes. It is granted by the resource owner and enforced by both the resource and authorization server.

2.1.1 Roles

OAuth2 consists of four distinct roles:

- *Client* is an application that requests access to a protected resource on behalf of and with the resource owner's authorization.
- *Resource Owner* is the owner of a protected resource and can grant access to the resource. When a resource owner is a person, it is referred to as an end-user.
- *Resource Server* is the server responsible for hosting the protected resource. It can accept and respond to requests using an access token for the protected resource.
- *Authorization Server* is the server responsible for issuing the access token to the client. This applies after authenticating the resource owner and obtaining authorization.

The resource and authorization server are sometimes interchangeable. Meaning they can be the same server, and other times they are separate entities. It is also possible to have one authorization server that grants access tokens accepted by several resource servers.

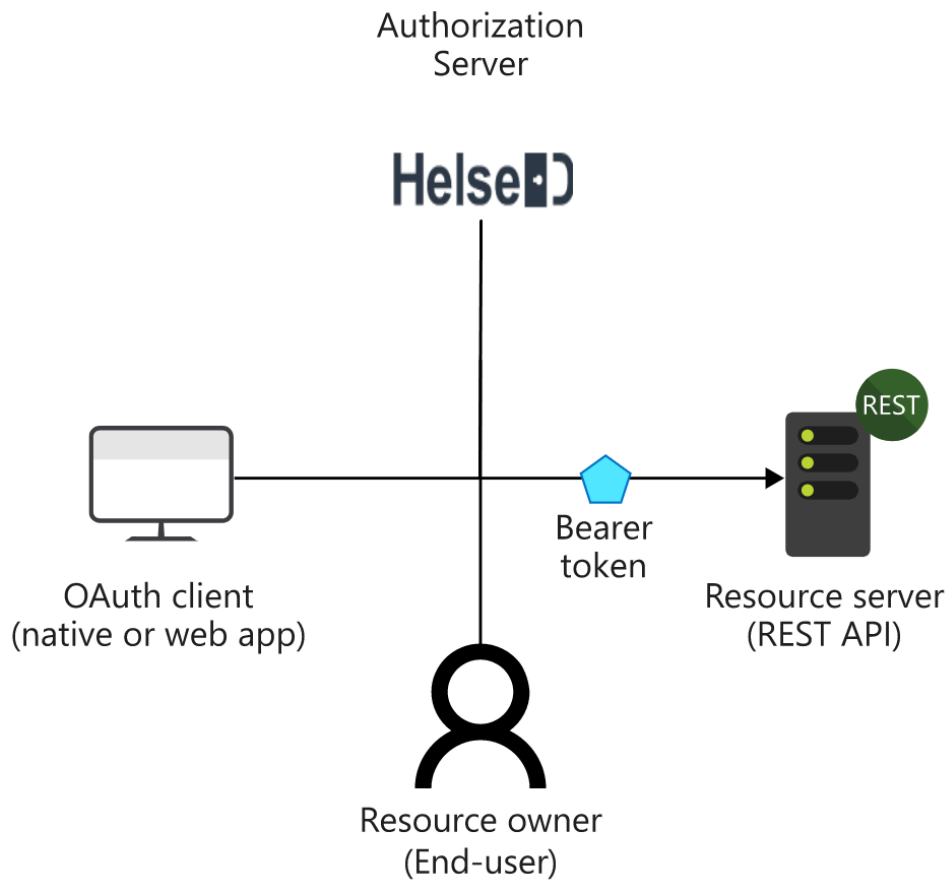


Figure 2.1: OAuth2 roles in the system. Figure altered from Microsoft.¹

In the case of this system, the resource owner is the end-user who logs onto the client. The authorization server is HelseID. HelseID is responsible for authenticating the end-user, granting and revoking access to restricted resources, and issuing tokens to the end-user. In addition, the resource server is the API owner. The client is configured to access the API along with an authenticated user.

2.1.2 Grant Types

Authorization grants, or grant types, are mechanisms used to represent the resource owner's authorization. The client uses the grant credentials to obtain an access token.

1. <https://docs.microsoft.com/en-us/azure/active-directory/develop/active-directory-v2-protocols>

- **Authorization Code** is an authorization grant that uses the authorization server as a middleman for an exchange between the client and resource owner. The client directs to the authorization server rather than requesting authorization directly. The resource owner is then directed back to the client with the authorization code.

The authorization server authenticates the resource owner and obtains authorization before directing it back to the client along with the authorization code. Thus, credentials are never shared between the client and resource owner since the authorization server authenticates the resource owner.

Authorization code is the most common authorization grant type, and the proof of concept application uses it.

- **Implicit Grant** is a simplified version of the authorization code grant. It is designed for browser-implemented clients using scripting languages like JavaScript. Rather than supplying the client with an authorization code, it is granted an access token.
- **Resource Owner Password Credentials** can obtain an access token directly. The only use case for this is when a high level of trust is established between the client and resource owner and when other grants are unavailable.
- **Client Credentials** is an authorization grant that can be used when the client requests access for a protected resource in which the client is the resource owner. In addition, the client credentials can be used when the client is acting on its own behalf.

2.2 OpenID Connect

OpenID Connect (OIDC) [21] is an identity layer that is situated on top of OAuth2. In OIDC, the identity of an end-user is verified by the client through the authentication performed by the authorization server. The client can obtain user profile information through authentication in an interoperable and similar manner to Representational State Transfer (REST) [13, 12]. Its interoperability stems from having a defined way of requesting and responding to claims. It contains some standard scopes, such as `openid`, `profile`, and `email`, an ID token that describes information about the authenticated user, and it has a `userinfo` endpoint. The `userinfo` endpoint is used to get attributes about the user and to translate tokens.

2.2.1 ID token

An identity token, or ID token, is a security token that is represented as a JSON Web Token (JWT). The ID token consists of claims about the end-user which stem from the authorization server's authentication of the end-user. In addition, the ID token may contain other claims.

The required claims and their explanations of an ID token are:

iss is the issuer and represents who signed and created the token.

sub is the subject that the token refers to.

aud represents the intended recipient(s) of the token.

exp is the expiration time of the token represented in seconds since Unix epoch.

iat is the time the token was issued at represented in seconds since Unix epoch.

ID token payload

```
{
  "nbf": 1625736971,
  "exp": 1625737271,
  "iss": "https://helseid-sts.test.nhn.no",
  "aud": "246d8675-176e-405d-a9ed-00368d3728cc",
  "nonce":
"637613337628362507.NjJiMWNjZjYtODdiZC00NzczLWE4YTQtOGY
5MDk5ZjMyNWZyZBh0WIyZTEtNm4Zi00Zjk4LWE3MzAtNDM2YzY2Nz
U4MzVh",
  "iat": 1625736971,
  "at_hash": "J694IRx7KGttVnqBN30oZQ",
  "s_hash": "kdtrmUrdGVoosGTXt9pySg",
  "sid": "78E4B1E614315F319D5519DF8172096D",
  "auth_time": 1625736970,
  "idp": "testidp-oidc",
  "helseid://claims/identity/security_level": "4",
  "helseid://claims/identity/pid": "28065501580",
  "helseid://claims/identity/assurance_level": "high",
  "helseid://claims/identity/pid_pseudonym":
"aDTZFZ4ui6cIpmaErSsHwdjeCpaqXkHtxsbIGksuC+Q=",
  "helseid://claims/identity/network": "internett",
  "sub":
"fjYP+S6uvnl2ENED16nL9mqiT8tJ4q7MPelfHPrB0hM=",
  "amr": [
    "external"
  ]
}
```

Figure 2.2: ID token payload data from an end-user in the proof of concept system.

Figure 2.2 presents the payload of an ID token directly from the proof of concept system using a fictional test subject. It contains the standard claims of an ID token along with the HelseID claims. The HelseID claims establish information about the user's identity. As seen in the figure, the claim relates to the user's security and assurance level and Personal Identifier (PID). The PID is the user's SSN. Security and assurance levels can determine what a given user can access, but it is not relevant to this proof of concept system.

2.3 IdentityServer4

Security can be divided into two main parts; Authentication and API Access.

Authentication is required when the current user's identity has to be known by the application. When an application requires authentication, it often acts on behalf of the user. It has to verify that the user is allowed to access the information it is attempting to access.

API Access is how this information is gathered or accessed. There are two main ways for an application to utilize an API - either by using the user's identity or its identity (the application).

OpenID Connect and OAuth2 offer solutions to these concerns and are designed to work together. As mentioned in Section 2.2 OIDC is an extension that sits on top of OAuth2. Although the protocols offer a solution to the concerns, it can be a rather difficult and time-consuming task to implement; therefore, IdentityServer4 was developed.

IdentityServer4 is an open-source OIDC and OAuth2 middleware framework for ASP.NET Core. It is an officially certified implementation of OpenID Connect[20]. The purpose of IdentityServer4 is to act as a central authentication server for various applications and simplify the use of OIDC and OAuth2 in your application. The framework offers a range of features. These features are; Authentication as a Service, Single Sign-on / Sign-out, Access Control for APIs, and Federation Gateways.

2.4 Summary

This chapter introduced technical background information for the thesis. The protocols, OAuth2, and OIDC, utilized in HelseID have been presented and explained. Further, the IdentityServer4 framework which HelseID is built upon has also been described.

/ 3

The NHN Customer Portal

Kundeportalen [26] is NHN's current customer portal, which is a website where customers can manage their business and membership. It provides customers with various self-service functions like the home office solution, which allows health care personnel to access their work computer remotely for work applications. The customer portal also provides self-services such as e-mail accounts, placing orders on various services, managing their bandwidth subscriptions, view invoices, and much more. Orders are placed in the portal using an account with administrative rights created by NHN, and the account is given to the administrative contact of the customer's principal. Unfortunately, health care principals, perhaps except for hospitals, often have outdated information about who can make administrative decisions for a given principal. When someone tries to place an order without being the registered administrative contact or the administrative contact is outdated, the problem occurs.

To better understand the necessity for a new system and how to design it, it is necessary to study the current system. For example, one can determine if there is a need for a unified login system by studying the current system. In addition, it is helpful when determining the requirements and specifications for the new system. For example, the current implementation uses an account lookup API, which triggers anytime a user attempts to log in, designed to check whether the account is valid or not.

3.1 Obtaining the data

Splunk [15] is a software platform developed for analyzing, visualizing, and searching through machine-generated log data from various sources within a business in real-time. It was created to make machine-generated log data easily accessible and manageable. The tool maintains a vast data lake consisting of log data. It utilizes indexing to access log events and create smaller and more manageable data sets that can be used to produce statistics, graphs, dashboards, and alerts.

Log entry in Splunk

```
2021-04-08 13:36:53Z [Debug] () Response from GET to url "accounts/lookup.json?username=[REDACTED] was "StatusCode: 200, ReasonPhrase: 'OK', Version: 1.1, Content: System.Net.Http.StreamContent, Headers:
{
  Transfer-Encoding: chunked
  Connection: keep-alive
  Status: 200 OK
  X-Request-Id: ffc1d225-f076-490f-a3d1-b251d070bab5
  X-Runtime: 0.042696
  Strict-Transport-Security: max-age=31536000
  Cache-Control: must-revalidate, max-age=0, private
  Date: Thu, 08 Apr 2021 13:36:53 GMT
  ETag: W/"f30f9fb03119e2d71b44acb450c14f3f"
  Server: nginx
  Server: +
  Server: Phusion
  Server: Passenger
  X-Powered-By: Phusion Passenger
  Content-Type: application/json
}"; Response string content was "{\"status\":\"ok\",\"message\":\"Token match\",\"id\":\"53188\",\"name\":\"[REDACTED]\",\"username\":\"[REDACTED]\",\"mobile_phone\":\"[REDACTED]\",\"email\":\"null\",\"type\":\"Generic\",\"customer_id\":\"12848\",\"security_stamp\":\"[REDACTED]\",\"homeoffice\":{\"location_ids\":\"[49758]\",\"enabled\":\"true\",\"changed_by\":\"[REDACTED]\",\"cert_serial\":\"null\",\"is_admin\":\"true\",\"is_impersonated\":\"false\",\"token_delivery_methods\":\"[\"sms\"]\"}"
Collapse
host = ptrd-kpo-web01prod.drift.nhn.no | source = D:\Logs\KPO\WebApp\kpo-20210408.log | sourcetype = kpo_applog | username = [REDACTED]
```

Figure 3.1: A typical log entry in Splunk. Personal information has been redacted.

Figure 3.1 presents a log event of a valid response to an account lookup request. The log event displays all the information about the account in the response string formatted as JavaScript Object Notation (JSON). From the log event, we can make several observations. For instance, the account is an admin, it does not have an email address listed on it, it has access to the home office solution, and the home office solution has never used it due to the `cert_serial` field being empty (null). It is easy to study one log event to determine the values of its fields to gain a better understanding of the given account. Once this scales to millions of log events that need to be aggregated, studying log events by hand is no longer an option. It has to be automated; thus, Regular Expression (REGEX) is required to specify and sort the wanted information. REGEX is a sequence of characters designed to search and match text patterns.

The results presented below in this chapter stem from machine-generated log data stored in Splunk, and all the graphs were produced using the tools within Splunk. First, the data was gathered using the specific index and app log for the customer portal's log events. Then, custom-made REGEXes were created using

a REGEX generator [24] and applied to the appropriate log events to capture the wanted data. Finally, search macros were developed to extract the required information using Splunk functions combined with the REGEXes.

Closely related requests

Time	Event
08/04/2021 15:36:53.000	2021-04-08 13:36:53Z [Debug] () Doing GET towards *accounts/lookup.json?username=[REDACTED]. Baseaddress: https://minside-proxy.flow.nhn.no/api/v1/ host = ptrd-kpo-web01.prod.drift.nhn.no source = D:\Logs\KPO\WebApp\kpo-20210408.log sourcetype = kpo_applog username = [REDACTED]
08/04/2021 15:36:53.000	2021-04-08 13:36:53Z [Debug] () Doing GET towards *accounts/lookup.json?username=[REDACTED]. Baseaddress: https://minside-proxy.flow.nhn.no/api/v1/ host = ptrd-kpo-web01.prod.drift.nhn.no source = D:\Logs\KPO\WebApp\kpo-20210408.log sourcetype = kpo_applog username = [REDACTED]
08/04/2021 15:36:52.000	2021-04-08 13:36:52Z [Debug] () Doing GET towards *accounts/lookup.json?username=[REDACTED]. Baseaddress: https://minside-proxy.flow.nhn.no/api/v1/ host = ptrd-kpo-web01.prod.drift.nhn.no source = D:\Logs\KPO\WebApp\kpo-20210408.log sourcetype = kpo_applog username = [REDACTED]
08/04/2021 15:36:52.000	2021-04-08 13:36:52Z [Debug] () Doing GET towards *accounts/lookup.json?username=[REDACTED]. Baseaddress: https://minside-proxy.flow.nhn.no/api/v1/ host = ptrd-kpo-web01.prod.drift.nhn.no source = D:\Logs\KPO\WebApp\kpo-20210408.log sourcetype = kpo_applog username = [REDACTED]
08/04/2021 15:36:52.000	2021-04-08 13:36:52Z [Debug] () Doing GET towards *accounts/lookup.json?username=[REDACTED]. Baseaddress: https://minside-proxy.flow.nhn.no/api/v1/ host = ptrd-kpo-web01.prod.drift.nhn.no source = D:\Logs\KPO\WebApp\kpo-20210408.log sourcetype = kpo_applog username = [REDACTED]
08/04/2021 15:36:52.000	2021-04-08 13:36:52Z [Debug] () Doing GET towards *accounts/lookup.json?username=[REDACTED]. Baseaddress: https://minside-proxy.flow.nhn.no/api/v1/ host = ptrd-kpo-web01.prod.drift.nhn.no source = D:\Logs\KPO\WebApp\kpo-20210408.log sourcetype = kpo_applog username = [REDACTED]

Figure 3.2: Six lookup requests within two seconds for a singular username. Personal information has been redacted.

For all these requests, the service responded with HTTP status code 200-OK, meaning the system found the requested account.

Due to how the current login system is implemented, it can appear as if the user attempts to log in several times within the same second and sometimes a few seconds apart. In order to clearly define one login attempt, closely related log events for the same username are aggregated as a singular login attempt. In this case, closely related is a time frame within 5 seconds of each other. Figure 3.2 shows an extreme case of this phenomenon, with six account lookup requests occurring within two seconds of each other and four of them being within the same second. For the sake of documentation, this example will count as one login attempt since it is improbable that a singular user has attempted six unique login attempts in a two-second period.

REGEXes were used to extract the HTTP status from all of the responses sorted by usernames, meaning the responses had to contain a username to be evaluated. If a response was "200 OK", the username was valid. A "404 Not Found" response determined that the username was invalid. When this was in place, it was effortless to count the number of valid and invalid usernames that made the login attempts shown in Figure 3.3. This was further aggregated to go deeper into each of the two categories. REGEXes were applied to the valid usernames to determine what account types were the most common, which is

represented in Figure 3.4. The amount of distinct and valid usernames were recorded, represented in Table 3.1, as a supplementary table for the previously mentioned figure. Several REGEXes were applied to the invalid usernames to show better what usernames NHN's customers attempt in the customer portal, displayed in Figure 3.5.

3.2 Statistics of the customer portal

The collected data results from a two-year and three-month period from the 1st of January 2019 to the 9th of April 2021, equal to 829 days. Figure 3.3 represents all the attempted logins during the given time period. Valid users are successful attempts, meaning the username is correct. Invalid users represent all failed attempts due to an incorrect username: figures 3.4 and 3.5 display the valid and invalid attempts in further detail.

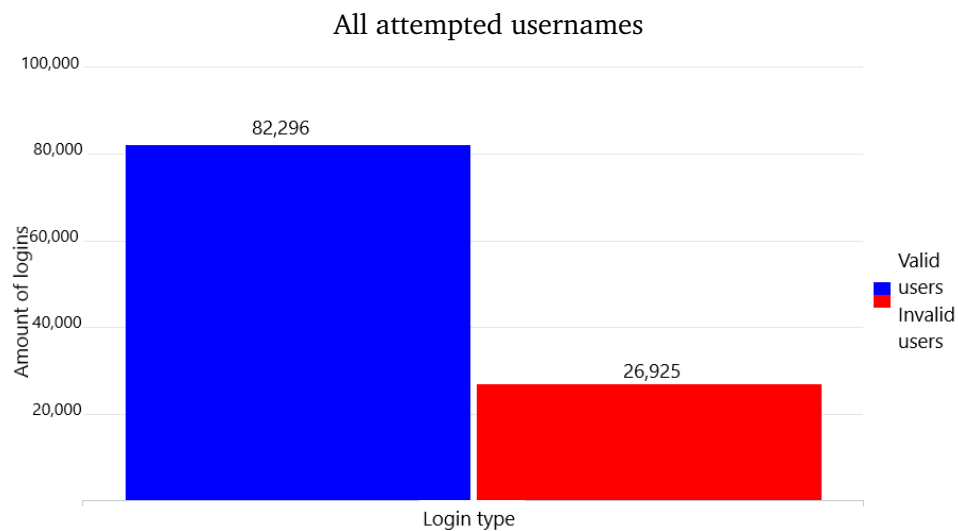
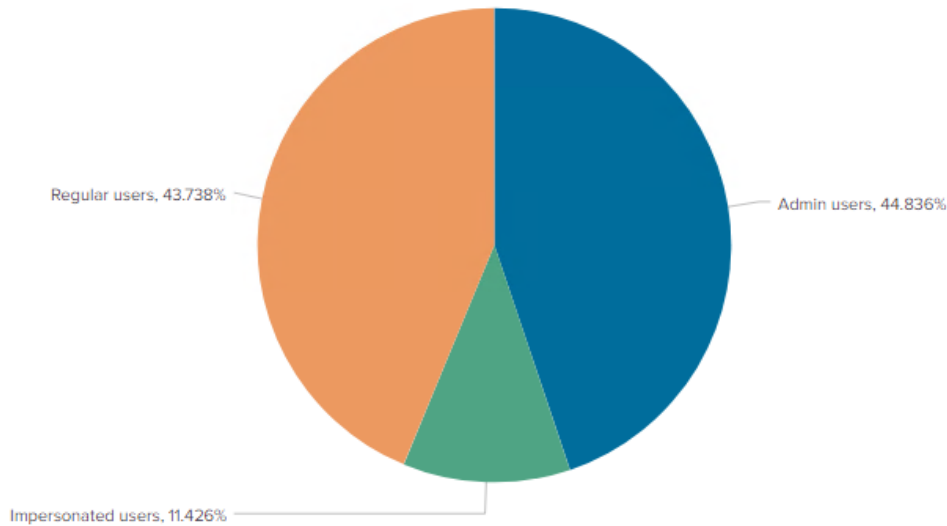


Figure 3.3: All username attempts within a given time period

A total of 109,221 attempted usernames were recorded within the given time period, with NHN having about 7300 customers as of the 9th of April 2021. This is about 132 login attempts per day, where several logins can result from the same individuals. 82,296 (75,3%) login attempts were successful and 26,925 (24,7%) of the attempted logins were invalid. It is expected that a majority of the valid users are a combination of avid home office users and recurring administrative accounts. A quarter of all attempted logins are invalid, which is quite a large chunk. It is expected that the invalid attempts can range from attempting the username of another NHN portal to a simple typo.

Statistics of valid login attempts

**Figure 3.4:** In-depth statistics of 82,296 valid usernames

As observed in Figure 3.4, out of the 82,296 valid users recorded, 44,8% are administrative users, 43,8% are regular users, and 11,4% of the login are impersonated users. As the name implies, administrative users are users with administrative rights who can make changes and place orders on behalf of their organization. Regular users are usually home office and e-mail accounts used by staff at its organization. They cannot make changes or place orders in the customer portal. Impersonated users are staff from the support center who have logged onto customers' accounts using an impersonate function implemented in the portal. This allows the support center to see exactly what the customer sees when they call for support, making it easier to understand their needs and aid them.

Distinct users table		
User type	Amount of users	Amount of home office users
Admin	2,845	1,181
Regular	5,455	4,437
Impersonated	3,361	
Total	11.661	5,618
Total w/o impersonated	8,300	5,618

Table 3.1: Displays the number of distinct users separated by user type

The valid login attempts consist of various individual users who have logged

in several times. Table 3.1 provides an overview of how many unique accounts have been successfully logged onto from the 1st of January 2019 to the 9th of April 2021. A total of 11,661 distinct accounts were recorded during the period, where 3,361 consists of impersonated users. Impersonates are mostly used by callees to aid the caller in real-time or for testing and experimentation purposes. Therefore, it is unnecessary to count the home office service users for impersonated users due to the nature of an impersonated account.

The remaining 8,300 unique users are actual customers who utilize the customer portal avidly. Less than a quarter (24,4%) of the total users are administrative accounts, and less than half of these (1,181 out of 2,845) have enabled the home office solution. Nevertheless, these administrative users account for 44,8% of the total successful logins. This helps to prove that the administrative accounts are used for administrative tasks such as account creation, placing orders, and checking documents. All of these are tasks that would require the same individuals to log in repeatedly.

Regular users, who account for 46,8% of the total users, are responsible for 43,8% of all valid logins. The majority of regular users are accounts with home office enabled, with 4,437 out of 5,455 having access to it. For the home office solution to work, the user must attach its buypass certificate serial number to the account. This is achieved by logging onto the customer portal and registering it by using a buypass login solution. Therefore, a regular user's only need for the customer portal is to register its buypass serial and change the password. In fact, most of the traffic generated from regular users consists of health personnel who want to utilize the home office or change their password in connection with either e-mail or the home office.

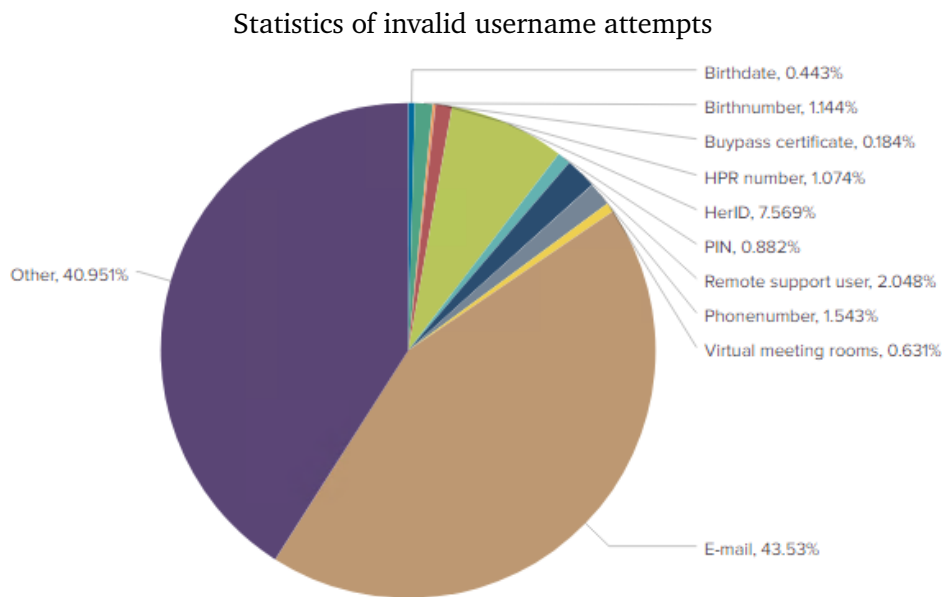


Figure 3.5: In-depth statistics of 26,925 invalid usernames

From the two years and three months of data, we observe that 24,7% of all authentication requests contain invalid usernames. Figure 3.5 displays the various usernames that were possible to sort out of all the invalid attempts. The vast majority of incorrect username attempts result from customers who attempt to log in with some sort of e-mail address (43,53%). The attempted e-mail addresses vary from personal, work, and NHN e-mail addresses. Customers likely attempt e-mail addresses because e-mails are often being used as a username for logins.

Another large chunk of the requests is HerID, which represents 7,57% of the requests. A HerID consists of digits and is utilized for secure communication between health care principals. HerID, as a login means, was an ID used for logging on to NHN's registry platform. The HerID part of the graph accounts for the organization users, consisting of the prefix "OrgUsr" followed by the organization number. The login consisted of the prefix "Her" followed by digits, which the REGEX has filtered in this graph. The HerID login provided administrative personnel to maintain and update their entry in NHN's Adresseregister (Address registry). HelseID replaced HerID as a login at the start of 2021, and previous owners could migrate their HerID account to HelseID. However, users attempt these out-of-date login credentials on the customer portal to this day.

It can be difficult to decipher what the user has attempted when logging in,

and sometimes it will be impossible. Some attempts to decipher and extract include personal information such as HPR numbers, birth dates, SSNs (birth number), phone numbers, PIN codes, and buypass certificates (totaling 5,27%). Customers have also attempted to log in using the ID for virtual meeting rooms (0,63%) and remote support accounts (2,05%), both supplied by NHN and used for other portals. The Other category accumulates all attempts that were not possible to extract using REGEX sufficiently. It contains all sorts of interesting attempts including but not limited to username typos, various passwords, and usernames meant for other NHN login portals and services. This provides evidence that customers are uncertain of their usernames and attempts all sorts of credentials that they can think of until they either give up or contact customer support.

3.3 Key insights

The customer portal is designed so that regular users only need it for self-services such as registering buypass serials and changing passwords. Most of the regular accounts are created for the sole purpose of utilizing the home office solution and/or being an e-mail account. However, administrative accounts utilize the same self-services along with several other services. For example, administrative accounts can create e-mail and home office accounts, place orders, and view important documents on behalf of its organization. Naturally, administrative accounts have repeating occurrences in the customer portal, as proven in Section 3.2, whereas regular accounts are used when self-services are required. Thus, administrative users, who are nearly half the number of regular users, make up nearly the same amount of logins as regular users do.

The valid users are often the customers who avidly use the customer portal, and they will oftentimes remember their login credentials. Those who rarely utilize it will often forget their login credentials or even not know they have this access. The results from Section 3.2 shows that a quarter of all login attempts were invalid. That is a huge margin considering there were only 11,661 distinct accounts over the course of 2 years and 3 months. Most of these are users who attempt any credentials they can think of to log in. This causes the customers to mix and match usernames within NHN because many portals and logins use different credentials. They will most likely either give up or call customer service for support when they cannot remember their login details. This generates lots of traffic for the service center, and it can be frustrating for the users.

This provides a need for a standardized login system, which in our case is HelseID. HelseID utilizes the SSN for identification. It is usually achieved

through IDPs such as Buypass or BankID. Buypass is used by most health personnel, thus making it easy for them to log in using this system. If they do not have a buypass card, they can use BankID, which most citizens in Norway have to access their bank, health journal, or taxes. Either way, they will have an easy and standardized way to log in to the customer portal, and they never have to remember their username. This eases the frustration on the customer's part of not knowing their login credentials. It also decreases customer portal-related traffic to the support center, which frees up resources to complete other tasks.

/4

Requirements Specification

This chapter outlines the requirements specification based on the problem definition in Section 1.2, the background knowledge provided in Chapter 2, and the information about the customer portal presented in Chapter 3. Both functional and non-functional requirements are stated, and the conceptual system model is described. Finally, an abstract overview of the proof of concept system and its main features and components are defined and outlined.

4.1 System Functional Overview

Some key functional requirements must be developed for the end-user to obtain its access and the correct role. These requirements are:

- **Authentication** of the user to guarantee that it is the correct individual that requests access.
- **Obtaining data** about the user from an API, and later from storage.
- **Role delegation** following the data collected from API.
- **Storage** of the end-users user identity for later use.
- **Presentation** of the data to the end-user.

- **Update** existing user identity in the event of affiliation changes.

Authenticating the user is necessary since the user requests access to restricted information that must maintain its security. In addition, authentication ensures that the user is who they claim to be; thus, they are allowed access to the resource. Authentication occurs by using the HelseID client as described in Section 1.3.

Once an IDP has authenticated the user through the client, the client attempts to obtain more information about the user. First, the client attempts to gather the information from its storage if the user already has an identity stored there. Otherwise, it extracts the information from the Altinn service owner API. Then, roles are delegated per the API response to the concerning user. Finally, the resulting user identity is stored in the database, and it is used in future authentications of the user.

The data must be presented to the end-user in the client. The API response is JSON formatted, and the data is stored in the database using the same manner. As a result, it is effortless to display the information for the end-user in an intuitive, simple way to understand.

If the affiliations of an end-user change, resulting in outdated information in the system's database, the user should be able to update their existing identity in the proof of concept system. The consequences of an affiliation change can be revoked access, gained a higher level of access, or simply obtaining or losing access to an organization. In any case, the user is aware of the change and shall therefore update their identity with an API call to Altinn.

4.2 System Model

From the requirements listed in the previous section, we devise a system consisting of three distinct units. Figure 4.1 presents an abstract overview of these three units.

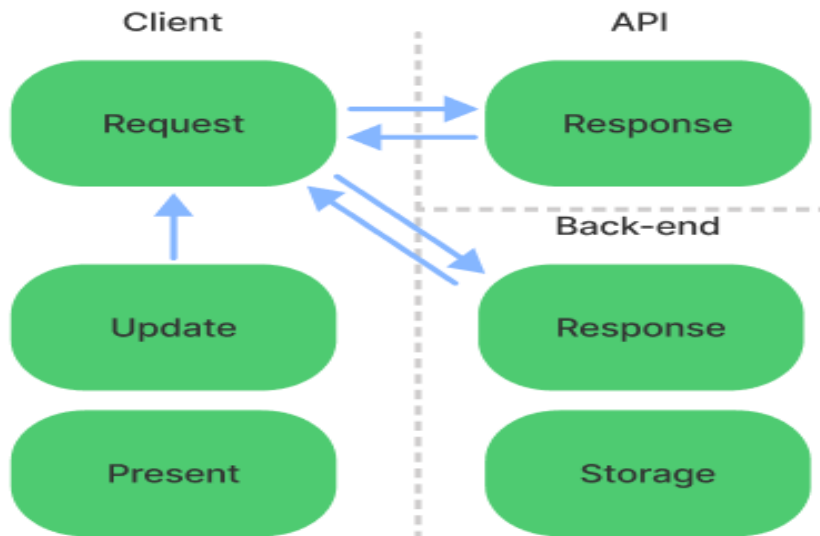


Figure 4.1: Abstract architecture of the system model

4.2.1 The client

The client represents the component with which the end-user interacts. The client will *request* information from the Altinn service owner API automatically when the user is authenticated and the user does not have an existing identity in storage. Then, the client will *present* the received information to the end-user who can interact with it, i.e., access its affiliation's pages.

4.2.2 API

The API component represents the Altinn service owner API. It receives a *request* from the client who asks for information about the end-user. The API *responds* accordingly with the information retrieved from the call. However, it will only reply to an authenticated service owner, meaning the client must be configured with the correct enterprise certificate and API-key.

4.2.3 Back-end

The back-end is the component used for the *storage* of user identities which are created after receiving a response from the API. The client *requests* the authenticated user's user identity, and the back-end *responds* with the identity

if it exists in the database. If not, the client has to *request* information from the API.

The back-end component should contain a cache for maintaining the correctness of the information. However, this is not implemented in the proof of concept system and is considered future work.

4.3 Non-functional requirements

This section discusses the necessary non-functional requirements for developing the proof of concept system [30] according to the problem definition in Section 1.2.

4.3.1 Security and Privacy

The system handles highly personal data, like SSNs; therefore, security and privacy are paramount to uphold. The security and privacy principles, such as authentication, are handled in the HelseID client by the underlying protocols and frameworks, i.e., OAuth2, OIDC, and IdentityServer4. Therefore, the system must handle the credentials safely and securely.

The information retrieved through the API over the web should use an end-to-end encryption protocol such as Hypertext Transfer Protocol Secure (HTTPS) to maintain its security. The data should be encrypted when stored to maintain privacy and security. This thesis only uses fictional data, so it does not encrypt the data when stored for its purposes. However, a realistic implementation should enforce encryption when storing the information.

4.3.2 Reliability and Availability

The system depends on information from the API. The availability of the system vastly increases with the use of database storage. The system's availability will be affected if a user does not have any affiliations in Altinn, and the availability will only affect that user and its access rights. The system will be unavailable because it tries to request information that is not there. Potential future work to overcome this is to allow users to input their desired affiliations, which employees with correct rights can then accept.

Reliability is a metric of the correctness of the information. Since the information is retrieved from trusted state service, it is assumed to be correct. Therefore,

the system does not change the incoming information; it assigns it to the user, presents it, and stores it for later use.

4.3.3 Fault-tolerance

An advantage of using an API developed by a state enterprise is that they make it fault-tolerant. In the event of failure, it will likely recover quickly, perhaps even without other parties noticing a failure. Thus, one can assume there will be no downtime and unavailability for the API. Redundancy in back-end storage can also improve fault tolerance, though this leaves the system vulnerable to inconsistent information. In the event of information inconsistency, one can depend on the API by invoking a call to correct the inconsistency. However, this is not implemented in the system and is considered future work.

4.3.4 Dependency

The system is dependent on the Altinn service owner API for role delegation and to update user affiliations. The API is developed by a trusted state enterprise, and it is under constant and further development. Therefore, it is improbable that the API is terminated in the near future.

It does not rely entirely on the API as the user identities are stored. The customer support center can delegate roles and update affiliations in the case of an API failure deeming it necessary.

4.3.5 Interoperability and Extensibility

The thesis is narrowed down to handle an API that returns responses formatted as JSON. Any API calls either from Altinn or BRREG use JSON as their response format. Therefore, the system must be able to handle the responses using this format.

The system design supports extensibility by making it easy to implement new features. When implementing features using other parts of the API, the developer needs to create a model for the information expected to be received and create a handler that calls the API. The developer does not need to change existing functionalities. The developer can rather use the existing functionalities as inspiration for implementing the new models and handlers.

4.3.6 Usability

The targeted users are health care personnel with varying degrees of computer skills. Due to this fact, the UI must be intuitive and simple to understand. An intuitive design of the UI usually has an inherently high degree of usability. A user survey would be beneficial to verify if the usability of the system and that it has a good UI. However, due to practical limitations, this was not possible to conduct. The system has a simple UI to prove that the role delegation works, and better design of the UI is considered future work.

4.3.7 Scalability and Performance

As NHN's amount of customers increase, so will the number of end-users the system will have to support. Therefore, the scalability of the system is dependant on HelseID and NHN's storage. HelseID is expecting the login platform to be used extensively in the near future; therefore, they aim to improve its scalability [28]. Storage of user identities is already established for the current customer portal. Due to practical reasons, evaluating the system's scalability with many end-users is out of the scope for this thesis.

The performance of a system is defined by the amount of time and resources needed to complete a given task. The performance is dependant on the third-party API, which is out of the system's control. An experiment on the latency of the API is performed in Chapter 6 to verify that the API is sufficient in terms of performance and usability.

Design choices, such as storing user identities, can help reduce the system's latency, thus increasing performance.

4.4 Summary

This chapter has presented the system's functional requirements and presented an abstract architecture based upon those requirements. It also stated the non-functional requirements, which are several criteria and limitations the system must adhere to.

/5

Design and Implementation

In this chapter, the design and implementation of the proof of concept system are presented. The design and implementation are based on the background information in Chapter 2, the need for a unified login system as discussed in Chapter 3, and the functional and non-functional requirements presented in Chapter 4.

The system model was presented in Section 4.2, which shows that there are three distinct components. The system description follows this template.

5.1 Determining the API

When designing and specifying the requirements for the system, both Altinn's and BRREG's APIs were explored. Both APIs allow the extraction of SSNs, roles, and affiliations of a given person or organization. However, the requirements and how to access the information differs between the two APIs.

Brønnøysundregistert's API

BRREG's API [7, 9] demonstrated it possible to extract the SSN for employees with signature and procurator rights from an organization's entry. To obtain the SSN of an organization's employee, one must be authenticated through the organization's enterprise certificate. This requires the user to have access to the organization's enterprise certificate. To utilize the enterprise certificate, the user must install the certificate in their browser, log in with the correct certificate, and NHN must store the organization user safely and securely for future use.

From the user's point of view, this could be a tedious and difficult task as they are likely not proficient with computers. However, it would only allow qualified individuals to gain access because enterprise certificates should only be shared among the heads of an organization.

Using the enterprise certificate, it would be possible to delegate roles using BRREG's API. NHN would have to trust that the certificate is strictly in the hands of an individual in charge of the organization. Oftentimes, health care principals outsource the installation of enterprise certificates to a third-party service provider such as CompuGroup Medical (CGM) [18] because the employees at the organization usually are not proficient enough with computers. CGM is a verified third-party supplier for NHN [27], which does establish trust in this example, but that might not be the case for all other third-party service providers. Another downside with this possible system is that it would have to support login with an enterprise certificate, which HelseID currently does not. Due to these issues, the Altinn service owner API is also explored.

Altinn's API

Altinn's service owner API [3] allows service owners access to excerpt information from organizations registered in Altinn. The API offers various features such as extracting all the information about an organization, its reportees, and what roles the reportees have. It is also possible to excerpt the exact roles and organizations any reportee belongs to. This can all be done in the client, and the user just has to log in with their credentials. All that is required on the client is proof that it is a service owner. The proof comes in the form of an API key and an enterprise certificate. The API key is issued by Altinn, and the enterprise certificate belongs to the organization that is a service owner. A client configured with the proof can utilize the entire service owner API. Because of the simplicity in using the API both for the user and the developer, the Altinn service owner API is chosen for this system.

5.2 The client

The client represents the front-end and login system. The client authenticates the end-user, and the user interacts with it and views its rights access information. In addition, the client is the component that displays the information to the user. Currently, the system does not offer many functionalities to perform on this information. However, it does offer the opportunity to replace the current login solution in the customer portal.

The client is based on the MVC software design pattern. MVC is one of the most frequently used industry-standard web development frameworks to create scalable and extensible applications¹. It consists of three logical components; model, view, and controller.

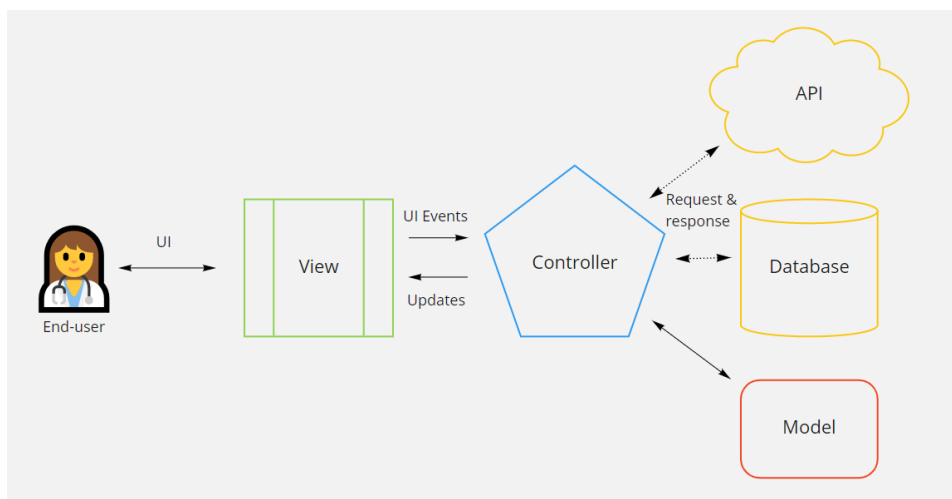


Figure 5.1: Presentation of the MVC design pattern.

In the MVC design pattern, the Model defines how the data should be represented, how to handle the data, and determines the application's state. In addition, the model defines how the information is represented in storage. The system has two important models; the user model and the reportee model. As the names state, they represent the end-users information, and the organizational information, respectively.

The View component is utilized for handling the UI. This is the component that the user interacts with, and it displays everything the user sees on the application, like the user model, given the current view. It consists of several views of various pages, for instance, the main page or an account details

1. https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm

page.

The Controller component acts as an interface between the model and view components. It handles the processing of all business logic. The controller handles communication with the database unit for storage, and it is the component that requests API calls and handles the response. It operates on the data from the model and interacts with the view to render the output for the user to see.

The client is a HelseID test client that is supplied by NHN. The client uses the HelseID test Universal Resource Location (URL) as its authority, and a client identity and client secret are supplied to set up the client. The client supports a range of scopes. The scopes include the default scopes in OIDC and some HelseID scopes relating to identity claims about the user. Additionally, The HelseID self-service team provided the API key and NHN test enterprise certificate to configure the HTTP client to support the service owner API.

HTTP client configuration

```
// Gets or sets the base URI for the ServiceOwner API endpoints, e.g. https://tt02.altinn.no/api/serviceowner/.
[Required]
1 reference
public string ServiceOwnerServiceUri { get; set; } = "https://tt02.altinn.no/api/serviceowner/";

// Gets or sets the API key to use when calling the ServiceOwner API.
[Required]
1 reference
public string ServiceOwnerApiKey { get; set; } = "████████████████████████████████████████";

// Configure http client with API endpoint and ApiKey
1 reference
public void ConfigureHttpClient(HttpClient httpClient)
{
    httpClient.BaseAddress = new Uri(ServiceOwnerServiceUri);
    httpClient.DefaultRequestHeaders.Accept.Clear();
    httpClient.DefaultRequestHeaders.Accept.ParseAdd("application/json");
    httpClient.DefaultRequestHeaders.Add("ApiKey", ServiceOwnerApiKey);
}

// Http message handler responsible for handling enterprise certificate
1 reference
public HttpResponseMessage CreateHttpMessageHandler()
{
    var cert = FindCertificate();
    var clientHandler = new HttpClientHandler { ClientCertificateOptions = ClientCertificateOption.Manual };
    clientHandler.ClientCertificates.Add(cert);
    return clientHandler;
}
```

Figure 5.2: Code snippet for configuration of the HTTP client to support the API. API key redacted.

The configuration setup of the HTTP client can be seen in Figure 5.2. It is configured to support the Altinn service owner API. First, the API key and base Uniform Resource Identifier (URI) for the API endpoint is defined. Then, they are added to the HTTP client. Lastly, the NHN test enterprise certificate is extracted from the local machine and added to the HTTP message handler. The client is initialized in the startup file using these functions as parameters.

The HTTP client allows the HelseID client to send requests to the Altinn API and receive responses.

Login page








Testklient for Thomas Fagermyr sin masteroppgave	
VELG ELEKTRONISK ID	
	ID-PORTEN Logg inn med ID-porten
	BUYPASS Logg inn med Buypass
	COMMFIDES MED JAVA Logg inn med Commfides med java
	COMMFIDES UTEN JAVA Logg inn med Commfides uten java
	HELSE MIDT Logg inn med Helse Midt
	HELSE SØR-ØST Logg inn med Helse Sør-Øst
TEST	TEST IDP Logg inn med Test IDP
	

Figure 5.3: Test client login page displaying IDPs

The HelseID login page of the system is displayed in Figure 5.3. It is the first page the user sees when accessing the system, and it presents the user with a variety of external and internal IDPs to authenticate through. The external IDPs are, such as ID-porten and Buypass, are familiar to most, if not all, health care personnel. Internal IDPs refer to regional providers who are aimed at specific regions and are known in those regions. Since this is a test client, it also has the test IDP option at the bottom. This allows the developer to log in using virtually any valid SSN to test the application.

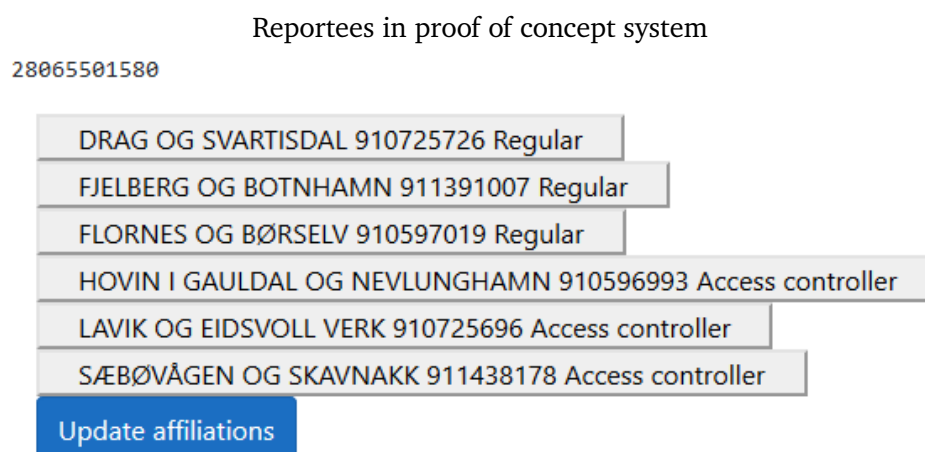


Figure 5.4: Displayed reportees for subject 28065501580 in proof of concept system.

Figure 5.4 provides an overview of a subject and its affiliations in the system. This is the view the user is presented with once they have logged in. In the figure, the subject has logged onto the system through HelseID test IDP, and its affiliations have been gathered.

When a user logs in, the controller is triggered to request the user's affiliations. If the user does not have an entry in the database, then the API calls are invoked. Two requests are issued to Altinn for the user; one for all affiliate roles and the other for the access controller role. This ensures that the user will have all of its roles.

As observed in the figure, roles have been delegated according to the API responses. The current implementation of the system only delegates two roles; regular and access controller. The regular role represents a standard role with limited access within the corresponding organization and is delegated to any user affiliation. The access controller role dictates a user who has full access to the organization, and it is delegated according to the access controller API response. There can be multiple people with the access controller role for an affiliation [25], and the roles define what privileges a user has regarding the

affiliation data [29].

The "Update affiliations" button is a placeholder action. The general idea for this function is to allow users to update their own affiliations when they know that their Altinn affiliations have changed. This function is considered future work.

5.3 API

The proof of concept system consists of a HelseID test client, which utilizes the Altinn service owner API to verify a user's affiliation. The client is configured with an API key and an enterprise certificate to access the Altinn service owner API in its test environment. It also utilizes a database to maintain the user identity. The database is used to limit the number of API calls required by the system. The user identities are stored in the database along with their roles and affiliations.

The complexity of implementing a system that utilizes the service owner API is simpler than implementing the API from BRREG. This is because the API from BRREG requires a log-in with an enterprise certificate for authentication, which the client does not support. Besides this fact, the implementation of both APIs is fairly similar. The biggest difference is the ease of use for the user. When using the Altinn API, the user just has to log in with their own credentials, and the client handles the acquisition of role delegation information.

The service owner API consists of a vast range of various API calls. The reportee call is the most relevant for this system, so the focus of this section will be on that specific API call. The reportee API call from Altinn's service owner API is used by the proof of concept to prove that it is possible to delegate roles using assignments data from a third party.

Altinn service owner - reportee API

Request Information		
Parameters		
Name	Description	Additional information
subject	The social security number of the user to retrieve the reportee list for.	Define this parameter in the request URI .
serviceCode	ServiceCode of the service the result should be filtered on. Optional	Define this parameter in the request URI .
serviceEdition	ServiceEdition of the service the result should be filtered on. Optional	Define this parameter in the request URI .
roleDefinitionId	The RoleDefinitionId the result should be filtered on. Optional. NB: Role definition ids can be different between production and test environment for the same role. Use the general role definitions API endpoint to retrieve/verify correct role definitions.	Define this parameter in the request URI .
queryOptions	OData Query details. This object is created by the Framework based on parameters in the url.	None.
showConsentReportees	Whether reportee from consent delegated rights should be included in the list.	Define this parameter in the request URI .

Figure 5.5: Request information with description about the reportee call. Figure from Altinn.²

Figure 5.5 presents the parameters in the reportee API call, along with descriptions and additional information about the various parameters. For example, the request must contain the query parameter "?ForceEIAuthentication" to enforce authentication through the enterprise certificate. Without the query parameter, the authentication fails, and the requesting client will receive a status code referring to an error in retrieving the information.

The subject is the current user based on its SSN. The parameters serviceCode and serviceEdition are optional arguments that can be used to determine the specific service one is after, for instance, HelseID self-service. The service parameters can be useful but are not required to obtain the current user's roles and affiliations. Finally, the roleDefinitionId is an optional parameter that allows for requesting specific roles. For example, it allows the system to request the access controller role of the given subject specifically. For the test environment, this id is 4, and it is utilized by the proof of concept only to obtain the affiliations of a user where the user has the access controller role.

2. https://www.altinn.no/api/serviceowner/Help/Api/GET-serviceowner-reportees_subject_serviceCode_serviceEdition_roleDefinitionId_showConsentReportees

Reportee API example



Figure 5.6: Reportee API call in the Altinn test environment using Postman.³

Figure 5.6 displays an example of the reportee call made in Postman. It requests to get the list of reportee entities that the subject can represent. The subject is a fictitious test person, and the requested role is the access controller. The subject is configured in Altinn to have access controller rights for a few organizations and itself. The response will list all the entities that the given subject can represent as an access controller. Figure 5.7 shows the corresponding response.

³. <https://www.postman.com/>

Reportee API example response

```
1  |
2  |   "_links": {
3  |     "self": {
4  |       "href": "https://tt02.altinn.no/api/serviceowner/reportees?subject={subject}"
5  |     }
6  |   },
7  |   "_embedded": {
8  |     "reportees": [
9  |       {
10 |         "Name": "ASKILDSEN BENDIK",
11 |         "Type": "Person",
12 |         "SocialSecurityNumber": "24065500317"
13 |       },
14 |       {
15 |         "Name": "DRAG OG SVARTISDAL",
16 |         "Type": "Business",
17 |         "OrganizationNumber": "910725726",
18 |         "ParentOrganizationNumber": "910597019",
19 |         "OrganizationForm": "BEDR",
20 |         "Status": "Active"
21 |       },
22 |       {
23 |         "Name": "FJELBERG OG BOTNHAMN",
24 |         "Type": "Enterprise",
25 |         "OrganizationNumber": "911391007",
26 |         "OrganizationForm": "AS",
27 |         "Status": "Active"
28 |       },
29 |       {
30 |         "Name": "FLORNES OG BØRSELV",
31 |         "Type": "Enterprise",
32 |         "OrganizationNumber": "910597019",
33 |         "OrganizationForm": "AS",
34 |         "Status": "Active"
35 |       }
36 |     ]
37 |   }
38 | }
```

Figure 5.7: Reportee API response from the Altinn test environment using Postman.⁴

As observed in the figure, the response provides information about what and whom the subject can represent. To illustrate, the subject can represent three fictitious organizations and himself as an access controller. This reflects that this user has access to three organizations as an access controller. Once the response has been received, access is granted to the user. All responses for test subjects with Altinn affiliations are displayed in Appendix A.

This API provides ease of use for the end-user who simply log in to the system, and its affiliations will be gathered from Altinn. The HelseID client uses simple and known login options through IDPs that most health care personnel are familiar with. The API is triggered with its required scopes and parameters

4. <https://www.postman.com/>

when the user logs in, as long as the user does not have an existing identity in the database. The user is then given access according to the API response.

5.4 Back-end

The back-end is the storage component of the system. It consists of a SQL Server to maintain and store user identities. The main reason for having the storage component is to increase the system's performance and fault-tolerance and reduce the number of requests sent to Altinn. For example, if the API is unreachable or has restricted access, and the system experiences this, the system is relatively unavailable. However, if users have previously been authenticated, their identities will be stored in the database, and they will not experience unavailability; thus, storage increases the system's fault-tolerance. In addition, having replication of the information further increases fault-tolerance. Storage also helps decrease the load performed on the API, and improving performance by accessing user information quicker.

In this system, the storage consists of a basic SQL Server database. Realistically, the database would consist of one or more of NHN's databases. In addition, the back-end component should contain a cache for maintaining information and reassuring that the information is correct and up-to-date. However, this is not implemented in the proof of concept system and is considered future work.

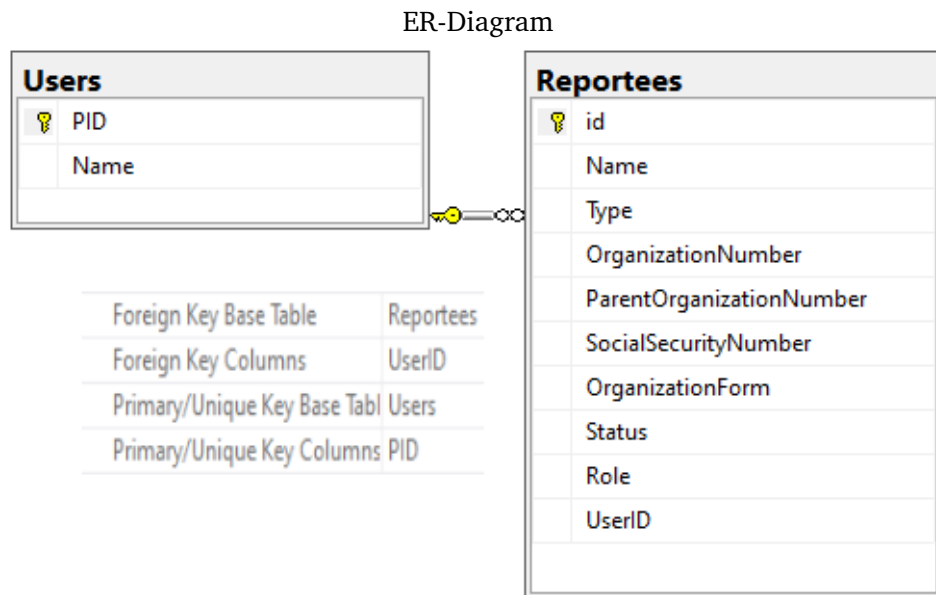


Figure 5.8: ER-Diagram of model relations

Figure 5.8 presents the database models and their relations in terms of a primary and foreign key. The user model consists of the user's name, PID, and a list of reportees. The user's primary key is its PID, and the primary key of the reportees is an automatically generated identification in the form of a number. The reportee list consists of a list of reportee models. This model contains useful information about the organization, such as name and organization number. In addition, an affiliation is added to a user's reportee list, which includes the user's PID and role tied to the given organization.

It is designed this way because a user can have several affiliations with various roles. This way, it is easy to separate them, and it makes it effortless to find all affiliations of a given user.

5.5 Summary

This chapter presented the design and implementation of the proof of concept system. It describes the system's three main components and how they interact to tackle the problem defined in Section 1.2. The client utilizes an API from Altinn, which provides third-party role assignments data for role delegation in

4. <https://www.nhn.no/samhandlingsplattform/helseid/hva-er-helseid>

the system. The user identities are stored in the back-end in the form of an SQL server.

/6

Evaluation and Results

This chapter presents the methods used to evaluate the proof of concept system. Experiments are conducted, results gathered, and discussed in regards to the specifications listed in Chapter 4.

6.1 Methodology and Methods

As stated in Section 1.4, the system follows the **Design** paradigm. The thesis uses the prototyping [23] design process, which is the optimal approach for a proof of concept system. First, the process starts with an idea, which shapes the entire system. The idea provides an abstract overview of the problem and starts the process of defining a solution for solving the problem. Then, the design and architecture of the system must be defined, containing all the necessary components to solve the problem in regards to the stated specifications. Lastly, the system is implemented according to the requirements specifications, and the system is evaluated.

As stated in Section 1.2, the system is a proof of concept, meaning its purpose is to prove that the general idea is functional. Therefore, evaluations of the system are done to investigate that the idea is, in fact, functional and that it fulfills the problem statement.

6.2 Experiments

All experiments were carried out on the following hardware and software:

- Dell Latitude 7390 with an Intel Core i5-8350U CPU 4C/8T @ 1.70GHz
- 8GB RAM
- Microsoft Windows 10 Enterprise
- Tested on Mozilla Firefox Version 68.9.oesr (32-bit) at the URL localhost on port 35454

The experiments demonstrate the different API responses in conjunction with the subject. The API's latency is benchmarked to obtain a better understanding of its performance. A variety of scenarios are evaluated. The various API requests include:

- A registered subject in Altinn with affiliations and the access controller role.
- Registered subject with affiliations but not the access controller role.
- Subject not registered in Altinn.

6.2.1 Role delegation experiments

According to the problem statement in Section 1.2, the system should automatically assign the appropriate role to any given user as long as they have affiliations registered in Altinn. NHN only has two known test subjects, 24065500317 and 28065501580, with affiliations in the Altinn test. The first subject will be used for testing and displaying results in the first experiment. The third subject, 15037104229, is a commonly known test subject. However, the subject is not registered in Altinn's test environment and will be used for testing in the second experiment.

In the **first experiment**, it is tested how a subject with several affiliations with a variety of roles in Altinn is represented in the system and how the various roles are delegated from the retrieved information. The figures below present how affiliations are viewed in Altinn and the implemented system.

Reportees in Altinn

Alle dine aktører Se alle underenheter Se slettede enheter

 ASKILDSSEN BENDIK Fødselsnr. 240655 00317
 FJELBERG OG BOTNHAMN Org.nr. 911 391 007
 FLORNES OG BØRSELV Org.nr. 910 597 019
 1 underenheter
DRAG OG SVARTISDAL Org.nr. 910 725 726
 HOVIN I GAULDAL OG NEVLUNGHAMN Org.nr. 910 596 993
 1 underenheter
LAVIK OG EIDSVOLL VERK Org.nr. 910 725 696
 SÆBØVÅGEN OG SKAVNAKK Org.nr. 911 438 178

Figure 6.1: List of reportees for subject 24065500317 in Altinn.

As observed in Figure 6.1, the subject has seven affiliations in total. It does not provide which roles he has. However, it is possible to access each entry to manage the organization and view the rights. In addition, the subject has an entry for himself, which will be excluded from the system since it only handles the organizational reportees. Four of the affiliations are organizations, with two of them having a single subunit each; thus, making them affiliates of the subject, giving the subject a total of six organizational affiliations.

Reportees in proof of concept system

24065500317

DRAG OG SVARTISDAL 910725726 Access controller
FJELBERG OG BOTNHAMN 911391007 Access controller
FLORNES OG BØRSELV 910597019 Access controller
HOVIN I GAULDAL OG NEVLUNGHAMN 910596993 Regular
LAVIK OG EIDSVOLL VERK 910725696 Regular
SÆBØVÅGEN OG SKAVNAKK 911438178 Regular

Figure 6.2: List of reportees for subject 24065500317 in proof of concept system.

Figure 6.2 provides a view of how the six affiliations are represented in the system. The figure displays that the user has access to the corresponding affiliations in conjunction with those shown in Figure 6.1. From the API response, the user has been delegated three regular roles. The "regular" role represents the standard role of the system with the most basic rights. In addition, three access controller roles have also been delegated to the subject.

In the **second experiment**, a subject that is not registered in Altinn's test environment is evaluated. The fact that the subject is not registered translates to the subject not having any affiliations to delegate a role for. The subject logs onto the client, which attempts to request organizational information about the subject.

Status: 400 Invalid social security number: 15037104229

Figure 6.3: API request for unregistered subject 15037104229

Figure 6.3 shows the API response when requesting the reportees for the given subject. As seen, the subject is not registered; thus, generating an "invalid SSN" response.

Subject not registered in Altinn

15037104229

You currently have no affiliations.

Request access to an organization by entering the organization number in the field above.

Figure 6.4: Response to the unregistered subject in the proof of concept system.

The user can log onto the client with its credentials, but as observed in Figure 6.4, the user will not be delegated any roles. Furthermore, since the user is not registered in Altinn, it does not have any affiliations registered either. Therefore, the user has not delegated any roles, as proven in the result.

In this case, the user is given the option to request access manually. The textbox and submit button are placeholders, as this is not implemented and is considered future work. In short, the idea is that the subject requests access which an access controller can grant for a given affiliation.

6.2.2 API Benchmark

The time it takes to load any element of a website or computer system must be short enough for the user to focus on the task at hand. The optimal speed of operations makes the user feel as if the task happens instantaneously. According to Jacob Nielsen in his book Usability Engineering [19], three important time limits have remained the same for a long period of time. The three limits are 0,1 seconds, 1 second, and 10 seconds. When the load time is 0,1 seconds, the user feels the system is instantaneous. At a 1 second limit, the user usually notices the delay, but it is short enough to keep its attention. At 10 seconds, the user's attention is usually lost, and they commit to other tasks while waiting for the computer to finish its task.

The general consensus is that 1 second is the maximum limit before users feel like the loading time is slow and begin to lose their attention. Therefore, when benchmarking the API a 1000 milliseconds were chosen as a maximum response time. Anything higher than 1 second would fail the test. The benchmark test consists of 100 iterations of the request displayed in Figure 5.6, measuring the response time. The test was run 10 times to get an average response time and exclude skewed results. Figure 6.5 displays the result of the benchmark.

API latency benchmark

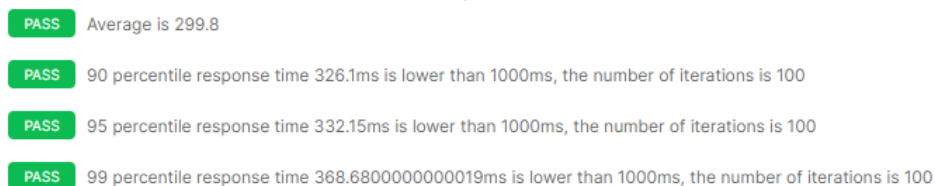


Figure 6.5: Latency benchmark of reportee API call done in Postman.

The results show that all the tests were passed. Furthermore, the API has an average latency of 299.8 milliseconds which is well under the 1-second limit. The percentile response times represent the highest latency in the top 10, 5, and 1 percentages. The highest response time, 368.68ms, is also well under the 1-second limit. The results prove that the performance of the API is good and more than sufficient enough to be used in the proof of concept system.

6.3 Summary

This chapter has evaluated the system to determine if it is feasible through proof of concept. It demonstrates that the system automatically delegates correct roles on login, in correspondence with the information from API, for any subject with affiliations registered in Altinn. Furthermore, the API performs well, with an average response time of 299.8 milliseconds.

/7

Conclusion

This chapter presents the achievements of the thesis, provides some concluding remarks, and outlines potential future work.

7.1 Achievements

In this thesis, a proof of concept system has been designed, implemented, and evaluated for role delegation using HelseID combined with third-party role assignments data. The problem definition stated the following in Section 1.2:

This thesis proposes a proof of concept system that aims to solve the problem of outdated rights access information and missing or forgotten login information for NHN's customers. The goal is to solve this problem by integrating HelseID [1] with third-party role assignments data from BRREG's APIs [7, 9] or Altinn's service owner API [3]. HelseID allows health care personnel to log in using their SSN through a well-known IDP such as BankID [6] or Buypass. With these APIs, the system will automatically assign the appropriate role to any users given they have registered affiliations, thus alleviating NHN from creating and delegating users.

With all of NHN's customers and their personnel changing, it is easy for access rights information to get outdated. Especially since most customers usually do

not report such changes. The proof of concept system provides a solution by utilizing an API to extract organizational roles from a third-party service.

The solution lies in designing and implementing a system that utilizes HelseID integrated with an API from a third-party service to request role assignments data. The proof of concept system authenticates its users with well-known IDPs through the HelseID client and automatically requests access rights information for the given user. The client issues a request to Altinn's service owner API with the user's SSN as the subject to acquire the access rights information. Roles are delegated for the user according to the API response, and the user identity is stored in a back-end for fault-tolerance and future use. The evaluations of the system show that it can delegate roles for any subject according to its registration in Altinn.

7.2 Concluding remarks

Through a proof of concept system, this thesis has shown that HelseID, in combination with Altinn's service owner API, makes it possible to dynamically maintain access rights of healthcare principals. It also demonstrates that it is possible to alleviate the traditional username and password accounts by using HelseID, which authenticates the user through known IDPs using SSNs.

One of the main issues when designing the system was to decide which third-party service to utilize. Two different APIs, serviced by BRREG's and Altinn, were explored to solve this task. Altinn's service owner API was chosen for the task as it was more user-friendly and easy to implement. Furthermore, the Altinn API offers a great opportunity for further development of the system and its functionalities.

7.3 Future Work

HelseID, with the use of the Altinn service owner API, offers the possibility for great features that can improve the user experience in NHN's customer portal and other NHN platforms. With the addition of the API, organizations can potentially be entirely self-sufficient in administering home office accounts, e-mail accounts, and perhaps even virtual meeting rooms. Any organization that has set up an access controller role in Altinn, and is a customer of NHN, can independently obtain the same access in this system. The access controller will have full access to the membership of the organization and all of its existing and upcoming features.

The access controller can grant or revoke access for other employees and delegate roles as they see fit. The API can be used to delegate other roles besides the access controller or regular role. The API offers the opportunity to extract any Altinn role; thus, allowing a developer to create their own role hierarchy that can delegate roles in correspondence with API responses.

A user that is a part of an organization, but does not have an affiliation or a role in Altinn, should have the option to input their desired organizations. From there, the access controller will have the option to grant or deny access to the potential employees. This feature can make the system fault-tolerant and increase its availability. The proof of concept system has a placeholder button for this feature, as seen in Section 5.2.

The proof of concept system does not focus on the UI. It only displays the information for the user to demonstrate that it works. The UI can be further developed to look better, provide more insight for the user, and add more general functionality.

In the future, caching should be implemented in the database used by the system. Caching would greatly help reduce outdated information by requiring older information to be verified after a certain amount of time, making the information more reliable and available for the user.



JSON Response from Altinn API

Response data from Altinn service owner API for all requests regarding test subjects with Altinn affiliations.

Test subject, 15037104229, without Altinn affiliation receives HTTP status 400 Invalid social security number.

A.1 Subject 24065500317

A.1.1 With roleDefinitionId as access controller (4)

```
{
  "_links": {
    "self": {
      "href":
        "https://tt02.altinn.no/api/serviceowner/reportees?subject={subject}"
    }
  },
  "_embedded": {
    "reportees": [
```

```

    {
      "Name": "ASKILDSEN BENDIK",
      "Type": "Person",
      "SocialSecurityNumber": "24065500317"
    },
    {
      "Name": "DRAG OG SVARTISDAL",
      "Type": "Business",
      "OrganizationNumber": "910725726",
      "ParentOrganizationNumber": "910597019",
      "OrganizationForm": "BEDR",
      "Status": "Active"
    },
    {
      "Name": "FJELBERG OG BOTNHAMN",
      "Type": "Enterprise",
      "OrganizationNumber": "911391007",
      "OrganizationForm": "AS",
      "Status": "Active"
    },
    {
      "Name": "FLORNES OG BØRSELV",
      "Type": "Enterprise",
      "OrganizationNumber": "910597019",
      "OrganizationForm": "AS",
      "Status": "Active"
    }
  ]
}

```

A.1.2 Without role specification

```

{
  "_links": {
    "self": {
      "href":
        "https://tt02.altinn.no/api/serviceowner/reportees?subject={sub}
    }
  },
  "_embedded": {
    "reportees": [
      {

```



```
    "Name": "ASKILDSEN BENDIK",
    "Type": "Person",
    "SocialSecurityNumber": "24065500317"
  },
  {
    "Name": "DRAG OG SVARTISDAL",
    "Type": "Business",
    "OrganizationNumber": "910725726",
    "ParentOrganizationNumber": "910597019",
    "OrganizationForm": "BEDR",
    "Status": "Active"
  },
  {
    "Name": "FJELBERG OG BOTNHAMN",
    "Type": "Enterprise",
    "OrganizationNumber": "911391007",
    "OrganizationForm": "AS",
    "Status": "Active"
  },
  {
    "Name": "FLORNES OG BØRSELV",
    "Type": "Enterprise",
    "OrganizationNumber": "910597019",
    "OrganizationForm": "AS",
    "Status": "Active"
  },
  {
    "Name": "HOVIN I GAULDAL OG NEVLUNGHAMN",
    "Type": "Enterprise",
    "OrganizationNumber": "910596993",
    "OrganizationForm": "ASA",
    "Status": "Active"
  },
  {
    "Name": "LAVIK OG EIDSVOLL VERK",
    "Type": "Business",
    "OrganizationNumber": "910725696",
    "ParentOrganizationNumber": "910579959",
    "OrganizationForm": "BEDR",
    "Status": "Active"
  },
  {
    "Name": "SÆBØVÅGEN OG SKAVNAKK",
    "Type": "Enterprise",
```

```

        "OrganizationNumber": "911438178",
        "OrganizationForm": "AS",
        "Status": "Active"
    }
  ]
}

```

A.2 Subject 28065501580

A.2.1 With roleDefinitionId as access controller (4)

```

{
  "_links": {
    "self": {
      "href":
        "https://tt02.altinn.no/api/serviceowner/reportees?subject={sub
    }
  },
  "_embedded": {
    "reportees": [
      {
        "Name": "HOVIN I GAULDAL OG NEVLUNGHAMN",
        "Type": "Enterprise",
        "OrganizationNumber": "910596993",
        "OrganizationForm": "ASA",
        "Status": "Active"
      },
      {
        "Name": "HÅGENSEN JEPPE",
        "Type": "Person",
        "SocialSecurityNumber": "28065501580"
      },
      {
        "Name": "LAVIK OG EIDSVOLL VERK",
        "Type": "Business",
        "OrganizationNumber": "910725696",
        "ParentOrganizationNumber": "910579959",
        "OrganizationForm": "BEDR",
        "Status": "Active"
      },
      {

```

```

        "Name": "SÆBØVÅGEN OG SKAVNAKK",
        "Type": "Enterprise",
        "OrganizationNumber": "911438178",
        "OrganizationForm": "AS",
        "Status": "Active"
    }
  ]
}

```

A.2.2 Without role specification

```

{
  "_links": {
    "self": {
      "href":
        "https://tt02.altinn.no/api/serviceowner/reportees?subject={subject}"
    }
  },
  "_embedded": {
    "reportees": [
      {
        "Name": "DRAG OG SVARTISDAL",
        "Type": "Business",
        "OrganizationNumber": "910725726",
        "ParentOrganizationNumber": "910597019",
        "OrganizationForm": "BEDR",
        "Status": "Active"
      },
      {
        "Name": "FJELBERG OG BOTNHAMN",
        "Type": "Enterprise",
        "OrganizationNumber": "911391007",
        "OrganizationForm": "AS",
        "Status": "Active"
      },
      {
        "Name": "FLORNES OG BØRSELV",
        "Type": "Enterprise",
        "OrganizationNumber": "910597019",
        "OrganizationForm": "AS",
        "Status": "Active"
      }
    ]
  }
}

```

```
{
  {
    "Name": "HOVIN I GAULDAL OG NEVLUNGHAMN",
    "Type": "Enterprise",
    "OrganizationNumber": "910596993",
    "OrganizationForm": "ASA",
    "Status": "Active"
  },
  {
    "Name": "HÅGENSEN JEPPE",
    "Type": "Person",
    "SocialSecurityNumber": "28065501580"
  },
  {
    "Name": "LAVIK OG EIDSVOLL VERK",
    "Type": "Business",
    "OrganizationNumber": "910725696",
    "ParentOrganizationNumber": "910579959",
    "OrganizationForm": "BEDR",
    "Status": "Active"
  },
  {
    "Name": "SÆBØVÅGEN OG SKAVNAKK",
    "Type": "Enterprise",
    "OrganizationNumber": "911438178",
    "OrganizationForm": "AS",
    "Status": "Active"
  }
]
}
```

References

- [1] Steinar Noem Alex Somby and Håvard Wang. Helse id dokumentasjon. Web site v1.8, Norsk Helsenett SF, <https://nhn.no/helseid/>, dec 2020. accessed 6 may 2021.
- [2] Altinn. About altinn 3. Web site, Digitaliseringsdirektoratet, <https://docs.altinn.studio/teknologi/altinnstudio/about/>. accessed 25 May 2021.
- [3] Altinn. Altinn service owner api. Web site, Digitaliseringsdirektoratet, <https://www.altinn.no/api/serviceowner/Help>, 2021. accessed 27 July 2021.
- [4] Altinn. Om altinn. Web site, Digitaliseringsdirektoratet, <https://www.altinn.no/om-altinn/>, 2021. accessed 25 may 2021.
- [5] Altinn. Om altinn samarbeidet. Web site, Digitaliseringsdirektoratet, <https://www.altinndigital.no/om-altinn/om-altinn-samarbeidet/>, 2021. accessed 31 August 2021.
- [6] BankID. Om oss. Web site, Vipps AS, <https://www.bankid.no/privat/om-oss/>, 2021. accessed 5 May 2021.
- [7] Brønnøysundregisterene. Fullmaktjenesten: Api-dokumentasjon. Web site v1.0, <https://data.brreg.no/fullmakt/docs/index.html>, jul 2021. accessed 30 April 2021.
- [8] Brønnøysundregisterene. Home page. Web site, <https://www.brreg.no/>, 2021. accessed 23 March 2021.
- [9] Brønnøysundregisterene. Åpne data - enhetsregisteret: Api-dokumentasjon. Web site v1.0, <https://data.brreg.no/enhetsregisteret/api/docs/index.html>, jul 2021. accessed 30 April 2021.

- [10] Douglas E Comer, David Gries, Michael C Mulder, Allen Tucker, A Joe Turner, and Paul R Young. Computing as a discipline. *Communications of the ACM*, 32(1):9–23, 1989.
- [11] Curity. Scopes vs claims. Web site, <https://curity.io/resources/learn/scopes-vs-claims/>, 2017. accessed 17 March 2021.
- [12] Roy T Fielding and Richard N Taylor. Principled design of the modern web architecture. *ACM Transactions on Internet Technology (TOIT)*, 2(2):115–150, 2002.
- [13] Roy Thomas Fielding. *Architectural styles and the design of network-based software architectures*. University of California, Irvine, 2000.
- [14] Dick Hardt. The OAuth 2.0 Authorization Framework. RFC 6749, October 2012.
- [15] Splunk Inc. The data-to-everything platform. Web site, <https://www.splunk.com/>, 2005. accessed 9 April 2021.
- [16] Simen Lomås Johannessen. Girji. metacode extensibility in girji. Master’s thesis, UiT Norges arktiske universitet, 2014.
- [17] Ida Jaklin Johansen. Láhttu-a system for retrieval and consolidation of personal data from activity-tracking web services. Master’s thesis, UiT Norges arktiske universitet, 2014.
- [18] CompuGroup Medical. Hva er et virksomhetssertifikat? Web site, https://www.cgm.com/nor_no/artikler/artikler/virksomhetssertifikat-hva-er-dette-og-hvorfor-trenger-vi-det-1.html, nov 2020. accessed 4 July 2021.
- [19] Jakob Nielsen. *Usability engineering*. Morgan Kaufmann, 1994.
- [20] OpenID. Openid certification. Web site, <https://openid.net/certification/>, 2021. accessed 6 July 2021.
- [21] OpenID. Openid connect: The internet identity layer. Web site, <https://openid.net/connect/>, 2021. accessed 7 July 2021.
- [22] Tutorials Point. Mvc framework - introduction. Web site, https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm, 2021. accessed 5 August 2021.

- [23] Roger S Pressman. *Software engineering: a practitioner's approach*. Palgrave macmillan, 2005.
- [24] regex101. build, test, and debug regex. Web site, <https://regex101.com/>, 2021. accessed 20 April 2021.
- [25] Ravi S Sandhu and Pierangela Samarati. Access control: principle and practice. *IEEE communications magazine*, 32(9):40–48, 1994.
- [26] Norsk Helsenett SF. Kundeportalen. Web site, <https://kunde.nhn.no>, apr 2021.
- [27] Norsk Helsenett SF. Tredjepartsleverandører i helsenettet. Web site, , 2021. accessed 10 August 2021.
- [28] Norsk Helsenett SF. Utbredelse av tjenesten. Web site, <https://www.nhn.no/samhandlingsplattform/helseid/hva-er-helseid/utbredelse-av-tjenesten>, 2021. accessed 27 July 2021.
- [29] Andrew S Tanenbaum and Herbert Bos. *Modern operating systems*. Pearson, 2015.
- [30] Maarten Van Steen and A Tanenbaum. Distributed systems principles and paradigms. *Network*, 2:28, 2002.

